D-CODE: DISCOVERING CLOSED-FORM ODES FROM OBSERVED TRAJECTORIES

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

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

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

ABSTRACT

For centuries, scientists have manually designed closed-form ordinary differential equations (ODEs) to model dynamical systems. An automated tool to distill closed-form ODEs from observed trajectories would accelerate the modeling process. Traditionally, symbolic regression is used to uncover a closed-form prediction function $a = f(b)$ with label-feature pairs (a_i, b_i) as training examples. However, an ODE models the time derivative $\dot{x}(t)$ of a dynamical system, e.g. $\dot{x}(t) = f(x(t), t)$, and the "label" $\dot{x}(t)$ is usually *not* observed. The existing ways to bridge this gap only perform well for a narrow range of settings with low measurement noise and frequent sampling. In this work, we propose the Discovery of Closed-form ODE framework (D-CODE), which advances symbolic regression beyond the paradigm of supervised learning. D-CODE uses a novel objective function based on the variational formulation of ODEs to bypass the unobserved time derivative. For formal justification, we prove that this objective is a valid proxy for the estimation error of the true (but unknown) ODE. In the experiments, D-CODE successfully discovered the governing equations of a diverse range of dynamical systems under challenging measurement settings with high noise and infrequent sampling.

1 INTRODUCTION

An ordinary differential equation (ODE) links the state of a continuous-time dynamical system $x(t)$ to its time derivative $\dot{x}(t)$ via a function f, e.g. $\dot{x}(t) = f(x(t), t)$. The ODE is *closed-form* when f has a concise and analytical expression (e.g. $f(x, t) = -t\log(x)$). Closed-form ODEs are ubiquitous in science and engineering. The popularity is partly due to their transparency and interpretability to human experts (Petersen et al., 2019). Their concise functional form also facilitates the analysis of many key properties of the dynamical system, e.g. asymptotic stability (Lyapunov, 1992).

However, the discovery of closed-form ODEs has been laborious and time-consuming as it heavily relies on human experts (Simmons, 1972). The goal of this work is to discover closed-form ODEs in an automated and data-driven way. We envision that such a tool would accelerate the quantitative modeling of dynamical systems, with applications ranging from discovering the kinetics of biochemical reactions to modeling tumor growth (Schuster, 2019; Geng et al., 2017).

Symbolic regression is an established approach to discover a closed-form function $a = f(b)$ for predicting the label a from the feature b. It operates in the supervised learning setting and requires a dataset of label-feature pairs for training, i.e. $\mathcal{D} = \{a_i, b_i\}_{i=1}^{N}$ (Schmidt & Lipson, 2009).

However, symbolic regression is not directly applicable to ODEs because we usually do *not* observe the time derivative $\dot{x}(t)$, i.e. the labels for regression are not available in the data. In contrast, we only have access to the measurements of state $y(t) = x(t) + \epsilon$ at some discrete time points and with noise. For instance, the datasets for tumor growth modeling typically contain tumor volumes measured at different clinical visits with substantial noise (Wilkerson et al., 2017; Mazaheri et al., 2009). We refer to this problem as the *equation-data mismatch*.

A simple way to resolve this mismatch is to *estimate* $\dot{x}(t)$ from data (e.g. by numerical differentiation (Bickley, 1941)), and use the *estimated* derivatives as labels for regression (Gaucel et al., 2014).

However, it is very challenging to recover the time derivative under high measurement noise or infrequent sampling (Cullum, 1971), and symbolic regression tends to perform poorly with inaccurate labels (Žegklitz & Pošík, 2021). This is an substantial drawback because many applications, such as healthcare, may involve noisy or infrequently sampled data (Jensen et al., 2014).

In this work, we develop the Discovery of Closed-form ODE framework (D-CODE), which extends symbolic regression beyond the supervised learning setting. The key insight behind D-CODE is the variational formulation of ODE (Hackbusch, 2017), which establishes a direct link between the trajectory $x(t)$ and the ODE f while bypassing the unobservable time derivative $\dot{x}(t)$. We develop a novel objective function based on this insight, and prove that it is a valid proxy for the estimation error of the true (but unknown) ODE. We demonstrate via extensive experiments that D-CODE can uncover the governing equations for a diverse range of dynamical systems while being substantially more robust to measurement artifacts than the alternative methods. Finally, D-CODE is designed as a general framework, where some of its components can be flexibly adapted based on the application.

2 Background and problem setting

Curating a dataset of trajectories for ODE discovery involves many decisions, e.g. What variables to include? When and for how long to take measurements? (we discuss dataset curation in Appendix C.) In this work, we assume the dataset is *given* and the variables can be modeled by a system of first-order autonomous ODEs (Eq. 1). We note that higher-order or time-dependent ODEs can be represented in this form by curating a dataset with additional variables (Simmons, 1972) (Appendix B). The system with $J \in \mathbb{N}^+$ variables is defined as

$$\dot{x}_j(t) = f_j(\boldsymbol{x}(t)),\ \forall j = 1, \dots, J,\ \forall t \in [0, T] \tag{1}$$

where we use Newton's notation $\dot{x}_j(t)$ for the time derivative. The functions $f_j : \mathbb{R}^J \to \mathbb{R}$ will be sometimes referred to as the ODEs directly. We denote $T \in \mathbb{R}^+$ as the maximum time horizon we have data for. We highlight the following distinction: the *trajectory* $x_j : [0, T] \to \mathbb{R}$ is a function of time, whereas the *state* $x_j(t) \in \mathbb{R}, \forall t \in [0, T]$ is a point on the trajectory[1]. We denote the state vector $\boldsymbol{x}(t) := [x_1(t), \dots, x_J(t)]^\top \in \mathbb{R}^J$ and the vector-valued trajectory function $\boldsymbol{x} := [x_1, \dots, x_J]$.

Let f_j^*'s be the *true* but *unknown* ODEs to be uncovered, and $\boldsymbol{x}_i : [0, T] \to \mathbb{R}^J$, $i \le N$, $N \in \mathbb{N}^+$ be the true trajectories that satisfy f_j^*'s. In practice, we only measure the true trajectories at discrete times and with noise. Denote the measurement of trajectory i at time t as $\boldsymbol{y}_i(t) \in \mathbb{R}^J$; we assume

$$\boldsymbol{y}_i(t) = \boldsymbol{x}_i(t) + \epsilon_i(t), \quad \forall i \le N,\ t \in \mathcal{T} \tag{2}$$

where $\epsilon_i(t) \in \mathbb{R}^J$ is zero-mean noise with standard deviation σ. The measurements are made at time $t \in \mathcal{T} = \{t_1, t_2, \dots, T\}$. We denote the dataset as $\mathcal{D} = \{\boldsymbol{y}_i(t) | i \le N, t \in \mathcal{T}\}$.

Closed form. The function $f_j : \mathbb{R}^J \to \mathbb{R}$ has a *closed form* if it can be expressed as a finite sequence of operations $(+, \div, \log, \dots)$, input variables $(x_1, x_2, \dots)$ and numeric constants $(1.5, 0.8, \dots)$ (Borwein et al., 2013). The *functional form* of f_j is the expression with all the numeric constants replaced by placeholders θ_k's, e.g. $f_j(x) = \theta_1 x \cdot \log(\theta_2 x)$. To fully uncover f_j^*, we need to infer its functional form and estimate the unknown constants θ_k's (if any).

Variational formulation. The variational formulation provides a direct link between the trajectory $\boldsymbol{x}$ and the ODE f_j *without* involving $\dot{x}$ (Hackbusch, 2017). We start with the following definition.

Definition 1. Consider $J \in \mathbb{N}^+$, $T \in \mathbb{R}^+$, continuous functions $\boldsymbol{x} : [0, T] \to \mathbb{R}^J$, $f : \mathbb{R}^J \to \mathbb{R}$, and $g \in \mathcal{C}^1[0, T]$, where $\mathcal{C}^1$ is the set of continuously differentiable functions. We define the functionals

$$C_j(f, \boldsymbol{x}, g) := \int_0^T f\big(\boldsymbol{x}(t)\big) g(t) dt + \int_0^T x_j(t) \dot{g}(t) dt; \quad \forall j \in \{1, 2, \dots, J\} \tag{3}$$

Importantly, the functional C_j depends on the *testing function* $g(t)$ and its derivative $\dot{g}(t)$ but *not* $\dot{x}_j$. Proposition 1 below provides the variational formulation of ODE. Essentially it specifies the infinitely many constraints that $\boldsymbol{x}$ has to satisfy in order to be a solution to the ODE. (see Appendix A).

[1]We restrict the domain of the function x_j to $[0, T]$ (rather than $\mathbb{R}^+$) because we only have data up to T. However, after learning the ODE, we can extrapolate x_j beyond T (Appendix D).

Table 1: Comparison of related works. "Data": the observed variables. "Allowed f^*": the space of discoverable functions. "Est.": the quantities estimated in the intermediate step. "$\dot{x}$ Free": is the method not reliant on $\dot{x}$? "$\boldsymbol{x}(0)$ Free": is the method not reliant on initial condition $\boldsymbol{x}(0)$? "Objective": the objective function. References: [1] Schmidt & Lipson (2009), [2] Brunton et al. (2016), [3] Gaucel et al. (2014) , [4] Chen et al. (2018).

Method	Data	Allowed f^*	Est.	$\dot{x}$ Free	$\boldsymbol{x}(0)$ Free	Objective
Symbolic Reg [1]	$a, \boldsymbol{b}$	Closed-form	None	-	-	$\|\|a - f(\boldsymbol{b})\|\|_2$
2-step Sparse [2]	$\boldsymbol{y}(t)$	$\sum \theta_k h_k(\boldsymbol{x})$	$\widehat{\dot{x}}$	×	✓	$\sum_t \|\|\widehat{\dot{x}}(t) - f(\boldsymbol{y}(t))\|\|_2$
2-step Symbolic [3]	$\boldsymbol{y}(t)$	Closed-form	$\widehat{\dot{x}}$	×	✓	$\sum_t \|\|\widehat{\dot{x}}(t) - f(\boldsymbol{y}(t))\|\|_2$
ODE Approx [4]	$\boldsymbol{y}(t)$	Neural nets	$\widehat{\boldsymbol{x}}(0)$	✓	×	$\sum_t \|\|\boldsymbol{y}(t) - \widehat{\boldsymbol{x}}(t)\|\|_2$
D-CODE	$\boldsymbol{y}(t)$	Closed-form	$\widehat{\boldsymbol{x}}$	✓	✓	Equation 5

Proposition 1. *(Hackbusch, 2017) Consider $J \in \mathbb{N}^+$, $T \in \mathbb{R}^+$, a continuously differentiable function $\boldsymbol{x} : [0, T] \to \mathbb{R}^J$, and continuous functions $f_j : \mathbb{R}^J \to \mathbb{R}$ for $j = 1, \ldots, J$. Then $\boldsymbol{x}$ is the solution to the system of ODEs in Equation 1 if and only if*

$$C_j(f_j, \boldsymbol{x}, g) = 0,\ \forall j \in \{1, \ldots, J\},\ \forall g \in \mathcal{C}^1[0, T],\ g(0) = g(T) = 0 \tag{4}$$

3 Related Work

3.1 Symbolic Regression

Symbolic regression attempts to uncover a closed-form prediction function $a = f(\boldsymbol{b})$ using supervised learning (Table 1). The main challenge is *optimization*: searching for the optimal f is thought to be NP-hard because the space of closed-form functions is vast and complex—it is combinatorial in the functional form and continuous in the numeric constants (Lu et al., 2016). Therefore, most existing work focuses on developing optimization algorithms. Genetic programming has been one of the most successfully and widely used methods (Koza, 1994; Schmidt & Lipson, 2009). It represents f as a *tree* where the internal nodes are operations and the leaves are variables or constants (Figure 1 B). It then applies genetic algorithm to search for the best tree representation (Forrest, 1993). More recently, AI Feynman introduces a set of heuristics to directly prune the search space, thereby improving the optimization efficiency (Udrescu & Tegmark, 2020). Optimization methods based on reinforcement learning (Petersen et al., 2019), pre-trained neural networks (Biggio et al., 2021), and Meijer G-functions (Alaa & van der Schaar, 2019) have also been proposed.

All methods above use prediction error (e.g. RMSE) as the objective, which is not applicable to ODE discovery because the label (time derivative) is not observed. D-CODE advances symbolic regression beyond the supervised learning setting and addresses the unique challenges for ODE discovery: the equation-data mismatch and the sensitivity to noisy or infrequent observations.

3.2 Data-driven discovery of closed-form ODEs

Two-step sparse regression performs sparse regression on the estimated time derivative $\widehat{\dot{x}}_j$ (Brunton et al., 2016; Rudy et al., 2017). It assumes that the true ODE can be symbolically expressed in a *linear* form: $\dot{x}_j(t) = \sum_{k=1}^{K} \theta_k h_k(\boldsymbol{x}(t))$ where $\theta_k \in \mathbb{R}$ are unknown constants and $h_k : \mathbb{R}^D \to \mathbb{R}$ are *pre-specified* candidate functions, such as monomials $x, x^2, \ldots$ (Table 1). The method uses L1 regularization on θ_k to ensure the learned function only contains a few terms. We review the methods to estimate time derivative in Appendix E.

However, the assumed linear form is very restrictive (e.g. $x^{3/2}$ or $1/(x + 2)$ are not allowed). In fact, it cannot symbolically represent many well-known ODEs (e.g. generalized logistic model). Furthermore, the choice of candidate functions heavily depends on the human experts, which deviates from our goal of automated and data-driven ODE discovery.

Two-step symbolic regression also uses the *estimated* time derivatives $\widehat{\dot{x}}_j$ as the label but applies symbolic regression instead (Gaucel et al., 2014) (Table 1). It may use *any* optimization method discussed in Section 3.1 to search for the optimal function. In this way, it removes the linear assumption on the functional form and can discover closed-form functions in general.

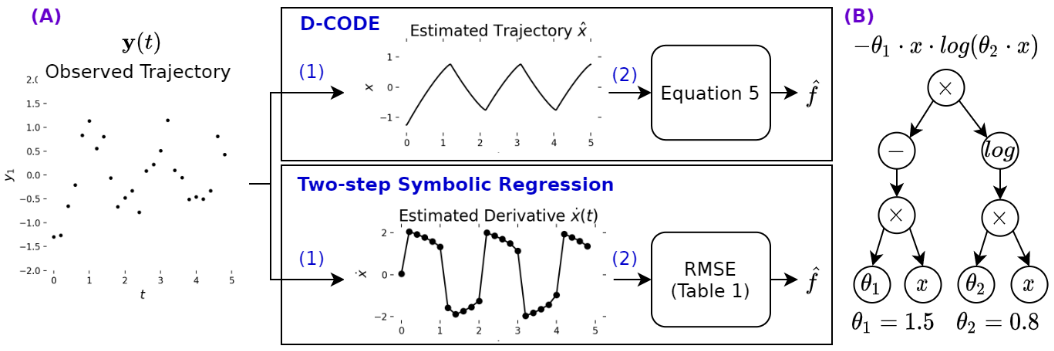

(A)
$\mathbf{y}(t)$
Observed Trajectory
D-CODE
Estimated Trajectory $\hat{x}$
(1)
(2)
Equation 5
$\hat{f}$
Two-step Symbolic Regression
Estimated Derivative $\dot{x}(t)$
RMSE (Table 1)
(B)
$-\theta_1 \cdot x \cdot log(\theta_2 \cdot x)$
$\theta_1 = 1.5$
$\theta_2 = 0.8$

Figure 1: **(A)** Illustration of D-CODE (compared with two-step symbolic regression). Both algorithms involve two steps, but they (1) estimate different variables in step one and (2) optimize different objective functions in step two. **(B)** An example of closed-form function and its tree representation.

The main difficulty is that the true label (the derivative) may not be accurately recoverable due to measurement and discretization error, but symbolic regression is sensitive to inaccurate labels because of the large search space (Agapitos et al., 2012). As we show experimentally in Section 5, even with the state-of-the-art methods for estimating derivatives, this approach still often fails when the noise is relatively low. We briefly review the methods for estimating derivatives from $\boldsymbol{y}(t)$ in Appendix E and show why this is generally a very challenging problem in Section 4.2.

The function approximator approach learns the true ODE with a function approximator $\widehat{f}$, such as a neural network (NN) (Chen et al., 2018; Rubanova et al., 2019) or a Gaussian process (GP) (Heinonen et al., 2018). It estimates the unknown initial condition $\widehat{\boldsymbol{x}}(0)$ and predicts the entire trajectory $\widehat{\boldsymbol{x}}$ by solving the initial value problem (IVP) of the approximated ODE $\widehat{f}$. The model is trained by minimizing the prediction error between predicted and the observed trajectories (Table 1).

Importantly, the function approximator approach does not give a concise closed-form expression to describe the dynamics, which is the main objective of this work. Furthermore, its performance strongly depends on the prediction horizon used for training. Using an inappropriate horizon may fail with certain type of systems, e.g. chaotic systems (Ditto & Munakata, 1995). We further discuss this issue and provide experimental results in Appendix H.2.

4 Method

4.1 The D-CODE Algorithm

D-CODE consists of a *preprocessing* step and an *optimization* step. Figure 1 provides a schematic illustration of D-CODE and Appendix B provides the pseudocode. The two steps are detailed below.

Preprocessing. D-CODE starts by estimating the trajectories $\widehat{\boldsymbol{x}}_i : [0,T] \to \mathbb{R}^J$, $i \le N$ from the noisy and discretely-sampled data $\mathcal{D}$ as an approximation to the true trajectories $\boldsymbol{x}_i$. Denoising and interpolating signals is a well-studied problem in statistics and signal processing, with many proven solutions including Gaussian process and spline regression (Bernardo et al., 1998; Marsh & Cormier, 2001). The D-CODE framework is agnostic to the exact choice of the smoothing algorithm. The user should choose a suitable algorithm based on the application and adopt best practices (e.g. Is the noise distribution Gaussian? Are the measurements made at regular intervals? etc.).

Optimization. After estimating the trajectories $\widehat{\boldsymbol{x}}_i$, we search for the function $\widehat{f}_j$, $\forall j \le J$ that is consistent with $\widehat{\boldsymbol{x}}_i$. Specifically, we solve the following optimization problems for all $j \le J$:

$$\widehat{f}_j = \arg\min_f \sum_{i=1}^{N} \sum_{s=1}^{S} C_j(f, \widehat{\boldsymbol{x}}_i, g_s)^2, \tag{5}$$

where C_j is the functional defined in Equation 3 and computed using numerical integration (Davis & Rabinowitz, 2007). The search space of f is the set of closed-form functions, and the user may instantiate D-CODE with *any* optimization algorithm proposed for symbolic regression (Section 3). We use a set of pre-defined testing functions $g_s \in \mathcal{C}^1[0, T]$, $s \leq S$ (to be discussed in Section 4.2).

Comparison with two-step symbolic regression. As depicted in Figure 1, D-CODE and two-step symbolic regression have two main differences. (1) The estimands in the pre-processing steps are different: D-CODE estimates the true trajectory $\boldsymbol{x}$ while the two-step symbolic regression estimates the time derivative. (2) The optimization objectives are different: D-CODE optimizes the loss function in Equation 5 while the two-step symbolic regression optimizes the prediction loss, e.g. RMSE.

4.2 Theoretical Results

In this section we provide a formal justification of the objective in Eq. 5 and describe the shortcomings of the objectives used in other methods (Table 1). For ease of exposition, we assume that our dataset contains only one trajectory, i.e. $N = 1$. The results can be easily extended to multiple trajectories.

Distance between f and f^*. A reasonable objective should measure the *distance* between a candidate function f and the target function f^* (i.e. the true ODE). By minimizing the distance, the candidate f would better approximate f^*. A common way to measure the distance between functions is to use the metrics induced by L^p norms of the function space. However, the L^p norms consider the values in the entire domain $\mathbb{R}^J$. This is a disadvantage because most dynamical systems in nature operate within specific range and scale, and $\mathbb{R}^J$ contains areas where we do not have knowledge about or cannot collect data from. A natural solution is to restrict the comparison on the trajectory $\boldsymbol{x}$, which can be achieved by composing f and f^* with $\boldsymbol{x}$. We propose the following distance function:

$$d_{\boldsymbol{x}}(f, f^*) := ||f \circ \boldsymbol{x} - f^* \circ \boldsymbol{x}||_2 = ||(f - f^*) \circ \boldsymbol{x}||_2 \tag{6}$$

where $\circ$ denotes function composition and $\boldsymbol{x}$ is a true trajectory satisfying the ODE f^*. In Appendix A, we further justify and discuss the properties of $d_{\boldsymbol{x}}(f, f^*)$. However, in practice, the $d_{\boldsymbol{x}}(f, f^*)$ cannot be computed from data because it depends on the unknown f^* and $\boldsymbol{x}$. Next, we show that the objective in Eq. 5, which *is* computable from data, can be used to approximate this distance.

Convergence to distance. The first step of our algorithm estimates a trajectory $\widehat{\boldsymbol{x}}$ from the measurements $\{\boldsymbol{y}(t) | t \in \mathcal{T}\}$ as an approximation to the true trajectory $\boldsymbol{x}$. For a suitable smoothing algorithm, if we increase the number of samples on the trajectory, the recovered trajectory should converge to the ground truth. This is achieved, for instance, by spline regression (Stone, 1985; 1994) or Gaussian processes Choi & Schervish (2007). If this is the case, Theorem 1 shows that given high enough sampling frequency and large enough number of testing functions our objective converges to the squared distance $d_{\boldsymbol{x}}(f, f^*)^2$, which measures how different f is from f^*. That justifies our objective.

Theorem 1. *Consider $J \in \mathbb{N}^+$, $j \in \{1, \dots, J\}$, $T \in \mathbb{R}^+$. Let $f^* : \mathbb{R}^J \to \mathbb{R}$ be a continuous function, and let $\boldsymbol{x} : [0, T] \to \mathbb{R}^J$ be a continuously differentiable function satisfying $\dot{x}_j(t) = f^*(\boldsymbol{x}(t))$. Consider a sequence of functions $(\widehat{\boldsymbol{x}}_k)$, where $\widehat{\boldsymbol{x}}_k : [0, T] \to \mathbb{R}^J$ is a continuously differentiable function. If $(\widehat{\boldsymbol{x}}_k)$ converges to $\boldsymbol{x}$ in L^2 norm. Then for any Lipschitz continuous function f*

$$\lim_{S \to \infty} \lim_{k \to \infty} \sum_{s=1}^{S} C_j(f, \widehat{\boldsymbol{x}}_k, g_s)^2 = d_{\boldsymbol{x}}(f, f^*)^2, \tag{7}$$

where $\{g_1, g_2, \dots\}$ is a Hilbert (orthonormal) basis for $L^2[0, T]$ such that $\forall i$, $g_i(0) = g_i(T) = 0$ and $g_i \in \mathcal{C}^1[0, T]$.

We provide the proof of Theorem 1 in the Appendix A.

Comparison with other methods. Theorem 1 does not place any additional constraints regarding the smoothing algorithm apart from the convergence of the regression function. In general, the convergence of functions does not imply the convergence of the derivatives (example in Appendix A). That means that in contrast to D-CODE, other objectives in Table 1 place constraints on the type estimation algorithm, i.e., the estimated derivative $\hat{\dot{x}}$ needs to converge to the true derivative $\dot{x}$. Even if the estimation algorithm satisfies this constraint (for instance, spline regression (Zhou & Wolfe, 2000)), the convergence rate for derivatives is slower than for the functions themselves (Stone, 1982). Convergence rate depends on the smoothness (differentiability class) of the function and differentiation reduces its differentiability class. Namely, if a true trajectory has a maximum number of m continuous derivatives then its derivative $\dot{x}$ has only $m-1$ continuous derivatives.

Choice of testing functions. As stated in Theorem 1, we should use testing functions that are orthonormal basis for $L^2[0,T]$ and satisfy $g(0) = g(T) = 0$. Ideally, they should also have analytical time derivatives $\dot{g}(t)$ for efficient computation of the functional C (Eq. 3). Hence, a natural choice is the sine functions $g_s(t) = \sqrt{2/T} \cdot \sin(s\pi t/T)$, which we will use in all the main experiments. Another possibility is the cubic spline functions (investigated in Appendix G). Although Theorem 1 uses infinitely many testing functions to establish convergence, we observe in the experiments that typically 40-60 sine functions are sufficient (a sensitivity analysis is conducted in Appendix G).

5 Experiments and Evaluation

In this section, we perform a series of simulations to evaluate whether the algorithms can discover the underlying closed-form ODEs that govern the observed trajectories.[2]

Choice of dynamical systems. In this study, we select five dynamical systems governed by nine closed-form ODEs (one of them is discussed and examined in Appendix G due to space limit). The selected systems have varying complexity and different properties of the temporal dynamics. We start with two common growth models, the Gompertz model and the generalized logistic model (Gompertz, 1825; Richards, 1959). Both models involve one variable governed by a nonlinear ODE and converging to a global fixed point. Next, we consider the glycolytic oscillator in biochemistry (Sel'Kov, 1968), which is a standard benchmark problem for dynamical system prediction and inference (Daniels & Nemenman, 2015a;b). The system involves two nonlinearly interacting variables that converge to a oscillatory limit cycle. Finally, we consider the chaotic Lorenz system, which involves three variables forming a strange attractor (Lorenz, 1963). Together, these systems represent a range of temporal dynamics and application scenarios.

Measurement settings. For each dynamical system, we consider different measurement settings specified by (1) the measurement noise level σ_R, (2) the sampling step size Δt, and (3) the number of trajectories N. Since the dynamical systems have different scales, we will report the noise-to-signal ratio $\sigma_R = \sigma/\text{std}(\boldsymbol{x}(t))$ for ease of comparison. We sample the system at regular intervals $\mathcal{T} = \{\Delta t,\ 2\Delta t,\ \dots\ T\}$ to show the effect of changing sampling frequency. For each setting, we perform 100 independent simulation runs to compute the evaluation metric and its confidence interval.

Data generation. For each trajectory, we first sample the initial condition $\boldsymbol{x}_i(0)$ from a uniform distribution (specified in Appendix F). Then we obtain the true $\boldsymbol{x}_i$ by solving the IVP computationally. Finally, we obtain $\boldsymbol{y}_i(t)$ by adding independent Gaussian noise and sampling at discrete time steps.

Evaluation metrics. We use three metrics to study various aspects of the algorithms. (1) The probability of successfully recovering the functional form (**Success Prob.**). We use a computer algebra system to judge whether the true and discovered functional forms are equivalent (Meurer et al., 2017) (e.g. $\theta_1 + x_1 \equiv x_1 + \theta_1$). (2) When the functional form is correct, we evaluate the accuracy of the estimated constants $\hat{\theta}$ using root mean square error (**RMSE**). (3) We evaluate how well the estimated $\widehat{f}$ approximates f^* using the distance function in Equation 6 (**Dist.**). Note that it is possible that $\widehat{f}$ has a wrong functional form but still approximates f^* well.

Algorithm and Benchmarks. We instantiate D-CODE with Gaussian process for smoothing and genetic programming for optimization due to their success and popularity in the literature (Gramacy, 2020; Schmidt & Lipson, 2009). Our main benchmark is the two-step symbolic regression method. We consider three variants with different methods for estimating the derivative: Total variation regularized differentiation (**SR-T**) (Chartrand, 2011), Spline-smoothed differentiation (**SR-S**) (Ahnert & Abel, 2007), and Gaussian process smoothed differentiation (**SR-G**) (detailed in Appendix E). Note that SR-G and D-CODE both fit Gaussian process in the pre-processing step but they use different objective functions. For fairness of comparison, we use genetic programming for optimization in all these benchmarks. We also compare with Neural ODEs in some cases, although it does not give closed-form expressions. Appendix F contains further implementation details.

[2]The code is available at `https://github.com/ZhaozhiQIAN/D-CODE-ICLR-2022`.

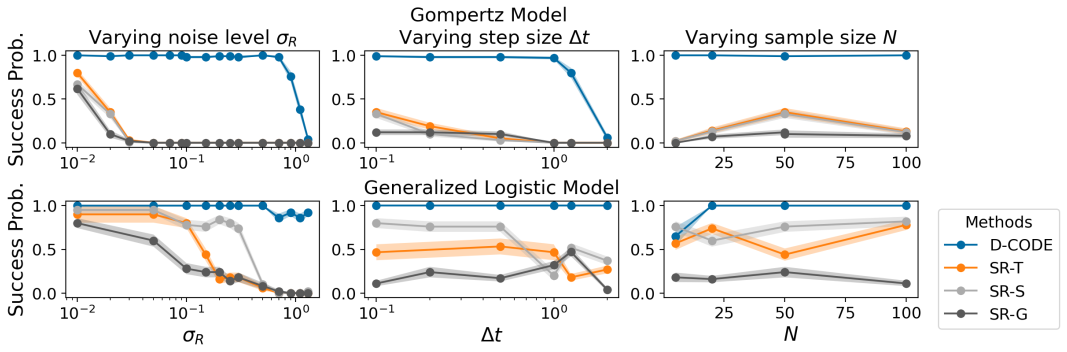


Figure 2: The success probability of the two growth models under different settings. The three columns correspond to various noise levels σ_R, step sizes Δt, and numbers of trajectories N. The two rows correspond to the Gompertz model and the generalized logistic model. The shaded area is the 95% confidence interval.

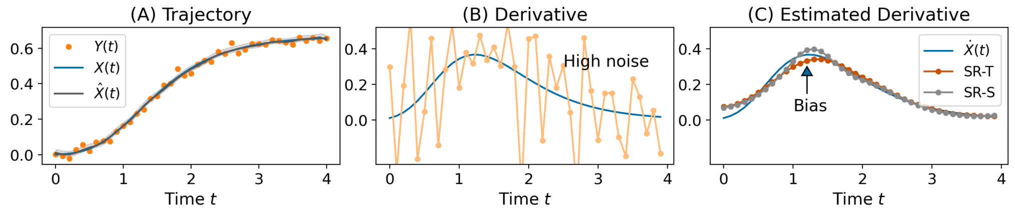


Figure 3: Illustration of performance gain using a trajectory with moderate noise $\sigma_R = 0.1$. **Panel A**: the trajectory $\widehat{\boldsymbol{x}}(t)$ estimated from the measurements $\boldsymbol{y}(t)$ is very close to the true trajectory $\boldsymbol{x}(t)$; hence, the first step of D-CODE tends to have small error. **Panels B and C**: even with moderate noise on $\boldsymbol{x}(t)$, the estimated derivatives $\dot{x}(t)$ may suffer from very high variance (B: finite difference) or systematic bias (C: SR-T, SR-S). Hence, two-step symbolic regression tends to under-perform due to high error in the first step.

5.1 RESULTS

Gompertz and generalized logistic growth models. These two models are widely used to capture asymmetric growth with saturation. Their governing equations are as follows:

$$\dot{x}(t) = -\theta_1 x(t) \cdot \log\big(\theta_2 x(t)\big) \qquad \text{Gompertz Model}$$

$$\dot{x}(t) = \theta_1 x(t) \cdot \big(1 - x(t)^{\theta_2}\big) \qquad \text{Generalized Logistic Model}$$

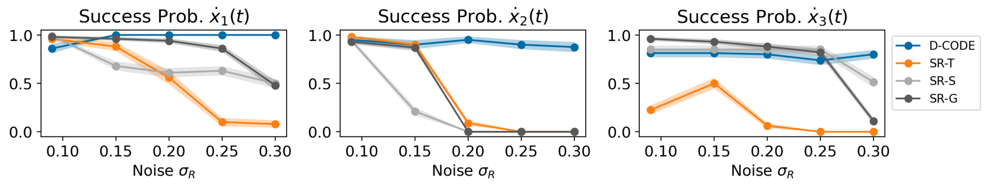
where $\theta_1, \theta_2 > 0$ are numeric constants. It is worth highlighting that the generalized logistic model involves a constant $\theta_2 \in \mathbb{R}^+$ in the power of $x(t)$. Therefore, it *cannot* be symbolically expressed in a sparse linear form $\sum_{k=1}^{K} \theta_k h_k(x(t))$ for some candidate functions h_k, and the two-step sparse regression method cannot discover the exact functional form of this equation (Section 3.2).

The main simulation results are presented in Figure 2 (the RMSE and distance metrics follow the same pattern and are shown in Appendix G. The discovered ODEs are shown in Appendix G Table 4). D-CODE achieves comparable performance as the benchmarks in the low noise, small step size settings ($\sigma_R = 0.01$), where the time derivative $\dot{x}$ can be accurately estimated from the observed trajectories. However, D-CODE consistently out-performs the benchmarks under the more challenging measurement settings. With increased noise or step size, the success probabilities decrease much more rapidly in the benchmarks compared to D-CODE. Note that although SR-G and D-CODE both use Gaussian process in the pre-processing step, D-CODE achieves much stronger performance. This suggests that D-CODE's performance gain is mainly due to its objective function.

Figure 3 contextualizes the reason for performance gain with an illustrative example. The panel A shows a typical observed trajectory with moderate measurement noise ($\sigma_R = 0.1$). Here, the Gaussian process can produce a good estimate $\widehat{\boldsymbol{x}}(t)$ by de-noising the measurements. Its posterior interval covers the true trajectory $\boldsymbol{x}(t)$. Hence, $\widehat{\boldsymbol{x}}(t)$ is suitable to guide the search for the ODE, as is done in D-CODE. On the other hand, the moderate measurement noise on $\boldsymbol{x}(t)$ translates into very high noise on the time derivative $\dot{x}(t)$ (panel B). Even with the state-of-the-art numerical differentiation methods used by SR-T and SR-S, the estimated derivative is still biased on certain

Table 2: Simulation results of the glycolytic oscillator under different noise levels. The success probability and the RMSE on θ_1, θ_3 are reported for the two equations. Standard deviations are shown in the brackets.

Equation	Method	Success Prob.			RMSE $\hat{\theta}$ (10^{-2})		
		$\sigma_R = 0.01$	0.1	0.2	$\sigma_R = 0.01$	0.1	0.2
Eq. 8	SR-T	0.45 (.05)	0.27 (.05)	0.00 (.00)	1.19 (.15)	2.37 (.38)	NA
	SR-S	0.44 (.06)	0.11 (.04)	0.00 (.00)	1.67 (.24)	2.07 (.47)	NA
	SR-G	0.44 (.05)	0.05 (.02)	0.00 (.00)	1.87 (.22)	2.18 (.46)	NA
	D-CODE	**0.58 (.05)**	**0.51 (.05)**	**0.26 (.05)**	**1.01 (.12)**	**1.55 (.28)**	**2.00 (.27)**
Eq. 9	SR-T	0.99 (.03)	0.88 (.03)	0.00 (.00)	0.14 (.03)	0.43 (.03)	NA
	SR-S	0.95 (.02)	0.04 (.02)	0.00 (.00)	0.10 (.01)	0.41 (.18)	NA
	SR-G	0.99 (.01)	**0.94 (.02)**	0.25 (.04)	0.22 (.02)	1.10 (.07)	2.85 (.02)
	D-CODE	**1.00 (.00)**	**0.91 (.03)**	**0.65 (.05)**	**0.07 (.01)**	**0.40 (.07)**	**0.61 (.05)**



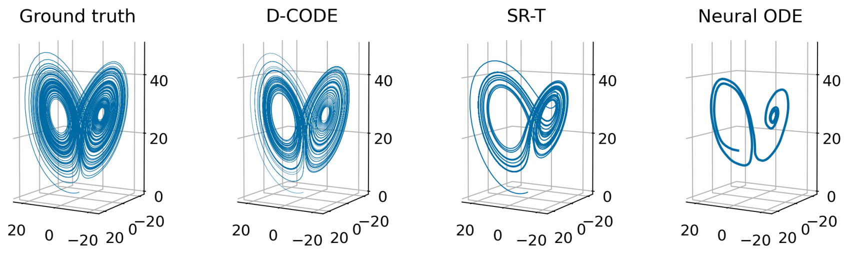


Figure 4: Simulation results of the chaotic Lorenz system. **First row**: the success probabilities for the three equations under different noise levels. **Second row**: simulated trajectories using true and estimated equations.

intervals (panel C). Thus, applying symbolic regression on the biased estimates is unlikely to perform well, as is shown in the benchmark results. The same reasoning also applies to the scenario with large sampling step size Δt, another challenging situation for estimating the time derivatives.

Glycolytic oscillator. This biochemical oscillator consists of two ODEs (Sel'Kov, 1968):

$$\dot{x}_1(t) = \theta_1 - \theta_2 x_1(t) - x_1(t)x_2(t)^2 \tag{8}$$

$$\dot{x}_2(t) = -x_2(t) + \theta_3 x_1(t) + x_1(t)x_2(t)^2 \tag{9}$$

where $\theta_1, \theta_2, \theta_3 > 0$ are numeric constants. Compared with the growth models above, the glycolytic oscillator involves more variables and has more complex expressions. The two variables in the equation interact in a highly nonlinear way through the term $x_1(t)x_2(t)^2$.

The simulation results for the glycolytic oscillator are shown in Table 2. Here we report the success probability in addition to the RMSE ($\times 10^{-2}$) on the constants θ_2, θ_3, whose true values are set to 0.1 (other metrics are reported in Appendix G). D-CODE performs better or equally well compared to the benchmarks in all metrics. Note that D-CODE is the only model that successfully recovered the functional form under $\sigma_R = 0.2$ (The benchmarks' RMSE are NA in this setting because they never recovered the correct functional form).

Chaotic Lorenz system. The Lorenz system is a model system for chaotic dynamics, defined as:

$$\dot{x}_1(t) = \theta_1\big(x_2(t) - x_1(t)\big); \quad \dot{x}_2(t) = x_1(t)\big(\theta_2 - x_3(t)\big) - x_2(t); \quad \dot{x}_3(t) = x_1(t)x_2(t) - \theta_3 x_3(t)$$

We set the constants as $\theta_1 = 10, \theta_2 = 28, \theta_3 = 8/3$, which leads to chaotic behaviour.

The simulation results for the chaotic Lorenz system are shown in Figure 4. D-CODE's robustness to measurement noise is still evident when the underlying system is chaotic and non-periodic. The benchmarks achieve high success probability only when the measurement noise is low. The second row of Figure 4 shows a simulated trajectory in the phase space using the true or estimated ODEs (with $\sigma_R = 0.2$). D-CODE generates a trajectory that is closest to the true one. SR-T fails because it is unable to recover the true ODE under this noise level. Although the Neural ODE is trained on a very short prediction horizon (0.04), it still fails to capture the non-periodic behavior of the system.

6 D-CODE IN ACTION

We apply D-CODE to model the temporal effect of chemotherapy on the tumor volume. For most patients, the tumor volume decreases initially in response to the treatment but it eventually grows again due to drug resistance. Although there exist many models for unintervened tumor growth, few models can capture the effect of chemotherapy or the possibility of relapsing—in many cases exponential decay (log cell-kill model) is used for a crude approximation (Geng et al., 2017).

We used the dataset collected by Wilkerson et al. (2017) based on eight clinical trials on cancer patients. The patients started chemotherapy at time $t = 0$, and the trajectory of tumor volume are measured for up to one year $T = 1$. In practice, the tumor volumes are inferred from medical scans and are subject to significant measurement noise (Mazaheri et al., 2009). After data cleaning, we obtained 310 trajectories, among which 200 is used for training and 110 for evaluation.

The following two ODEs are discovered by D-CODE and SR-T.

$$\dot{x}(t) = 4.48t^2x(t) + \log(t); \quad \text{D-CODE}$$
$$\dot{x}(t) = 4x(t)\log\big(tx(t)\big)\log\big(tx(t) + 2t\big); \quad \text{SR-T}$$

The equation discovered by SR-T is much more complex than the one found by D-CODE while still being a poor representation of the data. Figure 5 shows an observed trajectory and the trajectories generated by the two ODEs. D-CODE successfully captures the relapse at $t = 0.6$ and predicts that, under no further intervention, the tumor volume would keep growing after $t > 0.6$ until patient death. In comparison, SR-T's trajectory is almost constant.

We also quantitatively evaluate the prediction errors on the testing set. Given the tumor size measured at $t = 0$, we solve the IVP to obtain the full trajectory $\boldsymbol{x}$ and compute the RMSE between the estimated and the observed trajectory. D-CODE achieves an error of 0.220 compared with 0.637 of SR-T.

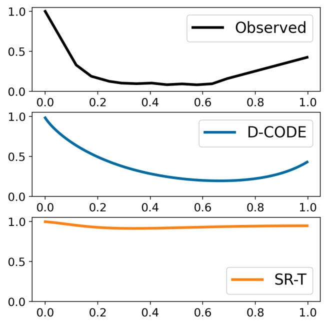


Figure 5: An observed trajectory and the trajectories generated by the two discovered ODEs. Values are indexed at $t = 0$.

7 DISCUSSION ON FAILURE MODES AND OPEN CHALLENGES

Scientific discovery is a challenging process with no success guarantee. The same holds for ODE discovery with D-CODE. Here we discuss the failure modes and the challenges for future work.

Unobserved variables. In some cases, certain external variables $\boldsymbol{u}(t) \in \mathbb{R}^L$ may also influence the temporal dynamics, i.e. $\dot{x}_j(t) = f\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big)$. When $\boldsymbol{u}(t)$ is observed or known, D-CODE can include these variables to discover the expanded equation. However, when they are *unobserved*, the identification of system dynamics is challenging or even impossible without strong assumptions (Ljung, 1998). In this case, D-CODE can still find an ODE based on the observed variables—this ODE may still be a useful approximation but it will leave some dynamics unexplained.

Complex equations. The discovery algorithm may fail to discover highly complex ODEs. The complexity can arise from two aspects: (1) $\boldsymbol{x}(t)$ may include many variables (high dimensionality) or (2) the equation f^* may include many mathematical operations (long expression). However, we highlight that these settings are difficult in general, even for human experts with domain knowledge.

Extreme measurement settings. D-CODE may still fail when the measurement noise or step size is too large. However, as we show experimentally, D-CODE is more robust to measurement artifacts than the alternative discovery methods.

Acknowledgments

This work is supported by the US Office of Naval Research (ONR), the National Science Foundation (NSF Grant number:1722516), and Roche. We thank the anonymous reviewers, Professor William Zame and the members of vanderschaar-lab for many insightful comments and suggestions.

Ethics Statement. In this work, we use a tumor growth dataset that is anonymized and licensed for research purpose (Section 6). The dataset has also been used in prior research in the literature (Wilkerson et al., 2017). We discuss the limitations and open challenges for D-CODE in Section 7. We stress that D-CODE is not intended to or capable of fully replacing human experts in modeling dynamical systems. We envision that D-CODE would facilitate the modeling process by automatically extracting closed-form ODEs from observed trajectories — allowing human experts to easily inspect, analyze and refine the distilled ODEs.

Reproducibility Statement. The proof of theoretical results is shown in Appendix A. The technical assumptions for the proof are listed in Proposition 1 and Theorem 1 in the main text and further discussed in Appendix A. The experiment settings are discussed in Section 5 and 6 and further detailed in Appendix F. The experiment code is available at `https://github.com/ZhaozhiQIAN/D-CODE-ICLR-2022`.

References

Alexandros Agapitos, Anthony Brabazon, and Michael O'Neill. Controlling overfitting in symbolic regression based on a bias/variance error decomposition. In *International Conference on Parallel Problem Solving from Nature*, pp. 438–447. Springer, 2012.

Karsten Ahnert and Markus Abel. Numerical differentiation of experimental data: local versus global methods. *Computer Physics Communications*, 177(10):764–774, 2007.

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.

J Bernardo, J Berger, APAFMS Dawid, A Smith, et al. Regression and classification using gaussian process priors. *Bayesian statistics*, 6:475, 1998.

WG Bickley. Formulae for numerical differentiation. *The Mathematical Gazette*, 25(263):19–27, 1941.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. *arXiv preprint arXiv:2106.06427*, 2021.

Bernd Blasius, Lars Rudolf, Guntram Weithoff, Ursula Gaedke, and Gregor F Fussmann. Long-term cyclic persistence in an experimental predator–prey system. *Nature*, 577(7789):226–230, 2020.

Jonathan M Borwein, Richard E Crandall, et al. Closed forms: what they are and why we care. *Notices of the AMS*, 60(1):50–65, 2013.

Felix E Browder. Fixed point theory and nonlinear problems. *Proc. Sym. Pure. Math*, 39:49–88, 1983.

Nicolas Brunel and Peter E Latham. Firing rate of the noisy quadratic integrate-and-fire neuron. *Neural computation*, 15(10):2281–2306, 2003.

Nicolas JB Brunel, Quentin Clairon, and Florence d'Alché Buc. Parametric estimation of ordinary differential equations with orthogonality conditions. *Journal of the American Statistical Association*, 109(505):173–185, 2014.

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, 2016.

Rick Chartrand. Numerical differentiation of noisy, nonsmooth data. *International Scholarly Research Notices*, 2011, 2011.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

Taeryon Choi and Mark J. Schervish. On posterior consistency in nonparametric regression problems. *Journal of Multivariate Analysis*, 98(10):1969–1987, November 2007.

S-N Chow and Jack K Hale. *Methods of bifurcation theory*, volume 251. Springer Science & Business Media, 2012.

Jane Cullum. Numerical differentiation and regularization. *SIAM Journal on numerical analysis*, 8 (2):254–265, 1971.

Bryan C Daniels and Ilya Nemenman. Automated adaptive inference of phenomenological dynamical models. *Nature communications*, 6(1):1–8, 2015a.

Bryan C Daniels and Ilya Nemenman. Efficient inference of parsimonious phenomenological models of cellular dynamics using s-systems and alternating regression. *PloS one*, 10(3):e0119821, 2015b.

Philip J Davis and Philip Rabinowitz. *Methods of numerical integration*. Courier Corporation, 2007.

Carl De Boor. *A practical guide to splines*, volume 27. springer-verlag New York, 1978.

William Ditto and Toshinori Munakata. Principles and applications of chaotic systems. *Communications of the ACM*, 38(11):96–102, 1995.

Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

L. E. Elsgolc. *Calculus of variations / L. E. Elsgolc*. International series of monographs in pure and applied mathematics ; Volume 19. Oxford, New York, Pergamon Press 1961, 1961. ISBN 1-4831-3756-2.

William F. Ford and James A. Pennline. When does convergence in the mean imply uniform convergence? *The American Mathematical Monthly*, 114(1):58–60, 2007.

Stephanie Forrest. Genetic algorithms: principles of natural selection applied to computation. *Science*, 261(5123):872–878, 1993.

Sébastien Gaucel, Maarten Keijzer, Evelyne Lutton, and Alberto Tonda. Learning dynamical systems using standard symbolic regression. In *European Conference on Genetic Programming*, pp. 25–36. Springer, 2014.

Changran Geng, Harald Paganetti, and Clemens Grassberger. Prediction of treatment response for combined chemo-and radiation therapy for non-small cell lung cancer patients using a bio-mathematical model. *Scientific reports*, 7(1):1–12, 2017.

Nitin S Gokhale. *Practical finite element analysis*. Finite to infinite, 2008.

Benjamin Gompertz. Xxiv. on the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies. in a letter to francis baily, esq. frs &c. *Philosophical transactions of the Royal Society of London*, pp. 513–583, 1825.

Robert B Gramacy. *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. Chapman and Hall/CRC, 2020.

Shota Gugushvili and Chris AJ Klaassen. $\sqrt{n}$-consistent parameter estimation for systems of ordinary differential equations: bypassing numerical integration via smoothing. *Bernoulli*, 18(3):1061–1098, 2012.

Wolfgang Hackbusch. *Variational Formulation*, pp. 159–180. Springer Berlin Heidelberg, Berlin, Heidelberg, 2017. ISBN 978-3-662-54961-2. doi: 10.1007/978-3-662-54961-2_7.

Wassim M Haddad and VijaySekhar Chellaboina. *Nonlinear dynamical systems and control*. Princeton university press, 2011.

Markus Heinonen, Cagatay Yildiz, Henrik Mannerström, Jukka Intosalmi, and Harri Lähdesmäki. Learning unknown ode models with gaussian processes. In *International Conference on Machine Learning*, pp. 1959–1968. PMLR, 2018.

Anders Boeck Jensen, Pope L Moseley, Tudor I Oprea, Sabrina Gade Ellesøe, Robert Eriksson, Henriette Schmock, Peter Bjødstrup Jensen, Lars Juhl Jensen, and Søren Brunak. Temporal disease trajectories condensed from population-wide registry data covering 6.2 million patients. *Nature communications*, 5(1):1–10, 2014.

Holger Kantz and Thomas Schreiber. *Nonlinear time series analysis*, volume 7. Cambridge university press, 2004.

John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994.

Lennart Ljung. System identification. In *Signal analysis and prediction*, pp. 163–173. Springer, 1998.

Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

Qiang Lu, Jun Ren, and Zhiguang Wang. Using genetic programming with prior formula knowledge to solve symbolic regression problem. *Computational intelligence and neuroscience*, 2016, 2016.

Aleksandr Mikhailovich Lyapunov. The general problem of the stability of motion. *International journal of control*, 55(3):531–534, 1992.

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, 2016.

Lawrence C Marsh and David R Cormier. *Spline regression models*. Sage, 2001.

Yousef Mazaheri, Hedvig Hricak, Samson W Fine, Oguz Akin, Amita Shukla-Dave, Nicole M Ishill, Chaya S Moskowitz, Joanna E Grater, Victor E Reuter, Kristen L Zakian, et al. Prostate tumor volume measurement with combined t2-weighted imaging and diffusion-weighted mr: correlation with pathologic tumor volume. *Radiology*, 252(2):449–457, 2009.

Volker Ludwig Mehrmann. *The autonomous linear quadratic control problem: theory and numerical solution*, volume 163. Springer, 1991.

Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

Brenden K Petersen, Mikel Landajuela Larma, 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. *arXiv preprint arXiv:1912.04871*, 2019.

Zhaozhi Qian, Ahmed Alaa, Alexis Bellot, Mihaela Schaar, and Jem Rashbass. Learning dynamic and personalized comorbidity networks from event data using deep diffusion processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 3295–3305. PMLR, 2020.

Markus Quade and Andy Goldschmidt. Numerical differentiation of noisy time series data in python. `https://pypi.org/project/derivative/`, 2020.

Jim O Ramsay, Giles Hooker, David Campbell, and Jiguo Cao. Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(5):741–796, 2007.

FJ Richards. A flexible growth function for empirical use. *Journal of experimental Botany*, 10(2): 290–301, 1959.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in Neural Information Processing Systems*, 32: 5320–5330, 2019.

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.

Dominik Schmidt, Georgia Koppe, Zahra Monfared, Max Beutelspacher, and Daniel Durstewitz. Identifying nonlinear dynamical systems with multiple time scales and long-range dependencies. In *International Conference on Learning Representations*, 2020.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

Peter Schuster. What is special about autocatalysis? *Monatshefte für Chemie-Chemical Monthly*, 150 (5):763–775, 2019.

EE Sel'Kov. Self-oscillations in glycolysis 1. a simple kinetic model. *European Journal of Biochemistry*, 4(1):79–86, 1968.

Xiaotong Shen, DA Wolfe, and S Zhou. Local asymptotics for regression splines and confidence regions. *The annals of statistics*, 26(5):1760–1782, 1998.

George Finlay Simmons. *Differential equations, with applications and historical notes*. New York, McGraw-Hill, 1972.

T Stephens. gplearn: Genetic programming in python, with a scikit-learn inspired and compatible api. https://gplearn.readthedocs.io/en/stable/index.html, 2021.

Charles J Stone. Optimal global rates of convergence for nonparametric regression. *The annals of statistics*, pp. 1040–1053, 1982. Publisher: JSTOR.

Charles J. Stone. Additive regression and other nonparametric models. *The Annals of Statistics*, 13 (2):689–705, 1985.

Charles J. Stone. The use of polynomial splines and their tensor products in multivariate function estimation. *The Annals of Statistics*, 22(1):118–171, 1994.

David Strong and Tony Chan. Edge-preserving and scale-dependent properties of total variation regularization. *Inverse problems*, 19(6):S165, 2003.

Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

Greg Welch, Gary Bishop, et al. *An introduction to the Kalman filter*. Chapel Hill, NC, USA, 1995.

Julia Wilkerson, Kald Abdallah, Charles Hugh-Jones, Greg Curt, Mace Rothenberg, Ronit Simantov, Martin Murphy, Joseph Morrell, Joel Beetsch, Daniel J Sargent, et al. Estimation of tumour regression and growth rates during treatment in patients with advanced prostate cancer: a retrospective analysis. *The Lancet Oncology*, 18(1):143–154, 2017.

Jan Žegklitz and Petr Pošík. Benchmarking state-of-the-art symbolic regression algorithms. *Genetic programming and evolvable machines*, 22(1):5–33, 2021.

Shanggang Zhou and Douglas A Wolfe. On derivative estimation in spline regression. *Statistica Sinica*, pp. 93–108, 2000.

Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, Perumal Nithiarasu, and JZ Zhu. *The finite element method*, volume 3. McGraw-hill London, 1977.

Table of Supplementary Materials

A Theoretical results: proof and discussion

A.1 Variational formulation of ODE

Before we prove Proposition 1, we need the following lemma which is a particular formulation of the Fundamental lemma of calculus of variations Elsgolc (1961).

Lemma (Fundamental lemma of calculus of variations). *Let h be a continuous function on a closed interval $[0, T]$. h is equal to 0 everywhere if and only if $\int_0^T h(t)g(t)\,dt = 0$ for all $g \in \mathcal{C}^1[0, T]$ such that $g(0) = g(T) = 0$.*

Proof. The forward direction is trivial, so let us focus on the converse.

Assume for contradiction that there is a point $t_0 \in [0, T]$ such that $h(t_0) \neq 0$. Without loss of generality, assume that $h(t_0) > 0$. From the continuity of h, we know that there is actually a point $t_1 \in (0, T)$, such that $h(t_1) > 0$. Continuity of h implies that there exists $\delta > 0$ and a small neighbourhood $(t_1 - \delta, t_1 + \delta) \subset (0, T)$ such that $h(t) > 0\ \forall t \in (t_1 - \delta, t_1 + \delta)$.

We can now define a non-negative continuous function $g : [0, T] \to [0, +\infty)$ such that $g(t) = 0$ outside the neighbourhood $(t_1 - \delta, t_1 + \delta)$ and $g(t) > 0$ for some interval inside $(t_1 - \delta, t_1 + \delta)$. See Figure 6 for an illustrative example.

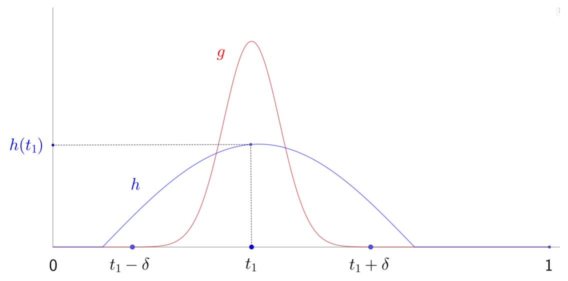


Figure 6: A visual illustration of the proof of Fundamental lemma of calculus of variations

Then

$$\int_0^T h(t)g(t)\,dt = \int_{t_1-\delta}^{t_1+\delta} h(t)g(t)dt > 0 \tag{10}$$

That contradicts $\int_0^T h(t)g(t)\,dt = 0$ for all $g \in \mathcal{C}^1[0, T]$ such that $g(0) = g(T) = 0$. $\square$

Proof of Proposition 1. Observe that

$$\dot{x}_j(t) = f_j(\boldsymbol{x}(t)) \iff f_j(\boldsymbol{x}(t)) - \dot{x}_j(t) = 0 \tag{11}$$

Now using Fundamental lemma of calculus of variations, we get that the following two statements are equivalent

$$\begin{gathered} f_j(\boldsymbol{x}(t)) - \dot{x}_j(t) = 0 \; \forall t \in [0, T] \\ \int_0^T (f_j(\boldsymbol{x}(t)) - \dot{x}_j(t)) g(t) dt = 0 \; \forall g \in \mathcal{C}^1[0, T], \; g(0) = g(T) = 0 \end{gathered} \tag{12}$$

By linearity and integration by parts, we get

$$\begin{aligned} \int_0^T (f_j(\boldsymbol{x}(t)) - \dot{x}_j(t)) g(t) dt &= \int_0^T f_j(\boldsymbol{x}(t)) g(t) dt - \int_0^T \dot{x}_j(t) g(t) dt \\ &= \int_0^T f_j(\boldsymbol{x}(t)) g(t) dt + \int_0^T x_j(t) \dot{g}(t) dt - x_j(T) g(T) + x_j(0) g(0) \\ &= \int_0^T f_j(\boldsymbol{x}(t)) g(t) dt + \int_0^T x_j(t) \dot{g}(t) dt \end{aligned} \tag{13}$$

where in the last equality we used the fact that $g(0) = g(T) = 0$. That means that

$$\int_0^T (f_j(\boldsymbol{x}(t)) - \dot{x}_j(t)) g(t) dt = C_j(f_j, \boldsymbol{x}, g) \tag{14}$$

This proves that

$$\dot{x}_j(t) = f_j(\boldsymbol{x}(t)) \; \forall j \in \{1, \ldots, J\} \tag{15}$$

if and only if

$$C_j(f_j, \boldsymbol{x}, g) = 0, \; \forall j \in \{1, \ldots, J\}, \; \forall g \in \mathcal{C}^1[0, T], \; g(0) = g(T) = 0 \tag{16}$$

□

Discussion. As we show in the proof above, the key idea of the variational formulation is to bypass the derivative of the trajectory $\dot{x}_j(t)$ with the derivative of a testing function $\dot{g}(t)$ — the swapping of derivatives is achieved by integration by parts (Equations 13). An important consequence is that we no longer need to know or estimate $\dot{x}_j(t)$ (which is unobserved and challenging to estimate), but only need to know $\dot{g}(t)$ for a given testing function $g(t)$. Indeed, if $g(t)$ has an analytical derivative, we immediately have access to $\dot{g}(t)$.

Note that the variational formulation builds on the *exact* condition $C_j = 0$. It does not consider the situation where $C_j \neq 0$, no matter how close C_j is to 0. Hence, the formulation itself is inadequate to guide the search of closed-form ODEs. For instance, suppose the C_j for two candidate functions $\hat{f}_1$ and $\hat{f}_2$ are 0.01 and 1000 respectively. The variational formulation does not imply $\hat{f}_1$ fits the trajectory $\boldsymbol{x}$ better than $\hat{f}_2$; rather, it implies that $\hat{f}_1$ and $\hat{f}_2$ are *equally* bad as $C_j \neq 0$ for both. In order to bridge this gap, we need to establish a notion of distance between functions and link the functional C_j to this distance — they are addressed in Appendix A.2 and A.3 respectively.

The formulation above is tailored for ODEs, which is the focus of the current work. However, the variational formulation can be generalized to other types of differential equations, e.g partial differential equations (PDE) and delay differential equations (DDE). In fact, the variational formulation is the theoretical foundation of the Finite Element Method (FEM) — the industrial standard for solving PDEs in complex engineering systems and fluid dynamics (Zienkiewicz et al., 1977; Gokhale, 2008). It has also been used to estimate the parameters of a *known* ODE or DDE (Brunel et al., 2014). We envision that future works may be built on the general variational formulation to discover closed-form PDE or DDE from data.

A.2 Distance function

In this section, we justify that $d_{\boldsymbol{x}}(f, f^*)$ is an appropriate metric for our problem.

There are two main ways of comparing functions. We can either compare their symbolic representations or measure the differences in values they take for a set of arguments. As we want this metric to be a limit of some objective computable from the data, we focus on the latter.

Commonly used distance functions are metrics induced by L^p norms. L^p norm is defined in the following way:

$$||g||_p = \left(\int_a^b |g(t)|^p dt \right)^{\frac{1}{p}} \tag{17}$$

In practice, L^1 or L^2 norm is used. The induced metric is then defined as the norm of the difference between the functions, namely $d(f, g) = ||f - g||_p$.

Now the question remains which difference we should take the norm of. We have three reasonable choices:

1. $f - f^*$
2. $\tilde{\boldsymbol{x}} - \boldsymbol{x}$, where $\tilde{\boldsymbol{x}}$ is trajectory prescribed by f computed from an estimated initial condition
3. $f - f^*$ composed with some other function

The first choice poses the challenge of choosing the range $[a, b]$ over which we integrate $f - f^*$. In general we do not make any assumptions on the domain of the functions. It also seems unreasonable to compare the functions far from trajectory, where we have no information about their behaviour.

The second choice is similar to what is done in the Neural ODE approach. As we mentioned before, this involves solving the Initial Value Problem (IVP). Even if the IVP has a unique solution, it might be very sensitive to the initial condition, making it unfit for chaotic systems. We verify this claim in the experiments (Figure 4)

The third proposition can solve the main issue of the first option. If we compose $f - f^*$ with the ground truth trajectory, $\boldsymbol{x}$, not only we have a well-defined interval over which to integrate $([0, T])$ but also we focus our attention on values of f^* where we have a chance to recover it, and where it matters for us the most.

That is why we decided to chose $d_{\boldsymbol{x}}(f, f^*) = ||(f - f^*) \circ \boldsymbol{x}||_2$ as our distance function. Moreover, as it is based on a familiar metric, it inherits many important properties, such as

1. $d_{\boldsymbol{x}}$ is a pseudo-metric[3], in particular, $f = f^* \implies d_{\boldsymbol{x}}(f, f^*) = 0$
2. $d_{\boldsymbol{x}}(f, f^*) = 0 \implies f = f^*$ on $\boldsymbol{x}([0, T])$, the image of $\boldsymbol{x}$
3. For a sequence f_l such that $d_{\boldsymbol{x}}(f_l, f^*) \to 0$ it holds that $f_l \circ \boldsymbol{x} \to f^* \circ \boldsymbol{x}$ in the mean[4]

The third property tells us that if we keep minimising $d_{\boldsymbol{x}}(f, f^*)$, we can expect f getting closer to f^* on $\boldsymbol{x}([0, T])$, the image of $\boldsymbol{x}$.

A.3 Proof of the Theorem

Proof of Theorem 1. Let us consider a Hilbert space $L^2[0, T]$ where the inner product is denoted by $\langle \cdot, \cdot \rangle$ and defined as:

$$\langle a, b \rangle = \int_0^T a(t) b(t) dt \tag{18}$$

where a and b are real functions which are square-integrable on the interval $[0, T]$. The L^2 norm can be written as:

$$||a||_2^2 = \langle a, a \rangle \tag{19}$$

[3]It can be considered to be a metric through metric identification. We would need to declare two functions being equal if they are equal on $\boldsymbol{x}([0, T])$

[4]This can be strengthened to imply uniform convergence if the sequence f_l is *equicontinuous* (Ford & Pennline, 2007). This can be satisfied if, for instance, they all share the same Lipschitz constant. In particular, this is true if their derivatives are bounded by the same constant.

Using this notation we can rewrite $C_j(f, \widehat{\boldsymbol{x}}_k, g_s)$ as:

$$C_j(f, \widehat{\boldsymbol{x}}_k, g_s) = \langle f \circ \widehat{\boldsymbol{x}}_k, g_s \rangle + \langle \widehat{x}_{kj}, \dot{g}_s \rangle \tag{20}$$

We assume that $\widehat{\boldsymbol{x}}_k$ converges to $\boldsymbol{x}$ in mean. For vector-valued functions that means that

$$\int_0^T ||\widehat{\boldsymbol{x}}_k(t) - \boldsymbol{x}(t)||_2^2 \, dt \to 0 \text{ as } k \to \infty \tag{21}$$

Note that $|| \cdot ||_2$ in this equation denotes the usual Euclidean norm as the arguments are vectors not functions. This is equivalent to saying that all components of $\widehat{\boldsymbol{x}}$ converge to the corresponding components of $\boldsymbol{x}$. Thus

$$||\widehat{x}_{kj} - x_j||_2 \to 0 \text{ as } k \to \infty \, \forall j \tag{22}$$

As the inner product with one argument fixed can be seen as a continuous operator, we get:

$$\langle \widehat{x}_{kj}, \dot{g}_s \rangle \to \langle x_j, \dot{g}_s \rangle \text{ as } k \to \infty \tag{23}$$

We assume that f_j is λ-Lipschitz continuous. That means that

$$\begin{aligned} ||f \circ \widehat{\boldsymbol{x}}_k - f \circ \boldsymbol{x}||_2^2 &= \int_0^T |f(\widehat{\boldsymbol{x}}_k(t)) - f(\boldsymbol{x}(t))|^2 \, dt \\ &\leq \int_0^T \lambda^2 ||\widehat{\boldsymbol{x}}_k(t) - \boldsymbol{x}(t)||_2^2 \, dt \\ &= \lambda^2 \int_0^T ||\widehat{\boldsymbol{x}}_k(t) - \boldsymbol{x}(t)||_2^2 \, dt \to 0 \text{ as } k \to \infty \end{aligned} \tag{24}$$

This means that

$$\langle f \circ \widehat{\boldsymbol{x}}_k, g_s \rangle \to \langle f \circ \boldsymbol{x}, g_s \rangle \text{ as } k \to \infty \tag{25}$$

Combining equations (23) and (25) with expression (20), we conclude that

$$\lim_{k \to \infty} C_j(f, \widehat{\boldsymbol{x}}_k, g_s)^2 = (\langle f \circ \boldsymbol{x}, g_s \rangle + \langle x_j, \dot{g}_s \rangle)^2 \tag{26}$$

Now, using integration by parts we get

$$\langle x_j, \dot{g}_s \rangle = \int_0^T x_j(t)\dot{g}_s(t)dt = -\int_0^T \dot{x}_j(t) g_s(t) dt + x_j(T)g_s(T) - x_j(0)g_s(0) \tag{27}$$

We use the assumption that $g_s(0) = g_s(T) = 0 \, \forall s$ to obtain

$$\langle x_j, \dot{g}_s \rangle = -\langle \dot{x}_j, g_s \rangle \tag{28}$$

We can substitute it back into Equation 26 and use the linearity of the inner product to get

$$\lim_{k \to \infty} C_j(f, \widehat{\boldsymbol{x}}_k, g_s)^2 = \langle f \circ \boldsymbol{x} - \dot{x}_j, g_s \rangle^2 \tag{29}$$

Now we can use Parseval's identity and the fact that the set $g_1, g_2, ...$ forms a Hilbert basis for $L^2[0, T]$ to infer that:

$$\lim_{S \to \infty} \sum_{s=1}^{S} \langle f \circ \boldsymbol{x} - \dot{x}_j, g_s \rangle^2 = ||f \circ \boldsymbol{x} - \dot{x}_j||_2^2 \tag{30}$$

Substituting Equation 29 into Equation 30 and using the fact that the inner limit does not depend on S we obtain the statement in the theorem

$$\lim_{S \to \infty} \lim_{k \to \infty} \sum_{s=1}^{S} C_j(f, \widehat{\boldsymbol{x}}_k, g_s)^2 = ||f \circ \boldsymbol{x} - \dot{x}_j||_2^2 = ||(f - f^*) \circ \boldsymbol{x}||_2^2 \tag{31}$$

$\square$

A.4 Divergent derivatives

The main advantage of our algorithm lies in estimating the trajectory instead of estimating its time derivative. We show how the derivatives of a uniformly convergent sequence of functions may fail to converge, and we illustrate how it impacts objectives based on estimating the derivative.

We highlight the following mathematical fact:
Remark. Uniform convergence of functions does not guarantee convergence of the derivatives

This is in contrast to definite integrals where such implication holds. Intuitively, this follows from the fact that the derivative is a very local property of the function. To see an example of such a sequence, consider a sequence of functions (h_k) where $h_k : [0, 1] \to \mathbb{R}$ is given by

$$h_k(t) = \frac{sin(kt)}{k} \tag{32}$$

It can be shown that (h_k) converges to $h(t) = 0$ uniformly. However, the derivative of h_k, given by

$$h'_k(t) = cos(kt), \tag{33}$$

fails to converge at any point.

We use this observation to show that the square of $||\hat{\dot{x}}_j - f_j \circ \hat{\boldsymbol{x}}||_2$, where $\hat{\dot{x}}_j$ is the derivative calculated from the recovered trajectory $\hat{\boldsymbol{x}}$, does not converge to $d_{\boldsymbol{x}}(f_j, f_j^*)^2$ when $\hat{\boldsymbol{x}}$ converges to the ground truth trajectory $\boldsymbol{x}$.

Consider the following ODE ($J = 1, T = 1$):

$$\dot{x}(t) = x(t) \ \forall t \in [0, 1] \tag{34}$$

so $f^*(x) = x$. We choose $x(t) = e^t$ as the ground truth trajectory from which we sample. Let $(\hat{x}_k)$ be a sequence of trajectories given by $\hat{x}_k = e^t + \frac{sin(kt)}{k}$. It can be shown that $(\hat{x}_k)$ converges to x uniformly. Then for any candidate function f

$$\begin{aligned} ||\hat{\dot{x}}_k - f \circ \hat{x}_k||_2^2 &= \int_0^1 (\hat{\dot{x}}_k(t) - f(\hat{x}_k(t)))^2 dt \\ &= \int_0^1 \left(e^t + cos(kt) - f\left(e^t + \frac{sin(kt)}{k}\right)\right)^2 dt \end{aligned} \tag{35}$$

If $||\hat{\dot{x}}_k - f \circ \hat{x}_k||_2^2$ were to converge to $d_x(f, f^*)^2$ then in particular $||\hat{\dot{x}}_k - f^* \circ \hat{x}_k||_2^2$ should converge to 0. However, we show this is not what happens.

$$\begin{aligned} ||\hat{\dot{x}}_k - f^* \circ \hat{x}_k||_2^2 &= \int_0^1 \left(e^t + cos(kt) - f^*\left(e^t + \frac{sin(kt)}{k}\right)\right)^2 dt \\ &= \int_0^1 \left(e^t + cos(kt) - \left(e^t + \frac{sin(kt)}{k}\right)\right)^2 dt \\ &= \int_0^1 \left(cos(kt) - \frac{sin(kt)}{k}\right)^2 dt \\ &= \int_0^1 cos^2(kt) + \left(\frac{sin(kt)}{k}\right)^2 - 2cos(kt)\left(\frac{sin(kt)}{k}\right) dt \end{aligned} \tag{36}$$

As $\left(\frac{sin(kt)}{k}\right)^2$ and $2cos(kt)\left(\frac{sin(kt)}{k}\right)$ converge uniformly to a 0 function, we can use the linearity of integration to obtain

$$\lim_{k\to\infty} ||\hat{\dot{x}}_k - f^* \circ \hat{x}_k||_2^2 = \lim_{k\to\infty} \int_0^1 cos^2(kt)dt = \lim_{k\to\infty} \left(\frac{sin(2k)}{4k} + \frac{1}{2}\right) = \frac{1}{2} \neq 0 \tag{37}$$

This example shows that objectives that estimate the derivative are highly dependent on the type of estimation algorithm used. In contrast, Theorem 1 tells us that the only property of the estimation algorithm we need is that the trajectories converge to the ground truth in L^2 norm.

Algorithm 1 D-CODE.

```
Input: Dataset $\mathcal{D} = \{\boldsymbol{y}_i(t) | i \le N, t \in \mathcal{T}\}$
Input: Smoothing algorithm $\mathcal{S}$, numerical integration algorithm $\mathcal{I}$, optimization algorithm $\mathcal{O}$
Input: Testing functions $g_s,\ s \le S$, initial guesses $f_{j0},\ j \le J$.
$\widehat{\boldsymbol{x}}_i = \mathcal{S}\big(\boldsymbol{y}_i(t_1), \boldsymbol{y}_i(t_2), \dots\big),\ i \le N$     {Preprocessing step: smooth input trajectories}
for $j \in [1, J]$ do
    $\widehat{f}_j = f_{j0}$     {Initialize the $j^{\text{th}}$ ODE}
    Converge = False
    while Not Converge do
        obj $= \sum_{i=1}^{N} \sum_{s=1}^{S} C_j(f_j, \widehat{\boldsymbol{x}}_i, g_s)^2$     {Calculate objective (Eq. 5).}
        $\widehat{f}_j$, Converge $= \mathcal{O}($obj$, \widehat{f}_j)$     {Run optimization step.}
    end while
end for
Output: The discovered ODEs $\widehat{f}_j,\ j \le J$
```

B D-CODE FRAMEWORK

B.1 PSEUDOCODE

The Pseudocode is presented in Algorithm 1.

B.2 DISCOVER HIGHER ORDER AND NONAUTONOMOUS ODES

In the main text, we focused on the system of first order autonomous ODEs (Equation 1). This is because higher order or nonautonomous ODEs can be transformed into first order autonomous systems by including more variables. For instance, consider the damped, driven harmonic oscillator, a model system for classical mechanics (Simmons, 1972):

$$\ddot{x}_1 + \dot{x}_1 + x_1 - \cos(t) = 0$$

The ODE is second-order as it involves the acceleration term $\ddot{x}_1$ and it is nonautonomous because of the time-dependent term $\cos(t)$. If we introduce two new variables: $x_2 = \dot{x}_1$ and $x_3 = t$, we obtain the following first order autonomous system:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_2 - x_1 + \cos(x_3) \\ \dot{x}_3 &= 1 \end{aligned}$$

Since the first and the third equations are implied by the definition of x_2 and x_3, they do not need to be discovered from data. One can apply the discovery algorithm to uncover the second equation from the measurements of x_1, x_2 and x_3.

C DISCUSSION ON DATASET CURATION

The success of ODE discovery depends on the availability of a well-curated dataset. However, curating a dataset is a highly nontrivial task, and it is well beyond the scope of the current work. In fact, the curation of dataset is sometimes an important scientific contribution in itself (Blasius et al., 2020). Here, we provide a brief discussion on this topic.

Selection of variables. The researcher needs to decide which variables to include in the dataset — this defines the scope of problem. In general, variable selection depends on the scientific problem being addressed and the background knowledge about the interactions between variables. For instance, one may use the known chemical reaction network to choose a small set of compounds to include in the data. When such knowledge is unavailable, the researcher may need to perform pilot studies to find out the interaction network (otherwise they may face a challenging high-dimensional problem). We note that there is a well-developed literature on uncovering the interaction network of a high-dimensional dynamical systems (Mangan et al., 2016; Qian et al., 2020).

Measurement settings. In practice, the measurement settings are often affected by practical constraints, e.g. the precision of measurement device, the measurement cost, the time limit, etc. However, the researcher should ensure that the measurement horizon $[0, T]$ covers the pattern of scientific interest. To illustrate this point, consider an application in climate modeling. If we are interested in studying the cyclic behavior of the system (e.g. seasonality), the measurement should cover at least one entire period (one year). On the other hand, if we are interested in the system's response to an external stimuli (e.g. short-term weather change due to volcanic activity), the measurement should cover the period before and after the stimuli (volcano eruption).

Iterative nature of discovery. Scientific discovery is by no means a single-step monolithic process. Rather, it is an iterative process involving many components: hypothesis generation, data curation, modeling, validation, etc. D-CODE focuses on the modeling component, but it can facilitate the entire loop of discovery because it distills an interpretable close-form ODE. For instance, if the discovered ODE converges to a wrong stationary point (as indicated by domain knowledge), the researcher may curate an additional dataset with longer time horizon and re-train D-CODE with the additional information.

D Utilities of the discovered closed-form ODE

In the main text, we focus on the problem of distilling closed-form ODE from data. Here we illustrate how the learned closed-form ODE can facilitate various downstream tasks and applications.

Prediction. The learned closed-form ODE can be used to make predictions about the future states of a dynamical system. Formally speaking, given a sequence of past observations $\boldsymbol{y}(t_1), \boldsymbol{y}(t_2), \ldots, \boldsymbol{y}(t_k)$, we would like to predict $\boldsymbol{x}(t^*)$ for a future time $t^* > t_k$. This prediction problem is very well studied and has many proven algorithms (e.g. the family of Kalman filters (Welch et al., 1995) and the Bayesian methods (Doucet et al., 2000)).

Simulation and synthetic data. The learned closed-form ODE can also be used as a simulation model to generate synthetic datasets. The user can specify different initial conditions $\boldsymbol{x}(0)$ and different external signals (if applicable) to generate a range of trajectories. The simulated trajectories may be used in downstream tasks, e.g. computing the statistical properties.

Planning and control. In some cases, the system contains variables that can be intervened on (e.g. external forces), and we would like to manipulate these variables to maximize certain utility function. This is the canonical problem setting for Control Theory, where decades of research have been devoted to deriving the optimal policy from the closed-form governing equations and the utility function (Mehrmann, 1991; Haddad & Chellaboina, 2011).

Understanding the properties. Often we are interested in understanding the high-level properties of the dynamical system. Examples include asymptotic stability, fixed points, periodicity, bifurcation, and so on. The closed-form ODE would facilitate the study of these properties because the analytical and the computational tools developed in the dynamical system literature are both applicable (Browder, 1983; Lyapunov, 1992; Chow & Hale, 2012).

E Background on numerical differentiation

The task of estimating time derivative $\dot{x}$ from discretely sampled and noisy data is commonly referred to as *numerical differentiation*. Finite difference is a family of the most basic numerical differentiation methods (Bickley, 1941). For example, the central difference estimator calculates

$$\hat{\dot{x}} = \frac{y(t + \Delta t) - y(t - \Delta t)}{2\Delta t}$$

The main disadvantage of finite difference is that it is highly sensitive to measurement noise (Chartrand, 2011).

A common way to address this shortcoming is to estimate the derivative on the *smoothed* trajectory. For example, spline-smoothed differentiation smooths the observed trajectories with cubic spline and estimates $\dot{x}$ from the fitted spline functions (Shen et al., 1998; Zhou & Wolfe, 2000). (This method is used in the benchmarks SR-S). However, this family of methods is usually based on assumptions on

Table 3: The detailed settings for each simulation: noise level σ_R, time step size Δt, number of trajectories N, time horizon T, range of initial conditions $[a, b]$, numeric constants θ. Bold values are the defaults.

System	σ_R	Δt	N	T	$[a, b]$	θ
Gompertz	$[0.01, 1.3]$	**0.1**, 0.2, 0.5, 1	5, 20, **50**, 100	4	$[0, 0.01]$	$\theta_1 = \theta_2 = 1.5$
Logistic	$[0.01, 1.3]$	0.1, **0.2**, 0.5, 1	5, 20, **50**, 100	10	$[0, 0.1]$	$\theta_1 = 1, \theta_2 = 0.5$
Glycolytic	0.01, 0.1, 0.2	**0.1**, 0.2, 0.5, 1	5, 20, **50**, 100	15	$[0, 0.1]$	$\theta_1 = 0.75, \theta_2 = 0.1$
Lorenz	$[0.01, 0.3]$	0.01, 0.02, **0.04**, 0.1	5, 20, **50**, 100	10	$[0, 10]$	$10, 28, 8/3$

the smoothness of the trajectory and its (higher-order) derivatives (Zhou & Wolfe, 2000): they encode a bias towards smooth and slow-changing derivatives. Such bias may the hamper the estimation of dynamical systems with abrupt changes (Chartrand, 2011).

To address this challenge, Chartrand (2011) proposes to use explicit regularization on the finite difference estimator. As we pointed out in Appendix A, the standard L2 regularization is not suitable for derivative estimation. The authors propose to use total variation regularization, a method originally developed in computer vision to obtain sharp denoised images (Strong & Chan, 2003). This type of regularization allows rapid (or even discountinous) change in $\dot{x}$. The method has been adopted in the two-step sparse regression literature (Brunton et al., 2016), and it is used in the benchmark SR-T.

F Implementation details of the experiments

F.1 Detailed settings for each experiment

The detailed settings for each experiment in Section 5 is shown in Table 3. The time horizon T and the range of initial conditions $[a, b]$ are chosen based on the scale and properties of the system. For example, we observe that the Gompertz model reaches saturation and converges to the constant $x = 1$ before $t = 4$ (hence we set the time horizon $T = 4$). In practical applications, the time horizon is decided by the problem scope and the availability of experimental or observational data (see the discussion on dataset curation in Appendix C).

F.2 Hyper-parameter settings

Smoothing algorithm. In the experiments, D-CODE uses Gaussian process in the pre-processing step. We use the radial basis kernel with the hyper-parameters tuned based on marginal log-likelihood (Bernardo et al., 1998).

Genetic programming. The hyperparameters of genetic programming is decided based on a pilot study. In the pilot study, we consider a noiseless and frequently sampled Gompertz growth model ($\sigma = 0$, $\Delta t = 0.01$). We use randomized search on Two-step Symbolic Regression to find the hyperparameters that achieves the best RMSE loss (Table 1). The same hyperparameters are used by all algorithms on all experiments. The values are listed below. We refer the reader to the documentation of `gplearn` library for a detailed explanation of these hyperparameters (Stephens, 2021).

1. population size: 15000
2. tournament size: 20
3. p crossover: 0.6903
4. p subtree mutation: 0.133
5. p hoist mutation: 0.0361
6. p point mutation: 0.0905
7. generations: 20
8. parsimony coefficient: 0.01

The set of allowed mathematical operations is

$$\{+, -, \times, \div, \log, \cos, \text{power}\}$$

D-CODE and all benchmarks use the same set of mathematical operations for fairness of comparison. To speed up computation, we run a pilot study for each dynamical system in the noiseless setting described above to select a subset of these operations for the main experiments.

Numerical differentiation. We use the python package `derivative` for numerical differentiation (Quade & Goldschmidt, 2020). The library contain implementations of the methods discussed in Appendix E. Currently, there lacks a principled way to tune the hyperparameters of numerical differentiation because the time derivative is not observed. Existing works often rely on heuristics (Chartrand, 2011). Here, we tune the hyperparameters based on the *true derivative* using grid search over $(0.001, 0.01, 0.1, 1, 10, 100, 1000)$. Note that this procedure may allow the time derivative to be estimated more accurately than what is possible (because the true derivative is not available in real data). Hence, it will give an *advantage* to SR-T and SR-S because D-CODE does not use the true derivative in any way. SR-G uses the same Gaussian process as D-CODE, which is tuned by the marginal log-likelihood on the observed trajectory.

Correctness of the functional form. As discussed in Section 2, the functional form of a closed-form function f is the expression with all the numeric constants replaced by placeholders $\theta_1, \theta_2, \ldots$ In practice, the uncovered function $\hat{f}$ may contain constants that are close to 0. For instance $f(x) = x + 0.001x^2$ is close to $f(x) = x$ when x is small. Hence, we apply thresholding on the estimated constants and set all constants between $[-0.05, 0.05]$ as 0 (only for logistic model and Glycolytic oscillator, where we found thresholding to be helpful). The thresholding is applied to the results of *all* methods to evaluate success probability of recovering the functional form – and it is *not* used when calculating the RMSE of parameter estimation or the distance $d_{\boldsymbol{x}}$. We use Python package `sympy` to decide whether the discovered functional form is correct (Meurer et al., 2017), i.e. success when `simplify`$(f^* - \hat{f})$ returns 0.

F.3 Cleaning tumor growth dataset

The original tumor growth dataset (Section 6) contains heterogeneous growth patterns and outliers. However, the dataset does not include any patient covariate that could help to model or explain the heterogeneity, i.e. some variables that would drive the tumor growth dynamics are not recorded in the data, e.g. age and comorbidity (see the discussion in Section 7). To address this issue, we performed k-means clustering on the observed trajectories (with 10 clusters), such that the growth pattern within each cluster is more or less homogeneous. We selected the largest cluster with 310 trajectories for the analysis.

G Additional results

G.1 Additional evaluation metric: $d_{\boldsymbol{x}}$

Here we report the distance function $d_{\boldsymbol{x}}(\hat{f}, f^*)$ between the true and estimated ODEs. This metric tracks how well $\hat{f}$ approximates f^*. The rationale for reporting this metric is that it may be possible that $\hat{f}$ has the wrong functional form but it still well approximates f^*. Hence, we need the distance $d_{\boldsymbol{x}}$ in addition to the discovery probability.

The results are presented in Figure 7 and 8. We observe that the patterns of $d_{\boldsymbol{x}}$ and discovery probability (reported in the main text Section 5) are very similar — the performance gap between the methods is minor in the low-noise frequent-sampling scenarios, but D-CODE significantly outperforms the benchmarks in the more challenging measurement settings. The agreement between $d_{\boldsymbol{x}}$ and discovery probability suggests that when the correct functional form is not recovered, the estimated $\hat{f}$ tends to perform poorly in approximating the true function as well.

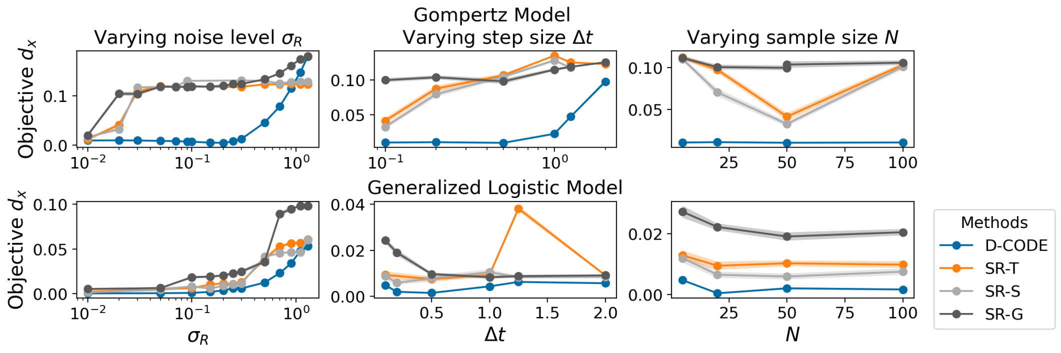


Figure 7: The metric $d_{\boldsymbol{x}}$ (Eq. 6) for the two growth models under different settings (**smaller better**). The three columns correspond to various noise levels σ_R, step sizes Δt, and numbers of trajectories N. The two rows correspond to the Gompertz model and the generalized logistic model respectively.

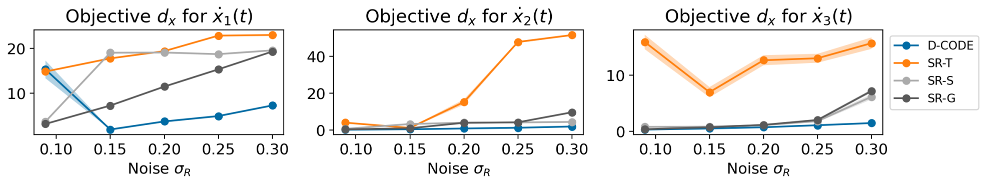


Figure 8: The metric $d_{\boldsymbol{x}}$ (Eq. 6) for the chaotic Lorenz system under different noise levels (**smaller better**).

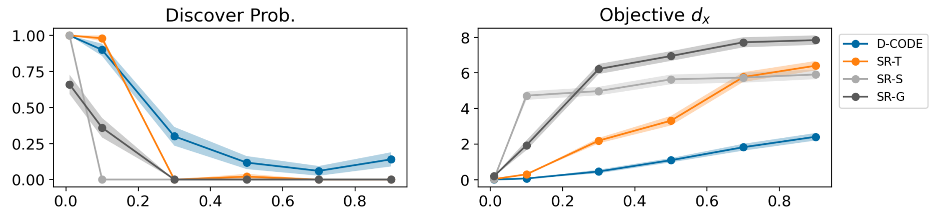


Figure 9: Simulation results for the dynamical system in Eq 38 under various noise levels. Left: the success probability (higher better) and Right: distance $d_{\boldsymbol{x}}$ (lower better).

G.2 Additional dynamical systems

Here we consider an additional dynamical system, which is an extension to the quadratic integrate and fire (QIF) model in Neuroscience (Brunel & Latham, 2003). The governing equation is

$$\dot{x} = \frac{x^2}{t + \theta} \tag{38}$$

We highlight that this ODE is *time-dependent* and it involves a *fraction*, which makes it impossible to be symbolically represented in a linear form $f(x) = \sum_k \theta_k b_k(x)$ (as required by the two-step sparse regression method). A similar example was given in Appendix B-2 of Brunton et al. (2016) to illustrate this failure mode of two-step sparse regression method.

The simulation results are shown in Figure 9. We observe the same pattern as the other dynamical systems considered in the main text.

G.3 Sensitivity analysis for testing function

Here we study the sensitivity of D-CODE to the choice and number of testing functions. We consider two families of testing function: sine functions and cubic spline functions, both of which satisfy the

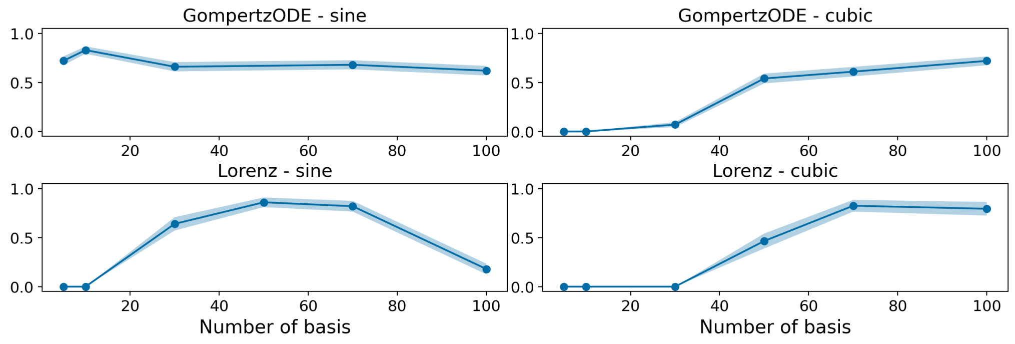


Figure 10: The success probability with different families and different numbers of testing functions g (sine and cubic spline).

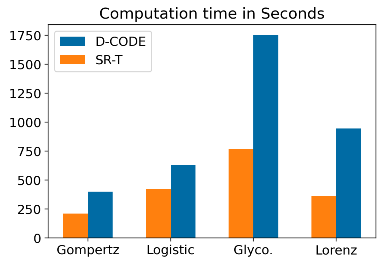


Figure 11: Comparison of average training time.

conditions in Section 4.2. We refer the readers to De Boor (1978) for the exact recursive definition of cubic spline functions.

Figure 10 reports the discovery probability for Gompertz model (first row) and Lorenz system (second row). We first observe that using too few testing functions (i.e. < 40) leads to performance loss. This agrees with Theorem 1 because the objective function Eq. 5 cannot well-approximate the distance $d_{\boldsymbol{x}}$ (Eq. 6) with too few testing functions. We also observe that the performance is reasonably stable when enough testing functions are included (e.g. 40 - 80 sine functions, and 80 - 100 cubic splines). Finally, we see a performance drop for using too many (e.g. 100) sine functions. After investigation, we find that the drop is due to the numerical error in the integration procedure — sine functions with very high frequency are oscillating too rapidly for accurate numerical integration. One may resolve this issue by using a smaller integration step size or a more advanced numerical integration method.

For practical applications, the user may experiment with different numbers of testing functions and examine the discovered ODEs. A key advantage of discovering close-form ODEs is that it allows the user to examine and refine the ODEs in an iterative way.

G.4 Training time

Figure 11 compares the average training time of D-CODE and two-step symbolic regression (SR-T). We expect D-CODE to have longer training time due to the numerical integration step in evaluating the functional C (Eq. 3). However, even for the longest-running task, D-CODE finishes training within 30 minutes on average. This can still be a very significant speed up compared with manual discovery by human experts. Experiments are run on a computer with Intel Xeon E3-12xx v2 CPU (16 cores) and 60 GB memory.

G.5 Table of discovered ODEs

The top-3 most common equations discovered by different methods for each dynamical system (among the 100 simulation runs) are listed in the table on the next page.

Table 4: The top-3 most common equations discovered by different methods for each dynamical system (among the 100 simulation runs). "Freq" denotes the chance of discovering the equation. We make the following observations: (1) SR-T and SR-S produce *over-simplistic* equations for the growth models (constant or zero), possibly because the estimated derivative $\widehat{\dot{x}}$ is too noisy to learn useful signal. (2) SR-T and SR-S may consistently *miss* a variable (e.g. the x_0 term in Glycolytic 1) or miss an interaction (e.g. the $x_0 x_2$ term in Lorenz 2). (3) SR-T and SR-S may include *unnecessary* terms in the equation (e.g. Lorenz 3). The measurements settings are the default ones listed in Table 3.

	D-CODE		SR-T		ST-S	
System	Discovered Equation $\widehat{f}$	Freq.	Discovered Equation $\widehat{f}$	Freq.	Discovered Equation $\widehat{f}$	Freq.
	$-\theta_1 x \log(\theta_2 x)$	1	θ_1	0.63	θ_1	0.67
Gompertz	-	-	$\theta_1 x - \theta_2 x^{\theta_3}$	0.04	$-\theta_1 x^{\theta_2}(x^{\theta_3} + \log(x))$	0.03
	-	-	$x^2 \log(x) \log(x^2)$	0.04	$-\log(\log(\theta_1 + x)$	0.03
	$\theta_1 x - x^{\theta_2}$	1	0	0.96	0	0.98
Logistic	-	-	θ_1	0.02	θ_1	0.02
	-	-	$\theta_1 x^{\theta_2} - x^{\theta_3}$	0.02	-	-
	$-\theta_1 x_0 + \theta_2 - x_0 x_1^2$	0.38	$\theta_1 - x_0 x_1^2$	0.5	$\theta_1 - x_0 x_1^2$	0.1
Glycolytic 1	$-\theta_1 x_0 - \theta_2 x_1 + \theta_3 - x_0 x_1^2$	0.03	$-\theta_1 x_0 + \theta_2 - x_0 x_1^2$	0.22	$-\theta_1 x_0 x_1^2 + \theta_2$	0.08
	$\theta_1 - x_0(\theta_2 + x_1^2)$	0.03	$-\theta_1 x_0 x_1^2 - \theta_2 x_0 + \theta_3$	0.04	$-\theta_1 x_0 x_1(\theta_2 + x_1) + \theta_3$	0.07
	$\theta_1 x_0 + x_0 x_1^2 - x_1$	0.78	$\theta_1 x_0 + x_0 x_1^2 - x_1$	0.75	$-\theta_1 x_1 + \theta_2 x_1^2 + \theta_3 + x_0 x_1$	0.09
Glycolytic 2	$\theta_1 x_0 + \theta_2 + x_0 x_1^2 - x_1$	0.13	$x_0(\theta_1 + x_1^2) - x_1$	0.09	$\theta_1 x_0 x_1^2 + \theta_2 x_0 - \theta_3 x_1$	0.08
	$\theta_1 x_0 + \theta_2 x_1(\theta_3 x_0 + x_0 x_1^2)$	0.01	$\theta_1 x_0 x_1^2 + \theta 3 x_0 - \theta_4 x_1$	0.02	$-\theta_1 x_1 - \theta_2 x_1^2 + \theta_3 + x_0 x_1 + x1^2$	0.06
	$-\theta_1 x_0 + \theta_2 x_1$	1	$-\theta_1 x_0 + \theta_2 x_1$	1	$-\theta_1 x_0 + \theta_2 x_1$	0.63
Lorenz 1	-	-	-	-	$-\theta_1 x_0 + \theta_2 x_1 + \theta_3$	0.37
	-	-	-	-	-	-
	$\theta_1 x_0 - x_0 x_2 - x_1$	0.33	$-\theta_1 x_0 + \theta_2 x_1 + \theta_3 x_2 - \theta_4$	0.99	$x_0(\theta_1 - x_2)$	0.73
Lorenz 2	$x_0(\theta_1 - x_2) + x_0 - x_1$	0.28	$\theta_1 x_0(-\theta_2 x_2 + \theta_3) + \theta_4 x_1 - x_0$	0.01	$\theta_1 x_0 - x_0 x_2$	0.2
	$x_0(\theta_1 - x_2) - x_1$	0.28	-	-	$x_0(\theta_1 - x_2) - x_0$	0.03
	$-\theta_1 x_2 + x_0 x_1$	0.8	$\theta_1 x_2 - \theta_2 + x_0 x_1 + x_0$	0.46	$\theta_1 x_2 + x_0 x_1$	0.51
Lorenz 3	$-\theta_1 x_2 + \theta_2 + x_0 x_1$	0.13	$-\theta_1 x_2 + x_0 x_1 + x_0$	0.11	$\theta_1 x_2 + x_0 x_1 + x_0$	0.11
	$-\theta_1 x_2 - \theta_2 + x_0 x_1$	0.08	$\theta_1 x_0 x_1 - \theta_2 x_2$	0.1	$\theta_1 x_2 + x_0 x_1 + x_1$	0.09

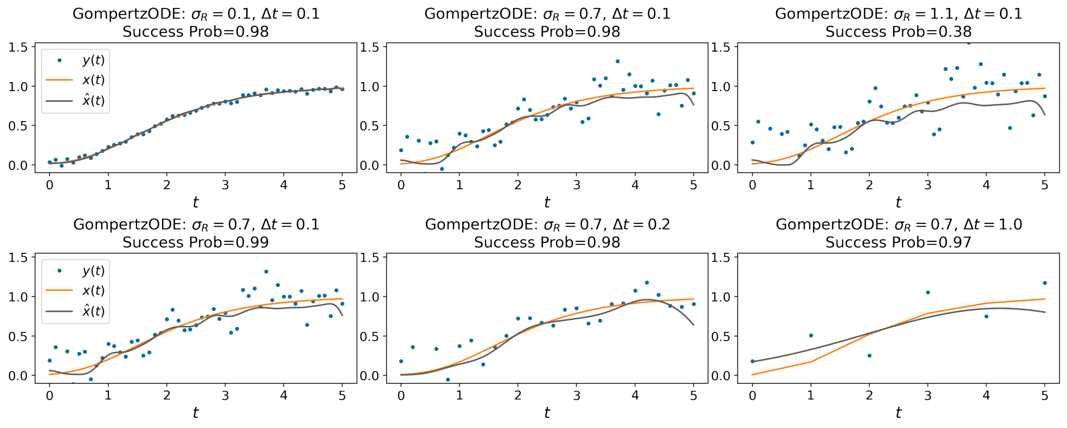


Figure 12: Visualization of the observations $\boldsymbol{y}(t)$, the true trajectory $\boldsymbol{x}(t)$ and the smoothed trajectory $\hat{\boldsymbol{x}}(t)$. **First row**: increasing noise level σ_R. **Second row**: increasing sampling sparsity Δt. The success probability of D-CODE is shown in the headings.

G.6 Visualization of smoothed and true trajectories

In this section we study whether D-CODE can still discover the ODE when the smoothed trajectory $\hat{\boldsymbol{x}}(t)$ is significantly different from the true trajectory $\boldsymbol{x}(t)$; for instance, when the signal-to-noise ratio is low or the observations are sparse.

In Section 5 we provided quantitative evaluation under scenarios with different noise levels (σ_R) and sampling sparsity (Δt).[5] For further illustration, here we visualize $\hat{\boldsymbol{x}}(t)$ and $\boldsymbol{x}(t)$ together with the observations $\boldsymbol{y}(t)$ in Figure 12. In the most favorable setting (top left), $\hat{\boldsymbol{x}}(t)$ is indistinguishable from $\boldsymbol{x}(t)$, and D-CODE achieves very high success probability as expected. As the observation setting becomes more challenging (middle column), $\hat{\boldsymbol{x}}(t)$ may recover the overall trend of $\boldsymbol{x}(t)$ but it may also contain some estimation artifacts (e.g. the unnecessary zigzag pattern). However, D-CODE is still able to recover the true ODE with very high probability. In the most challenging settings (right column), $\hat{\boldsymbol{x}}(t)$ differs significantly from $\boldsymbol{x}(t)$ for extended periods of time (top right $t > 2$, bottom right $t < 2$ and $t > 4$). Yet, D-CODE still makes successful discovery at times.

H Discussion on alternative methods

In this section, we discuss the alternative methods that, in theory, could also be applied to discovering closed form ODEs from data. However, as we will show, these methods suffer from various drawbacks that may severely limit their practical utility.

H.1 Double optimization method

The double optimization method (DO) was first proposed in the literature of ODE parametric estimation, where the functional form of f is known but it contains unknown parameters θ (see Ramsay et al. (2007) Section 1.3 for a review).[6] DO solves the following optimization problem (we consider the one-dimensional case to simplify notations):

$$\hat{f} = \arg\min_{f} \left[\min_{\hat{x}_i(0)} \sum_{i=1}^{N} \sum_{t \in \mathcal{T}} \left(y_i(t) - \hat{x}_i(t)\right)^2 \right], \quad \text{subject to } \hat{x}_i(t) = \hat{x}_i(0) + \int_0^t f(\hat{x}_i(s))ds \quad (39)$$

This is a nested optimization problem where the outer part searches for the unknown function f and the inner part searches for the unknown initial conditions $x_i(0)$ given the current guess of f. Note

[5]Note that, by definition, the signal-to-noise ratio is the inverse of σ_R, i.e. $1/\sigma_R$.

[6]DO is also referred to as the nonlinear least square method (NLS) in some literature.

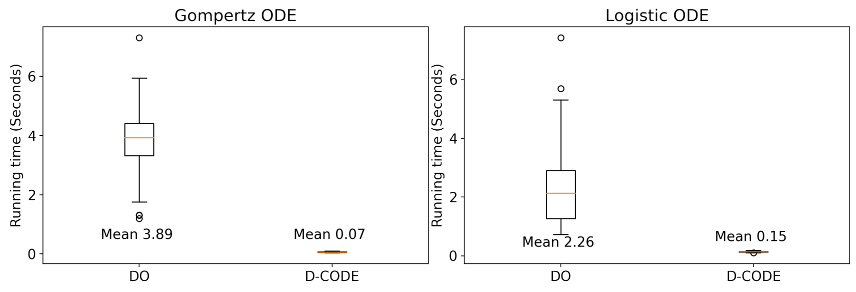


Figure 13: The computation time (in seconds) to evaluate a candidate function f. The data are generated from Gompertz and Logistic ODEs respectively. The results are calculated from 100 independent runs. The summary statistics (mean, median and quartiles) are shown in the boxplot. DO is 30 - 50 times slower than D-CODE.

that the inner and outer parts are *coupled* because the optimal $\hat{x}_i(0)$ depends on f, e.g. the best-fitting exponential curve would have a different initial condition from the best linear fit.

A well-known shortcoming of DO is its computational complexity (Gugushvili & Klaassen, 2012). The inner optimization over $x_i(0)$ is typically non-convex and may take many steps to converge. To further aggravate the problem, each inner optimization step requires solving the IVP numerically.

The computational challenge becomes even more pronounced in our setting, where the functional form of f is also unknown. Genetic programming needs to evaluate many candidate functions f but DO requires solving the inner optimization for each evaluation. In contrast, D-CODE (Equation 5) does not involve any nested optimization.

To illustrate this point, we have implemented DO and measured its computation time to evaluate a candidate f. The results are shown in Figure 13. We found that DO is typically 30 - 50 times slower than D-CODE. Its computational cost is too high to be considered as a feasible method for discovering closed-form ODEs.

H.2 Neural ODE with post-hoc distillation

Although Neural ODE (NODE) does not directly produce closed-form ODEs, one may attempt to distill a closed-form equation from the trained network. The method takes two steps: (1) train a NODE f_N using observed trajectories and query the trained NODE to produce a dataset $\{\boldsymbol{x}_i(t), f_N(\boldsymbol{x}_i(t))\}_{i,t}$; (2) apply symbolic regression on this dataset. This approach can also be viewed as a two-step regression method (Section 3), where the derivatives are estimated by a NODE.

The standard approach to train a NODE for time series modeling is to minimize the error over a prediction horizon $\mathcal{T}^* = [t_1, t_2, \ldots, T^*]$ (Rubanova et al., 2019), i.e. to minimize

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{t \in \mathcal{T}^*} ||\boldsymbol{y}_i(t) - \hat{\boldsymbol{x}}_i(t)||_2^2, \tag{40}$$

where $\hat{\boldsymbol{x}}_i(t)$ is obtained by solving the IVP of the NODE.

However, the choice of prediction horizon T^* is non-trivial and plays an important role in the training. For chaotic systems, the user should choose a smaller T^* because the system is unpredictable in the long term. On the other hand, making T^* too small may lead to a myopic model that fails to capture long-term patterns (Schmidt et al., 2020; Kantz & Schreiber, 2004).[7] In practice, the prediction horizon T^* is a hyper-parameter that may be difficult to decide (note that the Lyapunov time of the true system is unknown). In contrast, D-CODE does not require user to specify the prediction horizon T^*.

Furthermore, as we alluded to earlier, this approach is an instance of two-step regression, where the derivative is estimated by NODE instead of numerical differentiation. Although NODE is known to

[7]The limitations of minimizing short-term prediction loss for sequence modeling is well-recognized in machine learning; for example, the sequence-to-sequence models tackle this issue by minimizing the loss over a sequence of predictions rather than the one step ahead prediction.

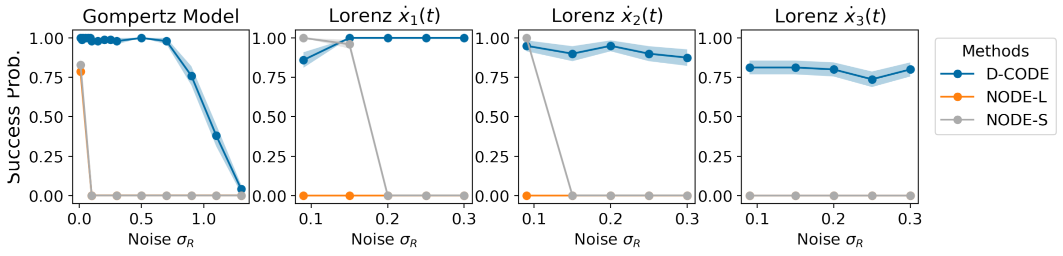


Figure 14: The probability of successfully recovering the unknown ODEs. Two versions of NODE are compared with D-CODE on the Gompertz ODE (first panel) and the Lorenz system (second to fourth panels). In the rightmost panel, the results of NODE-L and NODE-S overlap and only one line is displayed.

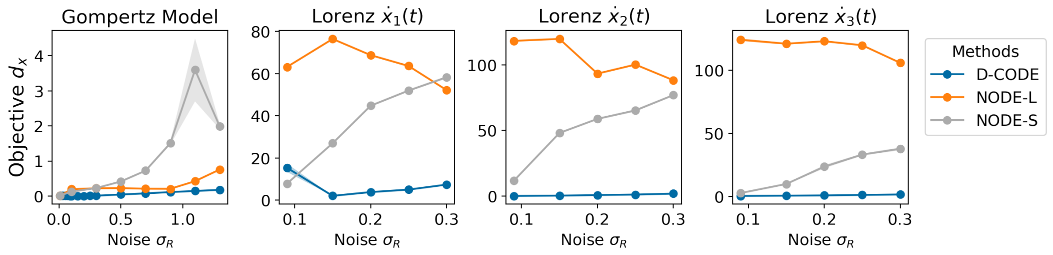


Figure 15: The distance $d_{\boldsymbol{x}}$ between the true and the learned ODE.

be a universal function approximator, its estimation efficiency (e.g. $\sqrt{n}$ consistency), finite sample performance, and robustness to noise are still largely under investigation. Hence, we should remind the readers that NODE is not guaranteed to outperform numerical differentiation in estimating derivatives.

To illustrate these points, we evaluate the performance of NODE with post-hoc distillation. We consider two versions of NODE: the **NODE-L** is trained on long term prediction ($T^* = 25$) and the **NODE-S** is trained on short term prediction ($T^* = 0.04$). The evaluation is performed on the Gompertz ODE (non-chaotic, one dimensional) and the Lorenz system (chaotic, three dimensional).

Figure 14 shows the success probability (bigger better) and Figure 15 shows the distance $d_{\boldsymbol{x}}$ (Equation 6, smaller better). For the non-chaotic Gompertz ODE, both NODE-L and NODE-S are only able to recover the correct functional form when the noise is very close to zero ($\sigma_R = 0.01$). Note that this behavior is very similar to other two-step regression methods (SR-T, SR-S and SR-G, Figure 3). We also observe that the NODE-L achieves consistently better $d_{\boldsymbol{x}}$ than NODE-S. This confirms that training over longer time horizon leads to better estimate of the ODE when the system is non-chaotic.

Turning to the chaotic Lorenz system, we observe that NODE-L always fails to discover the correct ODE and its $d_{\boldsymbol{x}}$ is higher than the other two methods. This verifies our expectation that training with long-term prediction is not feasible for chaotic systems. As before, NODE-S performs well only when the noise level is relatively small. We also observe that its $d_{\boldsymbol{x}}$ monotonically increases with the noise level. This suggests that higher noise level significantly reduces NODE-S's ability to approximate the unknown ODE, limiting its performance on noisy data.