RESEARCH-ARTICLE

When Data Transformations Mislead Symbolic Regression: Deceptive Search Spaces in System Identification

ALBERTO PAOLO TONDA, National Research Institute for Agriculture, Food and Environment, Paris, Ile-de-France, France

HENGZHE ZHANG, Victoria University of Wellington, Wellington, WGN, New Zealand

QI CHEN, Victoria University of Wellington, Wellington, WGN, New Zealand

BING XUE, Victoria University of Wellington, Wellington, WGN, New Zealand

MENGJIE ZHANG, Victoria University of Wellington, Wellington, WGN, New Zealand

ÉVELYNE LUTTON, National Research Institute for Agriculture, Food and Environment, Paris, Ile-de-France, France

Published: 14 July 2025

GECCO '25 Companion: Genetic and Evolutionary Computation Conference Companion
July 14 - 18, 2025
Malaga, Spain

Conference Sponsors:
SIGEVO

https://doi.org/10.1145/3712255.3734301
ISBN: 9798400714641

When Data Transformations Mislead Symbolic Regression: Deceptive Search Spaces in System Identification

Alberto Tonda
alberto.tonda@inrae.fr
UMR 518 MIA-PS, INRAE
Univ. Paris-Saclay, Palaiseau, France
UAR 3106 ISC-PIF CNRS
Paris, France

Hengzhe Zhang
hengzhe.zhang@ecs.vuw.ac.nz
Victoria University of Wellington
Wellington, New Zealand

Qi Chen
qi.chen@ecs.vuw.ac.nz
Victoria University of Wellington
Wellington, New Zealand

Bing Xue
bing.xue@ecs.vuw.ac.nz
Victoria University of Wellington
Wellington, New Zealand

Mengjie Zhang
mengjie.zhang@ecs.vuw.ac.nz
Victoria University of Wellington
Wellington, New Zealand

Evelyne Lutton
evelyne.lutton@inrae.fr
UMR 518 MIA-PS, INRAE
Univ. Paris-Saclay, Palaiseau, France
UAR 3106 ISC-PIF CNRS
Paris, France

Abstract

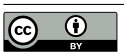
System identification is the task of automatically learning the model of a dynamical system, which can be represented as a system of ordinary differential equations (ODEs), using data points from time-series trajectories. This challenge has been addressed through various methods, including sparse regression, specialized neural networks, and symbolic regression. However, applying standard symbolic regression requires transforming the trajectory data to frame the problem as learning a set of regular equations. This study presents a first comprehensive comparison of the two most common data transformation approaches for system identification, evaluating their performance on a recently published benchmark suite of ODE systems. Our findings reveal that both approaches are highly sensitive to even moderate amounts of added noise, to different degrees. More surprisingly, we also show that data transformations can generate misleading search spaces, even under noise-free conditions. Further analysis indicates that reducing the data sampling step size significantly improves performance, suggesting that both transformation techniques are also affected by sampling frequency, and indicating possible future directions of research for system identification using symbolic regression.

CCS Concepts

• **Theory of computation → Genetic programming**; • **Mathematics of computing → Ordinary differential equations**; • **Computing methodologies → Supervised learning by regression**.

Keywords

Genetic programming, Ordinary differential equations, Symbolic regression, System identification



GECCO '25 Companion, July 14–18, 2025, Malaga, Spain

ACM ISBN 979-8-4007-1464-1/2025/07
https://doi.org/10.1145/3712255.3734301

ACM Reference Format:
Alberto Tonda, Hengzhe Zhang, Qi Chen, Bing Xue, Mengjie Zhang, and Evelyne Lutton. 2025. When Data Transformations Mislead Symbolic Regression: Deceptive Search Spaces in System Identification. In *Genetic and Evolutionary Computation Conference (GECCO '25 Companion), July 14–18, 2025, Malaga, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3712255.3734301

1 Introduction

Systems of differential equations are extensively employed in all fields of science and engineering to model a wide range of dynamical phenomena, from the evolution of predator-prey populations over time [21] to the development of atmospheric convection over time [20]. Scientists have traditionally worked on handcrafting equations to reverse-engineer the phenomena they were studying. Thanks to the latest development in machine learning it is now possible to perform *system identification* to automatically obtain predictive models of dynamical systems directly from time-series trajectory data, for example using recurrent neural networks [26]. Approaches based on black-box models, however, lose desirable properties of transparency and interpretability, and not having direct access to the underlying equations makes it impractical to analyze key properties of the systems, like asymptotic stability [22].

Solutions based on Genetic Programming (GP) [18] and in particular Symbolic Regression (SR) can overcome this issue, and learn human-readable models of dynamical systems from data. Ad-hoc SR frameworks specifically tailored for the system identification task have been proposed in literature [3, 15], but since most of the research effort on SR targets regular equations, it is not straightforward to adapt the latest developments of SR [6, 29] to the discovery of differential equations. In order to apply standard SR algorithms to the system identification problem, different data transformation approaches have been presented [13, 18, 25], all with the common goal of turning the problem from directly learning the equation for a derivative to discovering a regular equation from which the original derivative can later be obtained. The proposed transformations start from data acquired from one or more trajectories describing the development of the dynamical phenomenon over

$$\begin{cases} x_0'(t) &= x_0(x_2 - 0.7) - 3.5x_1 \\ x_1'(t) &= 3.5x_0 + x_1(x_2 - 0.7) \\ x_2'(t) &= 0.1x_0^3 x_2 - 0.3x_2^3 + 0.95x_2 - \\ & \quad -(x_0^2 + x_1^2)(0.25x_2 + 1.0) + 0.65. \end{cases}$$

$$x_0(0) = 0.1, x_1(0) = 0.05, x_2(0) = 0.05$$

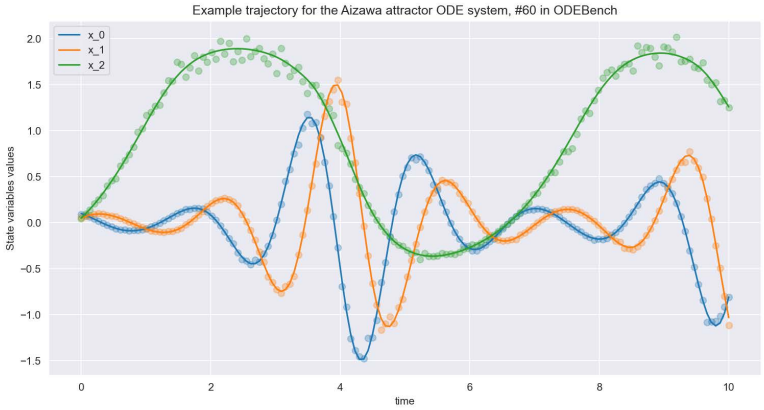


Figure 1: Example of a 3-variable ODE system, and the corresponding trajectories for the given initial conditions. The system in the example is Aizawa attractor [19], set with parameters and initial conditions from the ODEBench system identification benchmark suite [7, 8]. The dots represent system values with added noise, the colored lines are the original trajectories.

time, and generate a new search space to which standard SR can be applied. While each related work shows good performance of the data transformation on selected well-known dynamical systems, it is interesting to notice that, to the best of the authors' knowledge, they have never been compared on a large benchmark; this is also due to the fact that a specific SR benchmark specifically dedicated to system identification did not exist, until recently. The publication of the ODEBench system identification benchmark suite [7, 8], focused on systems of Ordinary Differential Equations (ODEs), finally offers the opportunity of performing a fair and thorough comparison.

In this work, we aim to present such a comparison, for data transformation approaches targeting standard SR. We are particularly interested in investigating: (i) whether the new search space created by each transformation can be efficiently explored by SR; and (ii) how the presence of noise can impact the accuracy of the transformations. Experimental results show that, while the transformations are generally effective, they can in some cases generate a misleading search space for SR; in other words, a search space where equations different from the ground truth might have a better value of the fitness function than the ground truth itself. Surprisingly, for some systems this happens even in absence of noise. Further experiments reveal a strong correlation between the sampling step Δt in the original trajectory data and the performance of the transformations, hinting that data transformations might deliver unsatisfactory results for trajectories where the value of the derivative changes quickly with respect to Δt.

The rest of the paper is organized as follows. Section 2 summarizes the necessary background knowledge on ODE systems, system identification, and data transformation approaches for SR; Section 3 introduces the proposed methodology for comparison, the benchmark suite and the selected performance evaluation metric. Section 4 presents the results of the experimental evaluation, while Section 5 concludes the paper with recommendations for future works.

2 Background

This section summarizes the minimal elements necessary to introduce the scope of the current work.

2.1 Systems of ordinary differential equations

An ODE system is a set of ordinary differential equations which describes the evolution of multiple interdependent variables as functions of a single independent variable, typically time. Each equation in the system specifies how the rate of change (derivative) of one variable depends on the variables in the system, and possibly on the independent variable.

Several notations can be used to describe the first derivative of a function x with respect to time t: $\frac{dx}{dt}$ (Leibniz notation), $x'(t)$ (Prime/Euler's notation), $\dot{x}(t)$ (Dot notation), $\dot{x}$ (Newton notation), $D[x(t)]$ (Functional notation) and so on. In the scope of this paper we will use Euler's notation $x'(t)$. A generic ODE system can thus be written as:

$$\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t), t) \tag{1}$$

where $\mathbf{x}(t) = [x_1(t), x_2(t), ..., x_n(t)]$ are the state variables, $\mathbf{f}(\mathbf{x}(t), t) = [f_1(\mathbf{x}, t), f_2(\mathbf{x}(t), t), ..., f_n(\mathbf{x}, t)]$ are the equations corresponding to the derivatives of the state variables, and t is the independent variable, typically time.

An ODE system can be solved for given initial conditions $x_0 = x(t_0)$, a time step Δt and a maximum time T, generating a dynamic that provides the value of its state variables ($\mathbf{x}$ in this example) for all the considered time instants in $[t_0, T]$. Such a dynamic can be called *trajectory*, *solution curve*, *evolution*, or *path*, depending on the application domain: In the following, we will use the term *trajectory*. An example of a 3-variable ODE system, describing the Aizawa attractor [19], is presented in Figure 1.

2.2 System identification

The goal of system identification is to automatically obtain a predictive model describing a dynamical phenomenon, directly from data

points associated to one or more trajectories starting from given initial conditions. A desirable output of system identification would be a set of human-interpretable equations. As the search space describing all ODE systems is too vast to be explored exhaustively, several solutions have been proposed in literature.

Sparse Identification of Nonlinear Dynamics (SINDy) [2] is an algorithm performing sparse linear regression on a manually predefined set of basic functions, implementing non-linear relationships. SINDy is highly computationally efficient, but it can only obtain models that are linear combinations of its basic functions.

There are several ways of tackling system identification with GP. The most intuitive approach is to encode a candidate solution as a set of trees, each one representing an equation in SR style, and evaluate each solution by integrating the system it represents for the given initial conditions, to then compare the resulting trajectory against the available data, as proposed for example by [3, 15]. Such a framework, however, presents several limitations: (i) a candidate solution encoding several equations generates a search space which is extremely complex to explore, and cannot take advantage of more recent innovations introduced in modern SR algorithms, focused on learning a single regular equation; (ii) an evaluation function requiring the integration of a system can be considerably time-consuming, especially when compared to an evaluation of mean squared error performed by standard SR. For these reasons, SR for system identification often relies on data transformation techniques which turn the problem from learning the equations of an ODE system into learning several regular equations, one at a time. The most common data transformation approaches in literature are discussed more in detail in the following Subsection 2.3.

More recently, a Transformer-based neural network called ODEFormer [7, 8] has been presented. The network has been trained using trajectories coming from randomly-generated ODE systems, with and without added noise, in order to help it identify systems even under difficult conditions. Interestingly, the authors found that their method is indeed more performing in presence of noise with large amplitude, but GP-based SR with classical data transformations outperforms ODEFormer when noise is absent or low.

2.3 Data transformation approaches

Data transformation techniques for system identification proposed in literature can be tracked back to three main ideas.

2.3.1 Δx. Introduced in the seminal works on GP [18], this transformation aims to numerically approximate the value of the derivative $x'(t)$ at time t_n taking advantage of known measured trajectory points $x_i \approx x(t_i)$. In particular:

$$x'(t_n) \approx \frac{\Delta x}{\Delta t}(t_n) = \frac{x_n - x_{n-1}}{t_n - t_{n-1}} \tag{2}$$

Once the values for $\Delta x/\Delta t$ are computed for each point in the original trajectory, finding the analytical form of an equation $f(x,t)$ fitting these points will result directly in obtaining an approximation for $x'(t)$. In the following, we will refer to this data transformation approach as Δx.

2.3.2 F_x. This data transformation technique has been originally introduced by [13] as an alternative to Δx. Given Euler's forward method [12]:

$$x'(t_n) = \lim_{\Delta t \to 0} \frac{x(t_n + \Delta t) - x(t_n)}{\Delta t} \tag{3}$$

knowing measured trajectory points $x_n \approx x(t_n)$ and $x_{n+\Delta t} \approx x(t_n + \Delta t)$, and assuming that $x'(t) = f(x,t)$, we can write $x_{n+\Delta t} = x_n + \Delta t \cdot f(x_n, t_n)$. It is thus possible to define a function $F_x(\Delta t, t, x)$ such that

$$F_x(\Delta t, t_n, x_n) = x_{n+\Delta t} - x_n = \Delta t \cdot f(x_n, t_n) \tag{4}$$

Then, once $F(\Delta t, t, x)$ is known, the original equation for the derivative $x'(t) = f(x,t)$ can be deduced as follows:

$$x'(t_n) = \lim_{\Delta t \to 0} \frac{x(t_n + \Delta t) - x(t_n)}{\Delta t} \approx \lim_{\Delta t \to 0} \frac{x_{n+\Delta t} - x_n}{\Delta t} =$$
$$= \lim_{\Delta t \to 0} \frac{F_x(\Delta t, t_n, x_n) - F_x(0, t_n, x_n)}{\Delta t} = \left.\frac{\partial F_x(\Delta t, t_n, x_n)}{\partial \Delta t}\right|_{\Delta t = 0} \tag{5}$$

In other words, applying this data transformation has the objective of generating points for a regular equation F_x; once an analytical form of F_x is found, performing a symbolic partial differentiation $\partial F_x/\partial \Delta t$ and setting all remaining terms containing $\Delta t = 0$ returns the original equation for $x'(t)$. In the rest of the paper, we will refer to this approach as F_x.

2.3.3 C_x. An interesting alternative approach recently introduced by [25] is to forego entirely trying to approximate the derivative value. Given sample data from a trajectory $\mathbf{x}$ for variable x in the interval $t = [0, T]$, and an arbitrary support function $g(t)$ such that $g(0) = g(T) = 0$, the approach defines a new function $C(f, \mathbf{x}, g)$ with the property $C(f, \mathbf{x}, g) = 0 \iff f = f^*(\mathbf{x}, t) = x'(t)$. In particular:

$$C(f, \mathbf{x}, g) = \int_0^T f(\mathbf{x}) g(t) dt + \int_0^T x(t) g'(t) dt \tag{6}$$

However, this is not enough to define a proper search space: Two candidate equations $f_1 \neq f^*$ and $f_2 \neq f^*$ might have $C(f_1, \mathbf{x}, g) > C(f_2, \mathbf{x}, g)$, but this does not imply that f_1 is further away from the correct solution than f_2. Thus, [25] also proposes a proper fitness function based on Eq. 6. If we call a given candidate function f_1 and the correct solution f^*, it is possible to define $d_{\mathbf{x}}(f_1, f^*) = ||f_1 \circ \mathbf{x} - f^* \circ \mathbf{x}||_2 = ||(f_1 - f^*) \circ \mathbf{x}||_2$. This distance cannot be directly computed as it relies on the unknown optimal function f^*. Let's take instead a function F which can be computed directly from the data, such as:

$$F(f) = \sum_{i=1}^{N} \sum_{s=1}^{S} C(f, \hat{\mathbf{x}}_i, g_s)^2 \tag{7}$$

where $g_s(t)$ are a set of arbitrary functions with the same property $g_s(0) = g_s(T) = 0$, $\forall s$. If, additionally, the g_s functions are continuous, derivable, and are a Hilbert (orthonormal) basis for $L_2[0, T]$, it is then provable that:

$$\lim_{S \to \infty} \lim_{i \to \infty} \sum_{s=1}^{S} C(f, \hat{\mathbf{x}}_i, g_s)^2 = d_x(f, f^*)^2 \tag{8}$$

So that Eq. 7 can be used as a good approximation of the real fitness function, given a sufficiently large N, number of samples, and S, number of support functions $g_s(t)$. It is important to notice

that, differently from Δx and F_x, C_x requires significant modifications to the fitness function. The target values for the other two data transformations can be computed completely offline from trajectory data, and the fitness function can then just be set as the mean squared error (or other another regression metric) of the data predicted by a candidate function against the precomputed values. On the other hand, C_x necessitates further online steps each time a candidate solution is evaluated, namely numerically solving the integrals in Eq. 6 to then evaluate the fitness function in Eq. 7.

In this study we only consider data transformations Δx and F_x, which are more easily comparable, while a more complete evaluation of C_x is left to future works.

2.3.4 Hyperparameters. It is interesting to notice that data transformation approaches contain *hyperparameters*, settings decided by the user which impact performance. For Δx, the user can set an *order* of the transformation, which describes the amount of consecutive points in the time series to be used to build the approximation of the derivative $x'(t_n)$. Eq. 2 shows a transformation of order 1. A Δx transformation at a given instant x_n, t_n of order 2 would instead be:

$$\frac{\Delta x}{\Delta t}(t_n) = \frac{\frac{x_n - x_{n-1}}{t_n - t_{n-1}} + \frac{x_{n-1} - x_{n-2}}{t_{n-1} - t_{n-2}}}{2} \tag{9}$$

In the following, we will use the notation $\Delta x^{(k)}$ to indicate a Δx transformation of order k.

For F_x, the user can set a hyperparameter also called *order*, which this time impacts the number of points created by the transformation. A F_x transformation of order 1 only computes values for $F_x(0, t_n, x_n)$ and $F_x(t_{n+1} - t_n, t_n, x_n)$; while a F_x transformation of order 2 also computes $F_x(t_{n+2} - t_n, t_n, x_n)$; and so on. The notation $F_x^{(j)}$ will be used for a F_x transformation of order j.

While C_x is not evaluated in this work, it is worth mentioning that it also requires decisions on several hyperparameter values. The most important are the number S of support functions $g_s(t)$ and their exact specification, although the authors of [25] report good results with $S = [40, 60]$ and $g_s(t) = \sqrt{2/T} \cdot \sin(\pi s t/T)$, where T is the highest value for t appearing in the trajectory data; another possibility the authors evaluate for $g_s(t)$ is fitting cubic spline functions [9].

Finally, as the performance of all techniques is likely to be impacted by the presence of noise, smoothing can be applied to the original data before performing the transformations. A classic approach to smoothing is employing a Savitzky–Golay (SG) filter [28], which is a generalized form of moving average able to fit the data using polynomials of user-defined degree, on a window sliding over the data points. The hyperparameters of the SG filter are the size of the window and the degree of the polynomial that fits the data. Other approaches to denoising, ranging from Gaussian smoothing [23] to wavelet denoising [11], can also be applied for this task, but each one comes with its own set of hyperparameters to be selected. For the following experiments considering noise, we will employ a SG filter.

3 Proposed approach

While system identification approaches based on SR have been part of the scientific literature almost since the initial presentation of GP [13, 18], each work was only tested on a relatively small number of case studies, with rare comparisons of methods mostly based on the Strogatz benchmark [4], which however only included ODE systems with two state variables. To the best of our knowledge, an extensive testing of data transformation techniques for system identification using SR has never been presented. The recent introduction of an ODE benchmark suite specifically focused on system identification [7, 8] provides a unique opportunity to perform a thorough comparison of data transformation techniques for system identification using SR, attempting to identify their unique strengths and weaknesses.

The research questions we are interested in answering with the experimental evaluation are:

(1) Are data transformation hyperparameters impactful for their performance?
(2) How does noise affect the performance of the data transformations?
(3) Noise aside, can data transformations introduce errors? Can we quantify those errors?
(4) Are data transformations creating a search space for SR where the ground-truth solution actually represents the global optimum, or can they mislead SR?

3.1 The ODEBench benchmark suite

ODEBench [7, 8] is a Python benchmark suite for system identification, focused on ODE systems. It contains a total of 63 different combination of systems and parameters, with 23 one-variable systems, 28 two-variable systems, 10 three-variable systems, and 2 four-variable systems. For each system in the benchmark, two trajectories with different initial conditions are provided, for a total of 234 trajectories. The values for the trajectories are obtained through an integration based on the Livermore Solver for ODE (LSODE) method [14, 24] implemented in the `scipy` package [16], with parameters `t_span=(0,10)`, `rtol=1e-5`, `atol=1e-7`, `first_step=1e-6`, `min_step=1e-10`, `t_eval=np.linspace(0, 10, 150)`, which results in 150 values uniformly sampled from the trajectory. Thus, the original ODEBench data set includes 150 values for each state variable in the ODE system, plus the corresponding values for time, with two such trajectories for each system. However, it is also possible to use ODEBench to obtain trajectories with different rates of sampling or different initial conditions.

3.2 Evaluating performance

Comparing the performance of the data transformations is not straightforward. As this is basically a regression problem, the intuitive solutions would be employing metrics like mean squared error (MSE) or $R2 = 1 - (\sum(\hat{x}_i - x_i)^2)/(\sum(x_i - \bar{x})^2)$. Between the two, $R2$ is probably the easiest to interpret, as its numerical value does not depend on the scale of the original data. Still, while it is clear that values of $R2$ close to 1.0 are excellent and values close to 0.0 are indicative of poor performance, $R2$ values can also be strongly negative, which makes it challenging to compute means or medians over different systems. The same difficulties have been discussed by [7, 8], and we decided to employ the same proposed approach: Fixing an arbitrary threshold for *satisfactory performance*, e.g. $R2 > 0.9$, and then just counting the number of trajectories for

which the performance of a candidate solution can be considered satisfactory.

4 Experimental evaluation

All the code for the experiments is implemented in Python 3, and is freely accessible on an open GitHub repository[1]. Data related to systems and trajectories is taken from the ODEBench [7, 8] repository[2]. The functions implementing the Δx transformation and the denoising algorithm are from the `pysindy` module [10, 17]. The function implementing F_x is taken from [13] and its related repository.

With respect to the hyperparameters discussed in Subsection 2.3.4, we test values $k = \{1, 2, 3, 4, 5\}$ and $j = \{1, 2, 3, 4, 5\}$ for the order of $\Delta x^{(k)}$ and $F_x^{(j)}$; and values $w = \{5, 11, 15, 21, 25\}$ for the window size of the SG filter, fixing degree $d = 3$ for the polynomial interpolation, the default value used by the `pysindy` function. The threshold used to define a high performance is $R2 > 0.9$.

4.1 Experiment 1: Data transformation errors

The objective of the first experiment is to assess the extent of the errors possibly introduced by the data transformation techniques Δx and F_x, especially in presence of noise. While using the transformations on the trajectory data makes it possible to apply standard SR to the problem, it may also generate a search space which is difficult to explore. In order to evaluate this possibility, we compare the known ground truth equations for each system in ODEBench against the transformed data of the provided trajectory. Obtaining the ground truth equations for the transformations is straightforward, because if $x'(t) = f(x, t)$, the equation for Δx is exactly $f(x, t)$, while the equation for F_x is $F_x(\Delta t, x, t) = \Delta t \cdot f(x, t)$. Following [7, 8], we apply Gaussian multiplicative noise to the trajectory data, with intensity $\sigma = \{0.00, 0.01, 0.05\}$, to also evaluate the impact of noise.

The expectations are that (i) the performance of the ground truth equations should be close to $R2 = 1.0$, especially in absence of noise; (ii) applying the SG filter should improve the results when noise is present; (iii) different configurations of hyperparameters should show different performance. Interestingly, experimental results show that while (ii) and (iii) appear as expected, (i) is true for most systems and trajectories, but not all. Figure 2 shows the global performance of Δx and F_x for all combinations of hyperparameters.

Figure 3 presents the same comparison for a selection of the most performing combination of hyperparameters, for each data transformation, for each level of noise. From the figure, it becomes evident that Δx is the best approach in absence of noise. Once even a small amount of noise is added to the trajectories, the performance of the data transformation techniques degrades severely, even after the application of the SG filter. The F_x transformation seems to be able to capitalize more from the resulting smoothing, showing a slightly better overall performance for all levels of noise. Still, the amount of disruption suffered by the data transformation techniques is much higher than expected. As a term of comparison, when noise amplitude $\sigma = 0.01$, evaluating the $R2$ of the noisy trajectories versus their noiseless counterpart reveals that 231/234 (98.7%) trajectories show $R2 > 0.9$, versus the 72.2% performance of the best corresponding transformation, $F_x^{(1)}, w = 15$; when $\sigma = 0.05$, 212/234 (90.6%) trajectories present an $R2$ value above the threshold, versus the 37.6% of the best transformation $F_x^{(2)}, w = 25$.

Interestingly, though, even when noise is absent, the best-performing transformation $\Delta x^{(4)}$ does not reach $R2 > 0.9$ on all trajectories. Now, this is not necessarily a problem for SR: Even if the ground truth equation does not perform perfectly, if the search space resulting from the transformations is shaped in a way that the ground truth still represents the global optimum, SR algorithms should be able to find it. In the next set of experiments, however, we will show that this is not true in practice.

4.2 Experiment 2: Misleading optima

In this set of experiments, the aim is to test whether the ground truth equation can be rediscovered through the data transformation approaches, even when its performance is not considered acceptable according to our previous arbitrarily set threshold ($R2 \leq 0.9$). Note that these experiments are not meant to be thorough, but rather an attempt to find counterexamples where, in the new problem space defined by a data transformation, an equation different from the ground truth has a better performance, e.g. represents a stronger optimum than what should in theory be the best attainable result. Finding such a counterexample would mean that the search space created by data transformations is misleading for SR algorithms. We will focus on ODEBench systems for which the results of the data transformation techniques on the noiseless trajectory data all present $R2 \leq 0.9$ for at least one of the state variables in the system: This includes systems 55 (Lorenz equations in a complex periodic regime [20]), 59 (Rössler attractor, chaotic [27]), and 61 (Chen-Lee attractor [5]).

The SR tool selected for this task is `PySR` [6], a Python package implementing a modern multi-population complexity/fitting Pareto approach to SR. The hyperparameters set for `PySR` include the function set $\{+, -, *, /, \sin, \cos, \exp, \log, \sqrt{}\}$, population size $\mu_p = 50$, number of populations $P = 31$ (default value), maximum generations $G = 1,000$. Only one run is performed for each case study, as the objective is to find at least one candidate equation with better fitness value than the ground truth. `PySR` returns a Pareto front of candidate solutions, each one a compromise between MSE and complexity, defined as the number of nodes in the GP tree representation of the solution. We manually inspect the Pareto fronts of each run, searching for equations with better $R2$ value than the ground truth and equal or lower complexity, which would be Pareto-dominant with respect to the ground truth equation using this approach; we also report the highest-complexity equation in the front, with the global best $R2$, as a reference. The complete results of this experimental run are reported in Table 1.

From the results in the table, it is noticeable how very often the data transformations create a search space with misleading optima. Such misleading optima can be roughly divided into two categories: Equations with the same structure as the ground truth, but different numerical parameter values; and equations with completely different structures. Taking for instance system 55, trajectory 1, for Δx `PySR` found the expression $x_0'(t) = -8.86x_0 + 8.86x_1$ ($R2 = 0.7158$) instead of the ground truth $x_0'(t) = -10.0x_0 + 10.0x_1$

[1] https://github.com/albertotonda/symbolic-regression-ode-systems
[2] ODEBench, https://github.com/sdascoli/odeformer/tree/main/odeformer/odebench

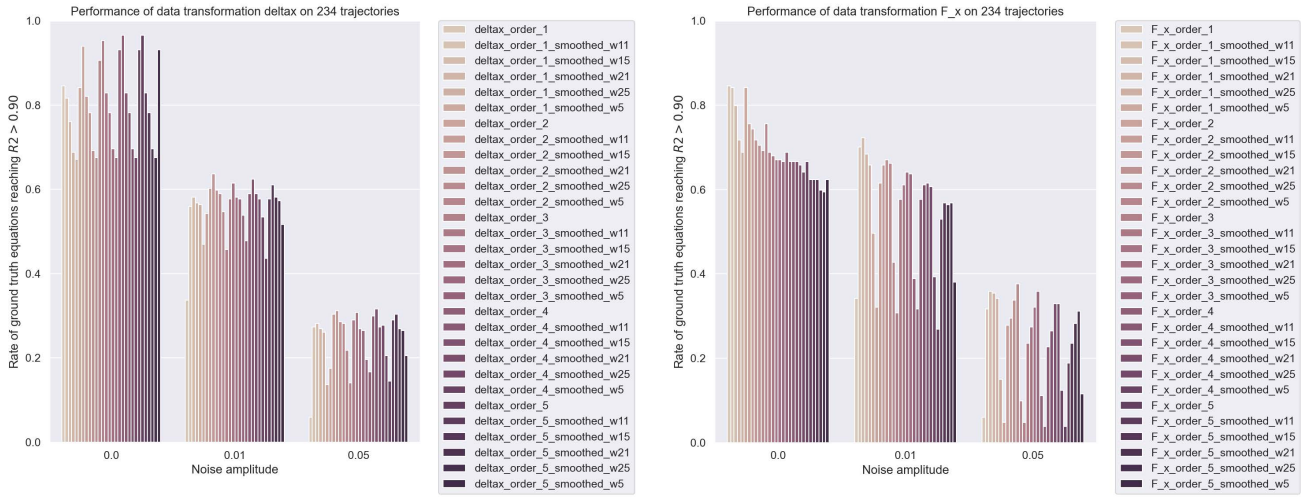


Figure 2: Performance of Δx (left) and F_x (right) considering all combinations of hyperparameter values. It is noticeable how the performance of all transformations degrades significantly when noise is added to the trajectory data.

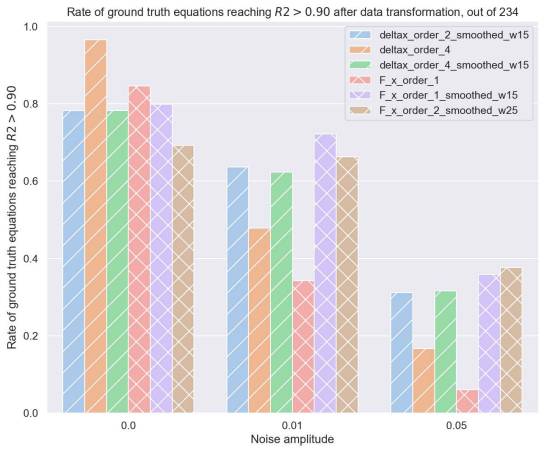


Figure 3: Best-performing combinations of hyperparameters for transformations Δx (stripes) and F_x (crosses). Δx performs the best in absence of noise, while F_x seems to be able to take the most advantage out of the denoising performed by the SG filter.

($R2 = 0.7039$). In this case, only the values of the numerical parameters are incorrect, and while ODE systems can be very sensitive to parameter values, if all equations found for a target ODE system presented the correct structure, the correct values of the parameters could be found by performing numerical optimization with respect to the trajectories in the original search space. However, considering again system 55, trajectory 1, the equation found using Δx for $x_2'(t) = \frac{x_1}{(t-0.00213)^{1.0}}$ ($R2 = 0.5221$) has a completely different structure from the ground truth $x_2'(t) = x_0 x_1 - 2.67 x_2$ ($R2 = 0.2345$)!

A different example of the same behavior for transformation F_x can be found for system 61, trajectory 1; the equation found $F_{x_0} = \Delta_t x_1 (-1.0 x_2 - 3.85) = \Delta_t (3.85 x_1 - x_1 x_2)$, ($R2 = 0.8687$), shares the same structure of the ground truth $F_{x_0} = \Delta_t (5.0 x_0 - x_1 x_2)$, ($R2 = 0.7793$), albeit with a different value of a numerical parameter. On the other hand, again for system 61, trajectory 1, the equation found for variable $F_{x_2} = \Delta_t \cos(0.107 x_0)$, ($R2 = 0.6461$), is radically different from the ground truth $F_{x_2} = \Delta_t (0.333 x_0 x_1 - 3.8 x_2)$, ($R2 = 0.5235$). Regarding the solutions found for transformation F_x, it is worth noting that it is still possible to further remove unsatisfying candidate solutions from the Pareto front by computing the symbolic derivation $\partial F_x / \partial \Delta_t$ and eliminating all equations that reduce to a constant, as proposed in [13], but in this case we did not perform this step, as we are more interested in evaluating the search space defined by F_x rather than computational efficiency.

A general trend observable from Table 1 is that when the $R2$ value of the ground truth equation is close to 1.0, SR is often able to find equations with the correct structure; but when the $R2$ of the ground truth in the data transformation space is far from an optimal value, SR converges on undesirable candidate solutions with completely different structures. It is also interesting to remark how, for each experimental configuration, `PySR` is almost always able to discover a higher-complexity equation with a better fitting, a finding which reinforces the validity of Pareto-based approaches to SR, at least for system identification, to avoid potential overfitting.

4.3 Experiment 3: Impact of the sampling step

One possible reason for the unexpected performance issues of the transformations on noiseless data might be related to the rate of change of the value of the derivative. Euler's forward method, at the basis of the F_x transformation, is well known for generating large errors in cases where the value of the derivative changes quickly over

Table 1: Results of the SR experiments using PySR on each system, trajectory, data transformation and state variable. From the Pareto front resulting from each experiment, the equation with complexity lower or equal to the ground truth and equal or better $R2$ value is reported, along with the highest-complexity equation, as a reference. $R2$ values in bold highlight the best result between the two equations with similar complexity. A $*$ near the value indicates a candidate solution with significantly different structure with respect to the ground truth. For system 61, trajectory 1, Δx_2, the algorithm was unable to find an equation with the desired characteristics.

System id	Trajectory	Variable	Ground truth equation		Eq. similar complexity		Highest-complexity eq. with best $R2$	
			Form	$R2$	Form	$R2$	Form	$R2$
55	1	Δx_0	$-10.0x_0 + 10.0x_1$	0.7039	$-8.86x_0 + 8.86x_1$	**0.7158**	$(54.4 - 9.83\log(x_2))\left(-1.17x_0 + x_1 + 1.0 - 9.1 \cdot 10^6 e^{-1.0x_2}\right)$	0.9824
		Δx_1	$-x_0x_2 + 100.0x_0 - x_1$	0.3922	$x_0 \cdot (101.0 - 1.0x_2)$	**0.4042***	$3.91t + 3.91x_1 + \frac{63.8\left(-1.0x_0 + \frac{107.0}{x_2^{1.0}}\right)(x_2 - 81.8)}{x_2^{1.0}}$	0.9085
		Δx_2	$x_0x_1 - 2.67x_2$	0.2345	$\frac{x_1}{(t-0.00213)^{1.0}}$	**0.5221***	$-1.0x_2 + \frac{57.9(x_0x_1 - 152.0)}{(-1.0t+x_2-10.4)^{1.0}} + 57.9e^{\cos\left(x_2^{0.5}\right)}$	0.9618
		F_{x_0}	$\Delta_t\left(-10.0x_0 + 10.0x_1\right)$	0.4926	$7.17\Delta_t\left(-1.0x_0 + x_1\right)$	**0.5832**	$\left(-1.0x_0 \cdot (0.0703x_2 - 5.73) + x_1\right)\sin\left(\Delta_t^{0.5}\right) + 2.98\sin\left(\Delta_t x_1\right)$	0.9580
		F_{x_1}	$\Delta_t\left(-x_0x_2 + 100.0x_0 - x_1\right)$	0.3694	$x_0\cos\left(0.0906x_0\right)$	**0.3740***	$\Delta_t\left(1.32x_0 + x_1\right)\left(-0.257x_2 + 6.49\cos\left(0.059x_1\right) + 22.9\right)$	0.9297
		F_{x_2}	$\Delta_t\left(x_0x_1 - 2.67x_2\right)$	-0.3324	$\cos\left(x_1\right)$	**0.0044***	$\frac{3.63\cdot10^3\sin\left(0.0378\Delta_t x_0\left(-1.0x_0+x_1\right)-1.0\Delta_t\right)}{\left(-2.0t+x_2\right)^{1.0}}$	0.8826
	2	Δx_0	$-10.0x_0 + 10.0x_1$	0.9349	$-8.81x_0 + 8.81x_1$	**0.9523**	$-\frac{1.32t}{(x_0-27.4)^{1.0}} + 4.15\left(-1.0x_0 + x_1\right)e^{\cos\left(0.0417x_0\right)}$	0.9947
		Δx_1	$-x_0x_2 + 100.0x_0 - x_1$	0.4488	$\frac{x_0}{(0.00233-1.0t)^{1.0}}$	**0.5189***	$\frac{0.973x_0}{(0.00233-1.0\sin(t(t+0.366)))^{1.0}} - 17.1x_0 + 8.57x_1$	0.8745
		Δx_2	$x_0x_1 - 2.67x_2$	0.6486	$x_0x_1 - 249.0$	**0.6754***	$\left(17.1x_0 + 17.1x_1\right)\sin\left(\frac{\frac{t}{x_0^{1.0}}+2.0x_0}{(x_2-43.7)^{1.0}}\right) - 293.0$	0.9572
		F_{x_0}	$\Delta_t\left(-10.0x_0 + 10.0x_1\right)$	0.6139	$7.59\Delta_t\left(-1.0x_0 + x_1\right)$	**0.6827**	$\frac{585.0\Delta_t\left(-0.0341x_0(x_2-54.0)+x_1\right)}{\left(x_2+e^{\Delta_t x_1}\right)^{1.0}}$	0.9582
		F_{x_1}	$\Delta_t\left(-x_0x_2 + 100.0x_0 - x_1\right)$	0.0305	$-0.41x_0$	**0.1740***	$\left(\Delta_t + \sin\left(\cos\left(\Delta_t x_0\right) - 1.0\cos\left(\frac{1.19\cdot10^3\Delta_t}{x_2^{1.0}}\right)\right)\right)\left(x_0 + x_1 + 3.99\right)$	0.9350
		F_{x_2}	$\Delta_t\left(x_0x_1 - 2.67x_2\right)$	0.3343	$x_1\sin\left(\Delta_t x_0\right)$	**0.3946***	$81.7\sin\left(\sin\left(\sin\left(\frac{160.0\Delta_t x_0\left(-1.0x_0+x_1\right)}{x_2^{2.0}}\right) - 1.0\Delta_t\right)\right)$	0.9507
59	1	Δx_0	$-5.0x_1 - 5.0x_2$	0.9945	$-4.93x_1 - 4.93x_2$	**0.9947**	$-4.99x_1 + 1.3x_2\cos\left(1.29\left(e^{x_2}\right)^{0.5}\right) - 5.36x_2$	0.9996
		Δx_1	$5.0x_0 + 1.0x_1$	**0.9998**	$4.99x_0 + x_1$	**0.9998**	$5.0x_0 + x_1 - 1.0x_2\sin\left(0.159\cos\left(\frac{x_1}{(x_0-0.397)^{1.0}}\right) - 0.0664\right)$	1.0000
		Δx_2	$5.0x_2\left(x_0 - 5.7\right) + 1.0$	0.6627	$3.53x_2\left(x_0 - 6.06\right)$	**0.8299***	$x_2 + \left(x_0 - 5.86\right)\left(\sin\left(\cos\left(x_2^{0.5}\right)\right) - 0.834\right)\left(t - 1.0\cos\left(e^{x_1}\right) - 27.2\right)$	0.9929
		F_{x_0}	$\Delta_t\left(-5.0x_1 - 5.0x_2\right)$	0.9267	$-4.76\Delta_t\left(x_1 + x_2\right)$	**0.9291**	$5.02\Delta_t\left(-0.495x_0\left(x_2 + 0.583\right)^{0.5} + 0.227x_0 - 1.0x_1 - 0.227x_2\right)$	0.9963
		F_{x_1}	$\Delta_t\left(5.0x_0 + 1.0x_1\right)$	0.9622	$4.87\Delta_t x_0$	**0.9916***	$\Delta_t\left(5.04x_0 + 0.161x_1\left(x_2 + \cos\left(\frac{x_2}{\cos^{1.0}(x_2)}\right)\right) - 1.0x_2\right)$	0.9998
		F_{x_2}	$\Delta_t\left(5.0x_2\left(x_0 - 5.7\right) + 1.0\right)$	-0.2643	$-0.181x_2$	**0.0908***	$\Delta_t x_2\left(-0.728x_0\left(x_1 - 4.03\right) - 1.0x_2 - 1.0\sin\left(0.952x_2\right) - 3.62\right)$	0.9943
	2	Δx_0	$-5.0x_1 - 5.0x_2$	0.9967	$-4.95x_1 - 4.95x_2$	**0.9968**	$-5.01x_1 - 5.01x_2 - 1.0\left(x_2 - 1.0\sin\left(x_2\right)\right)\sin\left(\cos\left(0.559x_0\right) + 0.0881\right)$	0.9997
		Δx_1	$5.0x_0 + 1.0x_1$	**0.9999**	$4.99x_0 + x_1$	**0.9999**	$5.0x_0 + x_1 - 0.0602 \cdot (0.736 - 2.56x_2)\cos\left(0.286x_0\right) + 0.014$	1.0000
		Δx_2	$5.0x_2\left(x_0 - 5.7\right) + 1.0$	0.8720	$x_2 \cdot (4.1x_0 - 23.4)$	**0.9149***	$\frac{\cos(t)}{x_0^{1.0}} + x_2\left(x_0 - 5.89\right)\left(-0.259x_1 - 0.259x_2 + 6.61\right) + 0.946$	0.9876
		F_{x_0}	$\Delta_t\left(-5.0x_1 - 5.0x_2\right)$	0.9346	$\Delta_t\left(-4.79x_1 - 4.79x_2\right)$	**0.9363**	$\Delta_t\left(-1.0x_0 - 5.1x_1 - 1.0x_2\left(e^{0.202x_0} + 0.576\right) - 0.526\right)$	0.9987
		F_{x_1}	$\Delta_t\left(5.0x_0 + 1.0x_1\right)$	0.9653	$4.89\Delta_t x_0$	**0.9944***	$\frac{\Delta_t\left(-167.0x_0 - 6.11x_1 + 6.11x_2\right)}{\left(-0.946x_2 - 32.9\right)^{1.0}}$	0.9999
		F_{x_2}	$\Delta_t\left(5.0x_2\left(x_0 - 5.7\right) + 1.0\right)$	0.3433	$0.00324\Delta_t e^{x_0}$	**0.4109***	$\Delta_t\left(x_2\left(e^{0.354x_0+0.354\cos\left(0.217x_2\right)} - 14.9\right) + 1.2\right)$	0.9829
61	1	Δx_0	$5.0x_0 - x_1x_2$	**0.9861**	$4.95x_0 - 1.0x_1x_2$	**0.9861**	$5.09x_0 + 0.985x_1\left(-1.0x_2 - 1.27\sin\left(\frac{3.1x_1}{x_0^{1.0}} + 0.575\right)\right)$	0.9926
		Δx_1	$x_0x_2 - 10.0x_1$	**0.9159**	$x_0x_2 - 9.97x_1$	**0.9159**	$-1.11x_1 + 1.11\left(x_0x_2 - 8.6x_1\right)\cos\left(0.0409x_1 - 0.0409\cos\left(x_1\right)\right)$	0.9894
		Δx_2	$0.333x_0x_1 - 3.8x_2$	**0.8217**	-	-	$-3.74x_2 + 2.39\left(-2.5x_0 + x_1\right)\sin\left(\frac{x_1}{(x_2-10.4)^{1.0}}\right) + 5.18$	0.9697
		F_{x_0}	$\Delta_t\left(5.0x_0 - x_1x_2\right)$	0.7535	$\Delta_t x_1\left(-1.0x_2 - 3.85\right)$	**0.8455**	$0.718\Delta_t\left(-1.0x_0 + x_1\right)\left(-1.0x_2 - 0.823\right) + 10.0\sin\left(0.89\Delta_t x_0\right)$	0.9875
		F_{x_1}	$\Delta_t\left(x_0x_2 - 10.0x_1\right)$	0.6561	$\Delta_t x_2\left(x_0 + x_1\right)$	**0.8110***	$\left(x_0 + 1.12x_1\right)\sin\left(\sin\left(\Delta_t^{0.5} \cdot (0.288x_2 + \sin\left(\cos\left(\Delta_t x_0\right)\right)\right)\right)\right)$	0.9863
		F_{x_2}	$\Delta_t\left(0.333x_0x_1 - 3.8x_2\right)$	0.4147	$48.7\Delta_t\cos\left(0.109x_0\right)$	**0.6007***	$129.0\Delta_t\left(-1.0\sin\left(\cos\left(\Delta_t x_2\right)\right) + \cos\left(\frac{x_0}{(-1.0\cos(0.213x_1)-18.3)^{1.0}}\right)\right)$	0.9581
	2	Δx_0	$5.0x_0 - x_1x_2$	**0.9787**	$5.0x_0 - 1.0x_1x_2$	**0.9787**	$5.26x_0 + 5.58x_1\left(-0.574x_2 + 0.574\sin\left(4.23t\right) - 1.0\right)^{0.5} - 1.0x_1$	0.9916
		Δx_1	$x_0x_2 - 10.0x_1$	**0.9679**	$x_0x_2 - 10.0x_1$	**0.9679**	$-1.0x_1 + \left(x_0x_2 - 1.0x_0 - 10.1x_1\right)\cos\left(\cos\left(0.146x_2 - 0.132\right)\right)$	0.9961
		Δx_2	$0.333x_0x_1 - 3.8x_2$	0.9387	$0.292x_0x_1 - 3.34x_2$	**0.9582**	$\left(0.37x_0x_1 - 4.24x_2\right)\sin\left(0.608\left(-x_2\right)^{0.5}\right) + \cos\left(0.508x_2\right)$	0.9963
		F_{x_0}	$\Delta_t\left(5.0x_0 - x_1x_2\right)$	0.7793	$\Delta_t x_1\left(-1.0x_2 - 3.65\right)$	**0.8687**	$\Delta_t x_1 \cdot \left(\frac{156.0}{x_2^{1.0}} - 2.82\cos\left(0.0744\Delta_t x_0 x_1\right) + 23.0\right)$	0.9643
		F_{x_1}	$\Delta_t\left(x_0x_2 - 10.0x_1\right)$	0.7248	$\Delta_t x_2\left(x_0 + x_1\right)$	**0.8746***	$\left(x_0 + 1.05x_1\right)\sin\left(\sin\left(\Delta_t\left(x_2 - 0.201\right)\right)\right) + \sin\left(2.0\Delta_t x_0\right)$	0.9911
		F_{x_2}	$\Delta_t\left(0.333x_0x_1 - 3.8x_2\right)$	0.5235	$52.8\Delta_t\cos\left(0.107x_0\right)$	**0.6461***	$\Delta_t^{0.5} + \frac{\left(x_0+x_1\right)\sin\left(\Delta_t x_1\right)}{(1.65\cos\left(\Delta_t x_0\right)+2.16)^{1.0}}$	0.9488

time with respect to the step size Δt; and in practice, it has in fact long since being replaced by more modern integration approaches, such as the family of Runge-Kutta methods or LSODE [24]. If this is true, reducing Δt should improve the performance of the data transformations.

In order to test this idea, we perform a set of experiments with setup identical to Experiment 1, testing the ground truth equations derived from the data transformations Δx and F_x against the transformed data on noiseless trajectories. This time, however, we use ODEBench to generate three versions of the same trajectory, with different sampling steps $\Delta t = 0.06$ (original settings, resulting in 150 data points per variable in the trajectory), $\Delta t = 0.03$ (300 points) and $\Delta t = 0.015$ (600 points).

Focusing again on systems 55, 59, and 61, results for the two best configurations of hyperparameters on noiseless data $\Delta x^{(4)}$ and $F_x^{(1)}$ are shown in Table 2, with cells color-coded, dividing results into near perfection ($R2 > 0.99$), small errors ($0.99 \leq R2 < 0.9$), considerable errors ($0.9 \leq R2 < 0.5$) and large errors ($R2 \leq 0.5$). As is noticeable, reducing the step size indeed improves the results for all transformations on all trajectories.

Table 2: Performance of the transformation approaches for decreasing values of Δt on noise-free trajectory data. Both transformations considerably improve their performance as the sampling step reduces.

System id	Trajectory	Variable	$\Delta x^{(4)}$			$F_x^{(1)}$		
			$\Delta t = 0.06$	$\Delta t = 0.03$	$\Delta t = 0.015$	$\Delta t = 0.06$	$\Delta t = 0.03$	$\Delta t = 0.015$
55	1	x_0	0.7039	0.9963	1.0000	0.4926	0.8794	0.9695
		x_1	0.3922	0.9838	0.9997	0.3694	0.7337	0.9286
		x_2	0.2345	0.9701	0.9997	-0.3324	0.7211	0.9302
	2	x_0	0.9349	0.9965	1.0000	0.6139	0.8939	0.9732
		x_1	0.4488	0.9591	0.9997	0.0305	0.7798	0.9417
		x_2	0.6486	0.9773	0.9999	0.3343	0.7784	0.9448
59	1	x_0	0.9945	0.9999	1.0000	0.9267	0.9780	0.9944
		x_1	0.9998	1.0000	1.0000	0.9622	0.9906	0.9977
		x_2	0.6627	0.9935	0.9999	-0.2643	0.7863	0.9455
	2	x_0	0.9967	1.0000	1.0000	0.9346	0.9830	0.9957
		x_1	0.9999	1.0000	1.0000	0.9653	0.9913	0.9978
		x_2	0.8720	0.9965	1.0000	0.3433	0.8263	0.9562
61	1	x_0	0.9861	0.9991	1.0000	0.7535	0.9327	0.9828
		x_1	0.9159	0.9983	1.0000	0.6561	0.9124	0.9778
		x_2	0.8217	0.9971	1.0000	0.4147	0.8564	0.9638
	2	x_0	0.9787	0.9996	1.0000	0.7793	0.9417	0.9853
		x_1	0.9679	0.9994	1.0000	0.7248	0.9279	0.9817
		x_2	0.9387	0.9990	1.0000	0.5235	0.8792	0.9696

5 Conclusions and future works

This work presented a first thorough comparison of data transformation techniques for applying standard symbolic regression to system identification, employing a recently published benchmark suite of systems of ordinary differential equations. The data transformations tested all aim to transform the problem from learning a differential equation to learning a regular equation, from which the original differential equation can later be obtained.

The experimental evaluation shows a predictable negative impact of noise on the quality of the results of data transformations, but also, perhaps surprisingly, the generation of a misleading search space even in absence of noise. In such search spaces, the ground truth equations, which are supposed to be the global optima, actually have worse fitness values than other candidate equations; and we show through an experimental run that a state-of-the-art SR algorithm is indeed misled by the search space. Further experiments show how the sampling step Δt seems to have a strong correlation with this phenomenon, and how reducing Δt helps mitigating the issue.

From the experimental results, it is possible to provide general recommendations for practitioners willing to use this approach to system identification: If absence of noise on data is guaranteed, the $\Delta x^{(4)}$ data transformation usually delivers the best results; in presence of noise, $F_x^{(1)}$ with SG smoothing (degree of polynomial regression $d = 3$, window size $w = 15$) or $F_x^{(2)}$ with SG smoothing ($d = 3, w = 25$) tend to perform better.

The issues highlighted by the experiments also point novel directions for future research. First of all, as noise obviously has a strong impact on the quality of the results, testing other denoising/smoothing techniques could be beneficial. Even in absence of noise, however, it could be extremely interesting to anticipate when the techniques will generate misleading search spaces starting from the characteristics of the trajectory data; a possible lead could be finding a good approximation for the Lyapunov stability [22]. As the sampling step Δt is influential, performing data imputation to artificially obtain a smaller step could be promising, as preliminary results seem to indicate [1]; however, data imputation relies on a considerable number of assumptions, which could lead to negative results, if they are not generally true for specific practical applications.

With the availability of the ODEBench benchmark suite, it will also be possible to perform an even more in-depth analysis. Future works will include assessing the impact of randomly dropping a given rate of data points from the trajectories; assessing the impact of using more than one trajectory as training data; testing more alternatives for smoothing; and finding a fair way of introducing data transformation C_x in the comparison.

References

[1] Authors. 2025. Omitted for double blind review. In *Omitted for double blind review.* Omitted for double blind review.

[2] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences* 113, 15 (March 2016), 3932–3937. https://doi.org/10.1073/pnas.1517384113

[3] Hongqing Cao, Lishan Kang, Yuping Chen, and Jingxian Yu. 2000. Evolutionary modeling of systems of ordinary differential equations with genetic programming. *Genetic Programming and Evolvable Machines* 1, 4 (2000), 309–337.

[4] William G. La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H. Moore. 2021. Contemporary Symbolic Regression Methods and their Relative Performance. *CoRR* abs/2107.14351 (2021), 1–28. arXiv:2107.14351 https://arxiv.org/abs/2107.14351

[5] Hsien-Keng Chen and Ching-I Lee. 2004. Anti-control of chaos in rigid body motion. *Chaos, Solitons & Fractals* 21, 4 (2004), 957–965. https://doi.org/10.1016/j.chaos.2003.12.034

[6] Miles Cranmer. 2023. Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl. https://doi.org/10.48550/ARXIV.2305.01582

[7] Stéphane d'Ascoli, Sören Becker, Alexander Mathis, Philippe Schwaller, and Niki Kilbertus. 2023. ODEFormer: Symbolic Regression of Dynamical Systems with Transformers. arXiv:2310.05573 [cs.LG] https://arxiv.org/abs/2310.05573

[8] Stéphane d'Ascoli, Sören Becker, Philippe Schwaller, Alexander Mathis, and Niki Kilbertus. 2024. ODEFormer: Symbolic Regression of Dynamical Systems with Transformers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, International Conference on Machine Learning, San Diego, CA, 1–28. https://openreview.net/forum?id=TzoHLiGVMo

[9] Carl De Boor. 1978. *A practical guide to splines* (1 ed.). Springer, New York, NY.

[10] Brian de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Kutz, and Steven Brunton. 2020. PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software* 5, 49 (2020), 2104. https://doi.org/10.21105/joss.02104

[11] David L Donoho and Iain M Johnstone. 1994. Ideal spatial adaptation by wavelet shrinkage. *biometrika* 81, 3 (1994), 425–455.

[12] Leonhard Euler. 1768. *Institutionum calculi integralis.* Vol. 1. imp. Acad. imp. Saènt., Saint Petersburg, Russia.

[13] Sébastien Gaucel, Maarten Keijzer, Evelyne Lutton, and Alberto Tonda. 2014. Learning Dynamical Systems Using Standard Symbolic Regression. In *Genetic Programming*, Miguel Nicolau, Krzysztof Krawiec, Malcolm I. Heywood, Mauro Castelli, Pablo García-Sánchez, Juan J. Merelo, Victor M. Rivas Santos, and Kevin Sim (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 25–36.

[14] Alan C Hindmarsh. 1983. ODEPACK, a systemized collection of ODE solvers. *Scientific computing* 1 (1983), 55–64.

[15] Hitoshi Iba. 2008. Inference of differential equation models by genetic programming. *Information Sciences* 178, 23 (2008), 4453–4468.

[16] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001–. SciPy: Open source scientific tools for Python. http://www.scipy.org/

[17] Alan A. Kaptanoglu, Brian M. de Silva, Urban Fasel, Kadierdan Kaheman, Andy J. Goldschmidt, Jared Callaham, Charles B. Delahunt, Zachary G. Nicolaou, Kathleen Champion, Jean-Christophe Loiseau, J. Nathan Kutz, and Steven L. Brunton. 2022. PySINDy: A comprehensive Python package for robust sparse system identification. *Journal of Open Source Software* 7, 69 (2022), 3994. https://doi.org/10.21105/joss.03994

[18] John R. Koza. 1992. *Genetic Programming. 1: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, Mass.

[19] W. F. Langford. 1984. Numerical Studies of Torus Bifurcations. In *Numerical Methods for Bifurcation Problems: Proceedings of the Conference at the University of Dortmund, August 22–26, 1983*, T. Küpper, H. D. Mittelmann, and H. Weber (Eds.). Birkhäuser Basel, Basel, 285–295. https://doi.org/10.1007/978-3-0348-6256-1_19

[20] Edward N. Lorenz. 1963. Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences* 20, 2 (March 1963), 130–141. https://doi.org/10.1175/1520-0469(1963)020<0130:dnf>2.0.co;2

[21] Alfred J. K. Lotka. 1925. *Elements of Physical Biology.* Williams and Wilkins Company.

[22] A. M. Lyapunov. 1992. The general problem of the stability of motion. *Internat. J. Control* 55, 3 (March 1992), 531–534. https://doi.org/10.1080/00207179208934253

[23] David Marr and Ellen Hildreth. 1980. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences* 207, 1167 (1980), 187–217.

[24] Linda Petzold. 1983. Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations. *SIAM J. Sci. Statist. Comput.* 4, 1 (1983), 136–148. https://doi.org/10.1137/0904010

[25] Zhaozhi Qian, Krzysztof Kacprzyk, and Mihaela van der Schaar. 2022. D-CODE: Discovering Closed-form ODEs from Observed Trajectories. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022.* OpenReview.net. https://openreview.net/forum?id=wENMvIsxNN

[26] Yulia Rubanova, Tian Qi Chen, and David Duvenaud. 2019. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 5321–5331. https://proceedings.neurips.cc/paper/2019/hash/42a6845a557bef704ad8ac9cb4461d43-Abstract.html

[27] O.E. Rössler. 1976. An equation for continuous chaos. *Physics Letters A* 57, 5 (1976), 397–398. https://doi.org/10.1016/0375-9601(76)90101-8

[28] Abraham Savitzky and M. J. E. Golay. 1964. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry* 36, 8 (July 1964), 1627–1639. https://doi.org/10.1021/ac60214a047

[29] Michael Schmidt and Hod Lipson. 2009. Distilling Free-Form Natural Laws from Experimental Data. *Science* 324, 5923 (April 2009), 81–85. https://doi.org/10.1126/science.1165893