POSTER

Comparing Data Transformation Techniques for System Identification With Standard Symbolic Regression

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/10.1145/3712255.3726673
ISBN: 9798400714641

Comparing Data Transformation Techniques for System Identification With Standard Symbolic Regression

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, Qi Chen,
Bing Xue, Mengjie Zhang
{name.surname}@ecs.vuw.ac.nz
Victoria University of 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. Applying standard symbolic regression, however, requires transforming trajectory data to frame the problem as learning a set of regular equations. This study presents a first thorough 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.

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

ACM Reference Format:

Alberto Tonda, Hengzhe Zhang, Qi Chen,, Bing Xue, Mengjie Zhang, and Evelyne Lutton. 2025. Comparing Data Transformation Techniques for System Identification With Standard Symbolic Regression. In *Genetic and Evolutionary Computation Conference (GECCO '25 Companion), July 14–18, 2025, Malaga, Spain.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3712255.3726673

1 Introduction

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 [18]. 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 [15].

Solutions based on Genetic Programming (GP) [13] 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 [1, 10], but since most of the research effort on SR targets regular equations, it is not straightforward to adapt the latest developments of SR [3, 21] 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 [8, 13, 17], 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. 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 dedicated to system identification did not exist, until recently. The publication of the ODEBench system identification benchmark suite [4, 5], focused on systems of Ordinary Differential Equations (ODEs), finally offers the opportunity of performing a fair and thorough comparison.

In this work, we 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.

2 Background

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. A generic ODE system can thus be written as (Euler's notation):

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

ACM ISBN 979-8-4007-1464-1/2025/07
https://doi.org/10.1145/3712255.3726673

$\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t), t)$ 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. 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 *trajectory*, i.e. the value of $\mathbf{x}$ for all $\mathbf{t}$ in $[t_0, T]$.

The goal of system identification is to automatically obtain a predictive model describing a dynamical phenomenon, directly from data points from one or more trajectories. A desirable output of system identification would be a set of human-interpretable equations. Solutions based on Symbolic Regression (SR) have been proposed in literature. They often relies on data transformation techniques which turn the problem from learning the equations of an ODE system into learning several regular equations.

Recently, a Transformer-based neural network, ODEFormer [4, 5], has been trained using trajectories coming from randomly-generated ODE systems, with and without added noise. 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.

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

(1) Δx. Introduced in the seminal works on GP [13], this transformation aims to numerically approximate $x'(t)$ at time t_n taking advantage of known measured trajectory points $x_i \approx x(t_i)$.

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

Finding the analytical form of an equation $f(x, t)$ fitting these points provides an approximation for $x'(t)$.

(2) F_x. This data transformation technique has been originally introduced by [8]. Given Euler's forward method [7]:

$$x'(t_n) = \lim_{\Delta t \to 0} \frac{x(t_n + \Delta t) - x(t_n)}{\Delta t} \tag{2}$$

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 $F_x(\Delta t, t, x) = x_{n+\Delta t} - x_n = \Delta t \cdot f(x_n, t_n)$. The original equation $x'(t) = f(x, t)$ then becomes:

$$x'(t_n) = \lim_{\Delta t \to 0} \frac{x(t_n + \Delta t) - x(t_n)}{\Delta t} = \left.\frac{\partial F_x(\Delta t, t_n, x_n)}{\partial \Delta t}\right|_{\Delta t = 0} \tag{3}$$

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 $x'(t)$.

Both data transformations depend on hyperparameters, 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. 1 shows a transformation of order 1. 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 the *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.

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. For the following experiments, we employ a classic approach to smoothing, a Savitzky–Golay (SG) filter [20], 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.

3 Proposed approach

ODEBench [4, 5] 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 [9, 16] implemented in the `scipy` package [11], 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.

Comparing the performance of the data transformations is not straightforward. We adopt an approach proposed in [4, 5]: Fix an arbitrary threshold for *satisfactory performance* based on $R2 = 1 - (\sum(\hat{x}_i - x_i)^2)/(\sum(x_i - \bar{x})^2)$, for example $R2 > 0.9$, and then count 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 are taken from the ODEBench [4, 5] repository[2]. The functions implementing the Δx transformation and the denoising algorithm are from the `pysindy` module [6, 12]. The function implementing F_x is taken from [8] and its related repository.

With respect to the hyperparameters discussed above, 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 $R2 > 0.9$.

Experiment 1: Data transformation errors. We compare the known ground-truth equations for each system in ODEBench against the transformed data of the provided trajectory. Obtaining

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

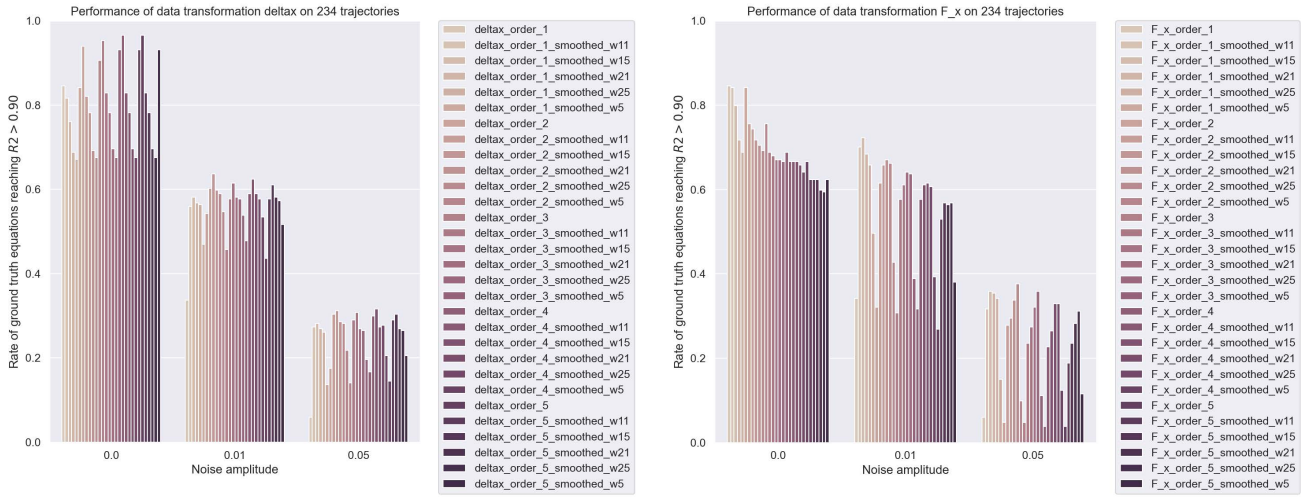

Figure 1: Performance of Δx (left) and F_x (right) considering all combinations of hyperparameter values. It is noticeable how the performance of all transformations degrades significantly when noise is added to the trajectory data.

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 [4, 5], 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 the absence of noise; (ii) the application of the SG filter should improve the results when noise is present; (iii) different configurations of hyperparameters should show different performances. Experimental results show that while (ii) and (iii) appear as expected, (i) is true for most systems and trajectories, but not all. Figure 1 shows the global performance of Δx and F_x for all combinations of hyperparameters.

Figure 2 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.

Interestingly, though, even in absence of noise, the best-performing transformation $\Delta x^{(4)}$ does not reach $R2 > 0.9$ on all trajectories.

Experiment 2: Misleading optima. The aim is to test whether the ground truth equation can be rediscovered through data transformation approaches, even when its performance is not considered acceptable according to the threshold ($R2 \leq 0.9$). Note that these experiments are 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. Finding

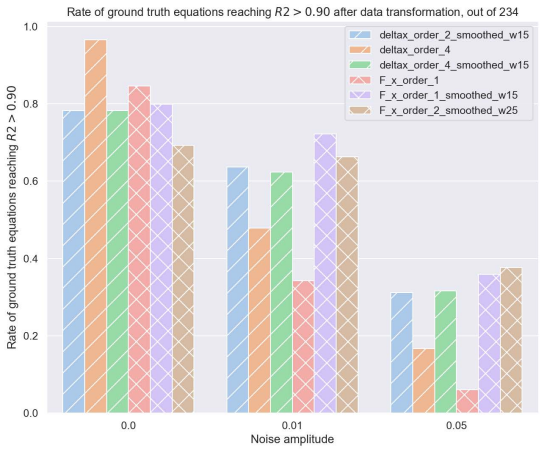

Figure 2: Best-performing combinations of hyperparameters for transformations Δx (stripes) and F_x (crosses). Δx performs the best in absence of noise, while F_x seems to be able to take the most advantage out of the denoising performed by the SG filter.

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 [14]), 59 (Rössler attractor, chaotic [19]), and 61 (Chen-Lee attractor [2]).

The SR tool selected for this task is PySR [3], with 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.

From those experiments, 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. 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 $x_0'(t) = -10.0x_0 + 10.0x_1$ ($R2 = 0.7039$). In this case, only the values of the numerical parameters are incorrect. However, for trajectory 1 of system 55, 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 $x_2'(t) = x_0x_1 - 2.67x_2$ ($R2 = 0.2345$)!

An example of the same behavior for transformation F_x is system 61, trajectory 1; the equation found $F_{x_0} = \Delta_t x_1\ (-1.0x_2 - 3.85) = \Delta_t\ (3.85x_1 - x_1x_2)$, ($R2 = 0.8687$), shares the same structure of the ground truth $F_{x_0} = \Delta_t\ (5.0x_0 - x_1x_2)$, ($R2 = 0.7793$), albeit with a different value of a numerical parameter. On the other hand, again for system 61, trajectory 1, the equation found for variable $F_{x_2} = \Delta_t \cos(0.107x_0)$, ($R2 = 0.6461$), is radically different from the ground truth $F_{x_2} = \Delta_t\ (0.333x_0x_1 - 3.8x_2)$, ($R2 = 0.5235$).

A general trend 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 low, SR converges on undesirable candidate solutions with completely different structures.

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.

From the experimental results, it is possible to provide general recommendations for practitioners: 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.

References

[1] 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.

[2] 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

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

[4] 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

[5] 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

[6] 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

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

[8] 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.

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

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

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

[12] 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

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

[14] 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

[15] 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

[16] 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

[17] 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

[18] 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

[19] 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

[20] 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

[21] 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