Discovering Symmetries of ODEs by Symbolic Regression

Paul Kahlmeyer, Niklas Merk, Joachim Giesen

Friedrich Schiller University Jena
paul.kahlmeyer@uni-jena.de, niklas.merk@uni-jena.de, joachim.giesen@uni-jena.de

Abstract

Solving systems of ordinary differential equations (ODEs) is essential when it comes to understanding the behavior of dynamical systems. Yet, automated solving remains challenging, in particular for nonlinear systems. Computer algebra systems (CASs) provide support for solving ODEs by first simplifying them, in particular through the use of Lie point symmetries. Finding these symmetries is, however, itself a difficult problem for CASs. Recent works in symbolic regression have shown promising results for recovering symbolic expressions from data. Here, we adapt search-based symbolic regression to the task of finding generators of Lie point symmetries. With this approach, we can find symmetries of ODEs that existing CASs cannot find.

Introduction

The study of dynamical systems plays a key role in all natural sciences (Strogatz 2000), as well as in economics (Lorenz 1993) and in the social sciences (Richardson and Dale 2014). Describing the state of dynamical systems explicitly as a function over time, that is, in integral form, is usually difficult. It is often much easier to describe the system by the change of its state over time, that is, in differential form. Systems of ordinary differential equations (ODEs) are such a description in differential form. Systems of ODEs are a key mathematical concept for modeling and understanding dynamical systems (Hirsch and Smale 1974; Tenenbaum and Pollard 1985). Transforming the differential form into integral form means solving an ODE system, that is, finding an explicit function that satisfies the ODE. Solving ODEs is crucial for predicting system behavior and for analyzing its dynamics. A large part of research has been devoted to finding solutions to ODEs by numerical or analytical methods. Numerical solutions involve simulating the system from a specific initial condition, offering effective solutions for a wide range of ODEs, but lacking the interpretability of analytical solutions.

For an example, consider a system, where we observe a quantity $y = [y_1, y_2]$ over time t. The dynamics of the system is such that the change of the first component y_1 is negatively correlated with the magnitude of the second component y_2, whereas the change of the second component is positively correlated with the magnitude of the first component. This simple system can be modeled by the following system of two linear ODEs

$$y_1' = -y_2$$
$$y_2' = y_1$$

where y_i' is the derivative of the time-dependent function y_i. Three numerical solutions for the example system are shown in Figure 1. Note that these solutions consist only of sample points. The system is simple enough that we can also provide an analytical solution

$$y_1(t) = a \cdot \cos(t + b)$$
$$y_2(t) = a \cdot \sin(t + b)$$

for some constants a and b. The three numerical solutions in Figure 1 correspond to different choices of the constants a and b.

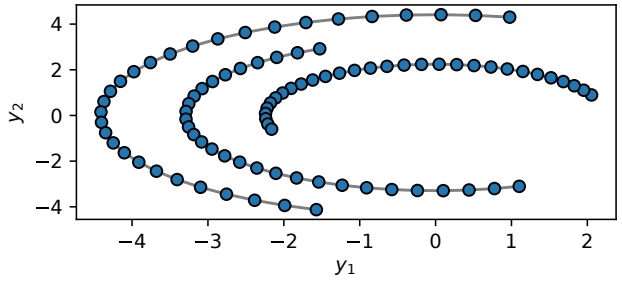

Figure 1: Three numerical solutions for the example ODEs $y' = [-y_2, y_1]$ in state space. Time steps are equally spaced between two adjacent points of a single trajectory.

Many natural phenomena, however, are described by nonlinear systems of ODEs, for instance chaotic systems. Nonlinear ODEs are typically much more difficult to solve analytically. Solutions are known only in special cases, when the nonlinearities can be resolved by transforming the system into an equivalent system of linear ODEs, such as Riccati or Bernoulli differential equations. Sophus Lie (Lie 1888) proposed a fundamental and unifying framework for the analysis of systems of ODEs using symmetries. A symmetry of an ODE system describes a mapping from one

solution to another one. Symmetries can be used to derive a coordinate transformation. In the transformed coordinate system, the number of equations of a system of ODEs is reduced by one. Sometimes the reduced system is easier to solve analytically, or it can be reduced further by exploiting another symmetry. Our linear example system has a particularly simple symmetry, it is invariant under positive scaling. Therefore, a parameterized family of symmetries is given by

$$h_\theta(t, y_1, y_2) = \left(t, e^\theta y_1, e^\theta y_2\right) =: \left(\hat{t}, \hat{y}_1, \hat{y}_2\right),$$

where $\theta \in \mathbb{R}$ parameterizes the family. This is easily verified, because

$$e^\theta y_1(t) = \hat{a} \cdot \cos(t + b) \quad \text{and} \quad e^\theta y_2(t) = \hat{a} \cdot \sin(t + b),$$

with $\hat{a} = e^\theta a$, is a solution of the ODE

$$\hat{y}_1' = -\hat{y}_2$$
$$\hat{y}_2' = \hat{y}_1,$$

where $\hat{y}_i'$ is the derivative of $\hat{y}_i$ with respect to $\hat{t}$. In the full version of the paper, we demonstrate how this family of symmetries can be used to simplify the example system. For nonlinear systems, the challenge becomes to find the symmetries.

Given an ODE system in symbolic form, the necessary condition for a symmetry can be expressed as a system of partial differential equations (PDEs). Traditionally, CASs have been used to solve the PDE system symbolically. This approach however is again limited by the capabilities of CASs to tackle nonlinear systems. In this work, we propose to use symbolic regression to search directly for expressions that satisfy the symmetry condition. Unlike traditional regression methods, symbolic regression does not fixate on a given class of functions, but rather searches in the space of *all* possible functions that can be constructed from a given vocabulary of operators. Recent progress in symbolic regression has produced efficient search algorithms that in contrast to previous work in symbolic regression can also be used to search for symmetries of ODE systems. We demonstrate that symbolic regression is able to find symmetries of nonlinear systems where the traditional CAS approach fails, making it a valuable extension for any ODE related toolbox.

Related Work

We organize our discussion of related work along three lines of research, namely, computer algebra systems for simplifying and solving ODEs, symbolic regression, and the related field of learning conserved quantities and invariants of dynamical systems.

Computer Algebra Systems

Computer algebra systems (CASs) are software programs designed for performing symbolic mathematical operations. Unlike numerical software, which primarily deals with numerical values and calculations, CASs manipulate mathematical expressions and symbols directly, enabling tasks such as algebraic simplification, equation solving, symbolic differentiation, and symbolic integration. The ability of CASs to solve ODEs or the corresponding symmetry PDEs relies on specialized algorithms for detecting systems with specific structures, such as, linear, Riccati-type, or Bernoulli-type, which have known solutions.

Examples of CASs include the commercial software systems Mathematica (Wolfram Research Inc. 2024) and Maple (Maplesoft, a division of Waterloo Maple Inc. 2019). Both systems provide a wide range of functionalities for analyzing ODEs and their symmetries. Maple supports symbolic symmetry finding for ODEs through the symgen package (Cheb-Terrab and von Bülow 1995; Cheb-Terrab, Duarte, and da Mota 1997, 1998), which iteratively tests for special structures, but notably cannot handle systems of ODEs. For Mathematica, two packages for finding symmetries exist: (1) the SYM package (Dimas and Tsoubelis 2005) and (2) the MathLie package (Baumann 2002). Both packages can handle systems of ODEs and compute symmetries by deriving the corresponding partial differential equations and trying to solve them.

Symbolic Regression

Given a training set of labeled data, the goal in symbolic regression is to find an interpretable expression, where the underlying function generalizes well beyond the training data. Symbolic regression algorithms differ in the way they search the space of expressions. The majority of algorithms follow the *genetic programming* paradigm (Koza 1994; Holland 1975). The paradigm has been implemented in the seminal Eureqa system by Schmidt and Lipson (2009) and in gplearn by Stephens (2016). More recent implementations are provided by La Cava, Spector, and Danai (2016); Kommenda et al. (2020); Virgolin et al. (2021). The basic idea is to create a population of expression trees and to recombine the best performing trees into a new population by exchanging subtrees. Alternatively, Bayesian inference methods leverage Markov Chain Monte Carlo (MCMC) sampling to draw expressions from a posterior distribution, with variations in the choice of prior distribution. Jin et al. (2020) use a hand-designed prior distribution on expression trees, whereas Guimerà et al. (2020) compile a prior distribution from a corpus of mathematical expressions. A different branch of work focuses on using deep learning: Petersen et al. (2021) and Mundhenk et al. (2021) train recurrent neural networks, Kamienny et al. (2022) train a transformer for generating expression trees token-by-token in preorder. Recently, Landajuela et al. (2022) have combined different symbolic regressors based on genetic programming, reinforcement learning, and transformers together with problem simplification into a unified symbolic regressor.

Overall, the idea of an unbiased, more or less assumption-free search has shown the greatest potential when it comes to recovering expressions from data. The AIFeynman project (Udrescu and Tegmark 2020) uses a set of neural-network-based statistical property tests to scale an unbiased search to larger search spaces. Statistically significant properties, for example, additive or multiplicative separability, are used to decompose the search space into

smaller parts, which can be addressed by brute-force search. Recently, Kahlmeyer et al. (2024) have proposed an unbiased search for expressions represented by directed, acyclic graphs (DAGs). Unlike the more popular expression tree representation, DAGs allow for criterion-based search considering all output dimensions simultaneously.

Conserved Quantities and Invariants

Conserved quantities and invariants are both closely related to symmetries. In a celebrated theorem, Noether (1918) has shown a correspondence between continuous symmetries and conservation laws for physical systems. Liu and Tegmark (2021) have designed and implemented the AI Poincaré algorithm for directly learning the number and, by using their symbolic regressor AIFeynman (Udrescu and Tegmark 2020), the symbolic form of conserved quantities from time series data. The extension Poincaré 2.0 by Liu, Madhavan, and Tegmark (2022) uses numerically generated time series data for learning n-th order conserved quantities from ODEs and some PDEs. Liu and Tegmark (2022) generalize this work and propose an automated workflow for the discovery of a set of six predefined symmetry classes by learning a coordinate transformation that makes these symmetries apparent. The coordinate transformation is approximated by a neural network and translated into a symbolic form using AIFeynman. Bondesan and Lamacraft (2019) follow a similar approach and learn a coordinate transformation, which makes it easier to identify conserved quantities.

Our proposed approach differs from the aforementioned work in two major aspects. First, conserved quantities directly correspond to a special kind of symmetry, namely, the Hamiltonian symmetry. While it can reveal interesting properties of the underlying system, we explicitly exclude these symmetries in this work, as they cannot be used to derive the coordinate transformation that simplifies the system. Second, we do not use an intermediary approximation by some black box module like a neural network, but apply symbolic regression directly to the underlying system.

Symmetries of ODEs

Before we can describe our approach for learning symmetries for ODEs, we briefly review ODEs and related symmetry concepts that are used in our approach. For a more detailed treatment of the mathematical background, we recommend the books by Bluman and Kumei (1989), Stephani (1993), or Lie (1888).

A first order ODE system is a system of equations of the form

$$y' = f(t, y),$$

where y' denotes the derivative of the function

$$y : U \subseteq \mathbb{R} \to \mathbb{R}^d,\ t \mapsto y(t)$$

and f is a function $f : V \subseteq \mathbb{R}^{d+1} \to \mathbb{R}^d$. Considering only first order ODEs is not really a restriction. Any higher order ODE system can be expressed by an equivalent first order ODE system using additional variables.

A (point) symmetry transformation of an ODE system is a differentiable, invertible map

$$h : (t, y) \mapsto \big(\hat{t}(t, y), \hat{y}(t, y)\big)\,,$$

which maps each solution $y(t)$ of the ODE to a solution $\hat{y}(\hat{t})$. For a differentiable symmetry transformation h, the symmetry condition, which requires that also $\hat{y}(\hat{t})$ is a solution of the ODE, is not difficult to check. However, the search space for symmetry transformations is vast, yet not unstructured. The set of symmetry transformations for an ODE has a group structure, where the group operation is the composition of transformations and the neutral element is the identity transformation. For solving ODEs, one-parameter subgroups of the symmetry group, namely one-parameter Lie groups, are particularly interesting. In our approach, the function f is known. Therefore, we can sample data from f and then use the data to learn Lie symmetries of the corresponding ODE system.

A one-parameter Lie group is a family of symmetry transformations

$$h_\theta : (t, y) \mapsto \big(\hat{t}(\theta, t, y), \hat{y}(\theta, t, y)\big)$$

that are smooth in t and y, and analytical in the parameter $\theta \in \mathbb{R}$. The group structure needs to satisfy

$$h_\theta \circ h_{\theta'} = h_{\theta+\theta'} \quad \text{and} \quad h_1 = \mathrm{id}.$$

Instead of learning individual symmetry transformations, we would like to learn the whole one-parameter Lie group at once. It turns out that this is possible, because the Lie group, that is, all its symmetry transformations, is determined by its generator, which is the tangent vector field X on $\mathbb{R}^{d+1}$ of the flow defined by $\theta \mapsto \{h_\theta(t, y)\}$ at $\theta = 0$. The generator is unique up to scaling. Using the inner product $\langle \cdot, \cdot \rangle$ on $\mathbb{R}^d$ and ∇_y, the gradient with respect to y, its components can be expressed as

$$X(t, y) = \xi(t, y)\partial_t + \big\langle \eta(t, y), \nabla_y \big\rangle$$
$$\text{with } \xi(t, y) \coloneqq \frac{\mathrm{d}\hat{t}}{\mathrm{d}\theta}\Big|_{\theta=0} \text{ and } \eta(t, y) \coloneqq \frac{\mathrm{d}\hat{y}}{\mathrm{d}\theta}\Big|_{\theta=0}.$$

The concept of a generator is visualized in Figure 2.

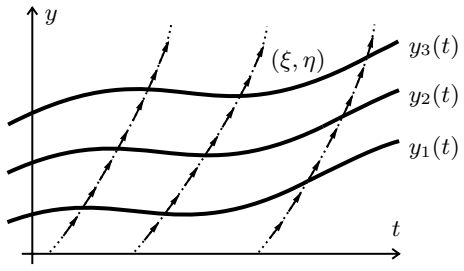

Figure 2: Generator of a Lie symmetry as a tangent vector field. y_1, y_2 and y_3 are solutions of the ODE for different initial values. The dotted lines represent the Lie symmetry, transforming the solutions into each other.

For learning the generator, it remains to express the symmetry condition $\hat{y}' = f(\hat{t}, \hat{y})$ in terms of the functions ξ and η. The symmetry condition states that the zero set of $\hat{y}' - f(\hat{t}, \hat{y})$ does not change under symmetry transformations, which for one-parameter Lie groups can be expressed

as

$$
\begin{aligned}
0 &= \frac{\mathrm{d}}{\mathrm{d}\theta}\Big|_{\theta=0}\big(\hat{y}' - f(\hat{t},\hat{y})\big) \\
&= \frac{\mathrm{d}\hat{y}'}{\mathrm{d}\theta}\Big|_{\theta=0} - \frac{\partial f}{\partial \hat{t}}\frac{\mathrm{d}\hat{t}}{\mathrm{d}\theta}\Big|_{\theta=0} - \Big(\nabla_{\hat{y}} f\Big)\cdot\frac{\mathrm{d}\hat{y}}{\mathrm{d}\theta}\Big|_{\theta=0} \\
&= \frac{\mathrm{d}\hat{y}'}{\mathrm{d}\theta}\Big|_{\theta=0} - \frac{\partial f}{\partial t}\frac{\mathrm{d}\hat{t}}{\mathrm{d}\theta}\Big|_{\theta=0} - \Big(\nabla_{y} f\Big)\cdot\frac{\mathrm{d}\hat{y}}{\mathrm{d}\theta}\Big|_{\theta=0} \\
&= \frac{\mathrm{d}\hat{y}'}{\mathrm{d}\theta}\Big|_{\theta=0} - \frac{\partial f}{\partial t}\xi(t,y) - \Big(\nabla_{y} f\Big)\cdot\eta(t,y),
\end{aligned}
$$

where $\nabla_y f$ is a Jacobi matrix. For the second to last equation, we have used that

$$\hat{t}(\theta=0,t,y) = t \quad \text{and} \quad \hat{y}(\theta=0,t,y) = y,$$

and the last equality follows from the definition of the generator X. The only expression in the symmetry condition that still involves $\hat{t}$ and $\hat{y}$ is $\frac{\mathrm{d}\hat{y}'}{\mathrm{d}\theta}\Big|_{\theta=0}$. For rewriting this expression as a function of t and y, we expand $\hat{y}$ and $\hat{t}$ in orders of the parameter θ,

$$
\begin{aligned}
\hat{y} &= y + \theta\cdot\eta(t,y) + o(\theta^2) \\
\hat{t} &= t + \theta\cdot\xi(t,y) + o(\theta^2).
\end{aligned}
$$

From plugging the expansions into the derivative $\hat{y}'$,

$$
\begin{aligned}
\hat{y}' &= \frac{\mathrm{d}\hat{y}}{\mathrm{d}\hat{t}} = \frac{\mathrm{d}\hat{y}}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\hat{t}} = \frac{\mathrm{d}\hat{y}}{\mathrm{d}t}\Big(\frac{\mathrm{d}\hat{t}}{\mathrm{d}t}\Big)^{-1} = \frac{y' + \theta\frac{\mathrm{d}\eta}{\mathrm{d}t} + o(\theta^2)}{1 + \theta\frac{\mathrm{d}\xi}{\mathrm{d}t} + o(\theta^2)} \\
&= y' + \theta\left(\frac{\mathrm{d}\eta}{\mathrm{d}t} - y'\frac{\mathrm{d}\xi}{\mathrm{d}t}\right) + o(\theta^2),
\end{aligned}
$$

where the last equality is implied by

$$\big(1 + \theta a + o(\theta^2)\big)^{-1} = 1 - \theta a + o(\theta^2),$$

it follows that

$$
\begin{aligned}
\frac{\mathrm{d}\hat{y}'}{\mathrm{d}\theta}\Big|_{\theta=0} &= \frac{\mathrm{d}\eta}{\mathrm{d}t} - y'\frac{\mathrm{d}\xi}{\mathrm{d}t} \\
&= \frac{\partial\eta}{\partial t} + \big(\nabla_y\eta\big)\cdot y' - \Big(\frac{\partial\xi}{\partial t} + \big\langle\nabla_y\xi, y'\big\rangle\Big)y'.
\end{aligned}
$$

Here, $\nabla_y\eta$ is also a Jacobi matrix.

The symmetry condition can finally be expressed only in terms of functions of t and y, namely,

$$\frac{\partial\eta}{\partial t} + \big(\nabla_y\eta\big)\cdot y' - \Big(\frac{\partial\xi}{\partial t} + \big\langle\nabla_y\xi, y'\big\rangle\Big)y' - \xi\frac{\partial f}{\partial t} - \big(\nabla_y f\big)\cdot\eta = 0.$$

Hamiltonian Symmetry

In principle, the symmetry condition is all we need to define and compute a symmetry loss function that can guide our search for symmetries. Given f, we search for functions $\eta(t,y)$ and $\xi(t,y)$, that fulfill the symmetry condition. In practice, however, there remain technical challenges when it comes to estimating a meaningful generator. Every ODE has at least one one-parameter symmetry group, namely, its Hamiltonian symmetry, that is, the time-evolution symmetry,

$$h_\theta(t, y(t)) = \big(t + \theta, y(t+\theta)\big),$$

with symmetry generator

$$X_H \coloneqq \partial_t + \langle f, \nabla_y\rangle.$$

The Hamiltonian generator $X_H(t,y)$ and actually also any multiple $\xi(t,y)X_H(t,y)$ are indeed symmetry generators. This can be seen by plugging ξX_H into the symmetry condition, which gives

$$\xi\left(\frac{\mathrm{d}f}{\mathrm{d}t} - \frac{\partial f}{\partial t} - \big(\nabla_y f\big)\cdot f\right) + f\frac{\mathrm{d}\xi}{\mathrm{d}t} - \frac{\mathrm{d}\xi}{\mathrm{d}t}y' = 0,$$

where the terms cancel, because $y' = f$. However, the Hamiltonian symmetry X_H and any multiple ξX_H do not provide more information than the already known function f and thus can not be used for simplifying the ODE system. Yet, a symmetry generator X might contain a multiple of the Hamiltonian symmetry,

$$
\begin{aligned}
X &= \xi\partial_t + \langle\eta, \nabla_y\rangle \\
&= \xi\partial_t + \xi\langle f, \nabla_y\rangle - \xi\langle f, \nabla y\rangle + \langle\eta, \nabla_y\rangle \\
&= \xi\big(\partial_t + \langle f, \nabla_y\rangle\big) + \langle\eta - \xi f, \nabla_y\rangle \\
&= \xi X_H + \langle\eta - \xi f, \nabla_y\rangle.
\end{aligned}
$$

Since the Hamiltonian symmetry as well as its multiples are not informative for our task, we remove the contribution of the Hamiltonian symmetry from any estimated symmetry generator, and only consider the non-trivial part

$$\langle\eta^\star, \nabla_y\rangle \coloneqq \langle\eta - \xi f, \nabla_y\rangle = X - \xi X_H.$$

The non-trivial part is also a generator, because generators form a vector space, X is a generator by definition, and we have established already that ξX_H is a generator. It provides information about a transformation from one time series into another, but not the trivial information about traversing a solution (time-evolution), which is given by the Hamiltonian symmetry.

Removing the trivial part of the generator also simplifies the symmetry condition, which becomes,

$$
\begin{aligned}
0 =& \frac{\partial\eta^\star}{\partial t} + \big(\nabla_y\eta^\star\big)\cdot y' - \Big(\nabla_y f\Big)\cdot\eta^\star \\
&= \nabla\eta^\star\cdot\begin{bmatrix}1\\ f\end{bmatrix} - \nabla f\cdot\begin{bmatrix}0\\ \eta^\star\end{bmatrix},
\end{aligned}
\tag{1}
$$

where $\nabla\eta^\star$ and ∇f are Jacobi matrices of derivatives with respect to t and y.

By definition, any generator $X = \langle\eta^\star, \nabla_y\rangle$, that satisfies the simplified symmetry condition, does not contain a multiple of the Hamiltonian symmetry generator. Therefore, the search space of generators is reduced to the space of informative generators, that do not contain redundant information from the Hamiltonian generator.

The implicit removal of the trivial part of symmetry generators also facilitates the simultaneous use of information from different symmetries, by considering multiple independent generators. A set $\{X_k\}_{k=1}^n$ of generators is called *dependent* if all the generators X_k are linearly dependent along all solutions, that is, there exist non-zero constants $\{c_k\}_{k=1}^n$ such that $\sum_{k=1}^n c_k X_k(t,y) = 0$ along all solutions of the

ODE. Without implicitly (or explicitly) removing the trivial part of the generators, it is not straightforward to find independent generators that are informative. To see this, consider the generators $X = \langle \eta^\star, \nabla_y \rangle$ and $X' = X + \kappa X_H$, with a non-zero continuously differentiable function $\kappa(t, y)$. The two generators contain, for our purpose, the same information. They are, however, independent, as the ∂_t component is zero for X and non-zero for X'. This shows, that ignoring multiples of the Hamiltonian symmetry generator makes independence of generators a meaningful measure of independent information.

Symbolic Regression for Symmetry Finding

Given a d-dimensional ODE system, our goal is use symbolic regression to search for a generator function

$$\eta^\star : \mathbb{R}^{d+1} \to \mathbb{R}^d$$

in symbolic form, that satisfies the symmetry condition in Equation 1. The structure of the symmetry condition is such that it requires us to compute all outputs of $\eta^\star$ simultaneously, which is a problem, because most of the established symbolic regressors split multi-output regression problems into single-output regression problems. An exception is the state-of-the-art symbolic regression framework (Kahlmeyer et al. 2024) that can directly deal with multi-output regression problems. Kahlmeyer et al. (2024) construct their regressor in two steps: First, expression DAG skeletons are created up to a given size that depends on the available computational budget. Here, DAG skeletons are expression DAGs whose operator nodes have not yet been labeled with function symbols. Second, each DAG skeleton is turned into a family of expression DAGs by validly labeling its operator nodes with function symbols. The expression DAGs are then scored by the given loss function, and the regressor is given by the expression DAG that incurs the smallest loss. The construction process is illustrated in Figure 3.

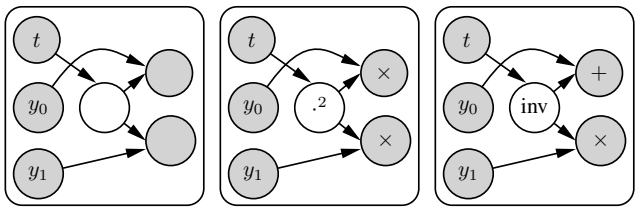

Figure 3: The symbolic regression framework (Kahlmeyer et al. 2024) works by creating DAG skeletons (left), that are turned into expression DAGs by assigning valid function symbols to the empty operator nodes (middle and right). Here, the assignment in the middle produces the expression $\eta^\star(t, y_1, y_2) = (t^2 y_1, t^2 y_2)$ and the assignment on the right produces the expression $\eta^\star(t, y_1, y_2) = (y_0 + t^{-1}, y_1 t^{-1})$.

For symbolic regression, Kahlmeyer et al. (2024) use the mean-squared error (MSE) between a DAG's output and the given observations as loss function. For our task of finding symmetries of ODE systems, we need to replace this loss function by a loss function that is based on the symmetry condition. Such a loss function is constructed as follows: Let $f : \mathbb{R}^{d+1} \to \mathbb{R}^d$ be the symbolic form of the ODE system under investigation. We first solve the ODE numerically by the Runge-Kutta algorithm (Dormand and Prince 1980) that computes a sample $\mathcal{D} = \big(t_i, y(t_i)\big)_{i=1,\dots,n}$ of the ODE system's solution. For each dimension $k = 1, \dots, d$ of the ODE system, we can evaluate a candidate symmetry generator $\eta^\star$ at a given value pair $(t, y) \in \mathcal{D}$, which gives

$$S^k_{\eta^\star}(t, y) = \nabla \eta^\star_k(t, y) \begin{bmatrix} 1 \\ f(t, y) \end{bmatrix} - \nabla f_k(t, y) \begin{bmatrix} 0 \\ \eta^\star(t, y) \end{bmatrix}.$$

The loss of the candidate symmetry generator $\eta^\star$ is then given as

$$L_{\mathcal{D},f}(\eta^\star) = \frac{1}{nd} \sum_{(t,y)\in\mathcal{D}} \sum_{k=1}^{d} \big(S^k_{\eta^\star}(t, y)\big)^2.$$

The loss function evaluates to zero if, and only if the symmetry condition is satisfied at all points in the numeric solution $\mathcal{D} = \big(t_i, y(t_i)\big)_{i=1,\dots,n}$ in every dimension of the ODE system. For implementing the loss function we need to compute the gradients $\nabla\eta^\star$. The internal representation of the DAG facilitates calculating these gradients via automatic differentiation, for instance, by the autodiff engine provided in PyTorch (Paszke et al. 2017).

Finally, note that the generator $\eta^\star = 0$ always satisfies the generator condition. The corresponding symmetry however gives no additional information about the system, as it maps every solution back onto itself. In order to avoid the trivial generator, our implementation discards candidate expression DAGs for $\eta^\star$ where the *median absolute value* of $\eta^\star$ evaluated at the data points is smaller than a threshold value of $\varepsilon = 0.01$.

The whole procedure for finding a symbolic generator $\eta^\star$ is summarized in Figure 4.

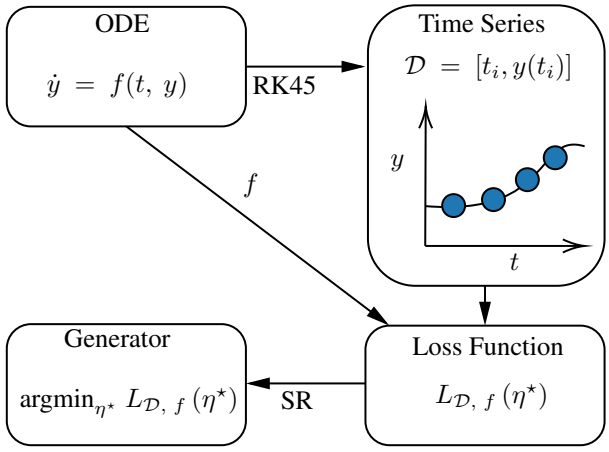

Figure 4: Flowchart of our method: From a given ODE, time series data $\mathcal{D}$ are simulated using the Runge-Kutta algorithm (RK45). Based on this data and f, a loss function for expressions is derived. We finally use a symbolic regression framework (SR) to search for a symbolic symmetry generator $\eta^\star$ that minimizes the loss function.

Discussion

We provide examples on which our symbolic regression method finds symmetry generators while the established algebraic symmetry finding methods struggle to do so. That is, the symbolic regression approach enlarges the set of problems that can be tackled by symmetry finding methods. Here, we compare our approach only to Mathematica's well-maintained SYM package (Dimas and Tsoubelis 2005), because it essentially subsumes the other packages. Maple's symgen package cannot handle ODE systems, and the MathLie package heavily relies on ODE-specific user interaction. In the following, we first use the example ODE system from the introduction to illustrate different modes that are used by SYM for finding symmetries, before we provide example problems that are difficult for SYM. Finally, we briefly discuss the added benefit of searching for independent generators and provide running times for the search-based symbolic regression approach.

SYM's Symmetry Finding Approach

We illustrate SYM's symmetry finding approach, that comprises an automatic and an interactive mode, on the simple ODE system from the introduction,

$$f(t, y_1, y_2) = \begin{bmatrix} -y_2 \\ y_1 \end{bmatrix}.$$

We are looking for a generator

$$X(t, y_1, y_2) = \left\langle \begin{bmatrix} \xi(t, y_1, y_2) \\ \eta_1(t, y_1, y_2) \\ \eta_2(t, y_1, y_2) \end{bmatrix}, \begin{bmatrix} \partial_t \\ \partial_{y_1} \\ \partial_{y_2} \end{bmatrix} \right\rangle$$

in the form of the functions ξ and η_i. Note that, since this ODE system can be solved directly by Mathematica, there is no need for a symmetry analysis when it comes to solving the system. The system is useful, however, for demonstrating SYM's algebraic symmetry finding approach.

Automatic mode. Given the symbolic form of f in our example system, the SYM package derives the PDEs for the general generator condition,

$$\begin{aligned} 0 = \quad & \eta_2 + y_1 \frac{\partial \eta_1}{\partial y_2} + y_1 y_2 \frac{\partial \xi}{\partial y_2} - y_2 \frac{\partial \eta_1}{\partial y_1} \\ & - y_2^2 \frac{\partial \xi}{\partial y_1} + \frac{\partial \eta_1}{\partial t} + y_2 \frac{\partial \xi}{\partial t} \\ 0 = \quad & \eta_1 - y_1 \frac{\partial \eta_2}{\partial y_2} + y_1^2 \frac{\partial \xi}{\partial y_2} + y_2 \frac{\partial \eta_2}{\partial y_1} \\ & - y_1 y_2 \frac{\partial \xi}{\partial y_1} - \frac{\partial \eta_2}{\partial t} + y_1 \frac{\partial \xi}{\partial t} \end{aligned}$$

with the unknown functions ξ and η_i. SYM then iteratively attempts to reduce the number of PDEs by solving one PDE for an unknown function and substituting the resulting expression into the remaining PDEs until only one PDE remains. The remaining PDE acts as a constraint for the solutions that were used as substitutions. In our example, SYM solves the second PDE for η_1, that is,

$$\begin{aligned} \eta_1 = \quad & y_1 \frac{\partial \eta_2}{\partial y_2} - y_1^2 \frac{\partial \xi}{\partial y_2} - y_2 \frac{\partial \eta_2}{\partial y_1} \\ & + y_1 y_2 \frac{\partial \xi}{\partial y_1} + \frac{\partial \eta_2}{\partial t} - y_1 \frac{\partial \xi}{\partial t}, \end{aligned}$$

that is plugged into the first equation, which then acts as a constraint. For finding generators, one now has to solve the PDE constraint and use the solution to calculate η_1. In general, we consider this approach successful if all PDE constraints can be solved by Mathematica. This is, however, already not the case for our simple example system.

Interactive mode. Alternatively, the user can *interactively* fix all but one of the unknown functions of the PDE constraints and then use Mathematica to solve for the remaining functions. For our example system, when setting $\eta_2 = \xi = 0$ in order to exclude the Hamiltonian symmetry, the first PDE reduces to

$$0 = y_1 \frac{\partial \eta_1}{\partial y_2} - y_2 \frac{\partial \eta_1}{\partial y_1} + \frac{\partial \eta_1}{\partial t},$$

which Mathematica can solve with the result

$$\eta_1 = y_1 \cos(t) + y_2 \sin(t).$$

Our search-based symbolic regression approach implicitly excludes the Hamiltonian symmetry by setting $\xi = 0$ and directly finds the generators $\eta_1 = \cos(t)$ and $\eta_2 = \sin(t)$, which can be verified to satisfy the generator condition.

Hard Problems for Mathematica and SYM

We demonstrate the added value of the symbolic regression approach on ten representative ODE systems for which Mathematica does not find an explicit solution directly, that is, without the SYM package. More details on the ODEs can be found in the full version of the paper.

We test SYM's symmetry finding abilities on the ten examples by running it in three modes: First, the automatic mode, where SYM's result is a closed form expression for one of the generator parts, with constraints. The constraints are themselves a system of PDEs. We consider SYM successful, if the constraint PDEs can be solved by Mathematica. The second and third approach are variants of the interactive mode. We either set all but one component to zero (SYM-Z) or to the values of a known generator (SYM-S) and solve for the missing components. Note, however, that the latter approach is usually infeasible in practice. The results are shown in Table 1.

The automatic procedure of SYM fails on all the examples, that is, not all the PDE constraints of the reformulated symmetry generator condition can be solved by Mathematica. When setting all but one generator function to zero, Mathematica is able to solve the determining equations only in one case. Even in the unrealistic case that all but one generator function are known, Mathematica fails to find a generator in three cases.

Our search-based symbolic regression approach automatically finds symmetry generators for all ten problems.

	ODE: $f(y_1, y_2) =$	SYM	SYM-Z	SYM-S
1	$\begin{bmatrix} y_1(t + y_2 y_1^{-1})^2 \\ t^2 y_1 \end{bmatrix}$	✗	✗	✓
2	$\begin{bmatrix} y_1^2 y_2 \exp(-y_1) \\ t \exp\left(-\frac{1}{y_1}\right) \end{bmatrix}$	✗	✗	✓
3	$\begin{bmatrix} t y_1 (y_2 - \ln(y_1)) \\ t + y_2 - \ln(y_1) \end{bmatrix}$	✗	✗	✓
4	$\begin{bmatrix} \frac{2y_1 + y_2 \exp(-\frac{y_1}{t^2})}{t} \\ y_2 \end{bmatrix}$	✗	✓	✓
5	$\begin{bmatrix} y_1(t - \ln(y_1)\tan(t)) \\ y_2(1 - \ln(y_1)\tan(t)) \end{bmatrix}$	✗	✗	✓
6	$\begin{bmatrix} y_1(t y_2 y_1^{-1} + \frac{2\ln(y_1)}{t}) \\ \frac{2y_2 \ln(y_1)}{t} \end{bmatrix}$	✗	✗	✓
7	$\begin{bmatrix} \frac{y_2}{y_1} \exp(-\frac{y_1^2}{2t^2}) + \frac{y_1}{2t} \\ -\frac{1}{2t^3} y_1^2 y_2 \end{bmatrix}$	✗	✗	✓
8	$\begin{bmatrix} \sin(y_2)\exp(-t) \\ \sin(y_1)\exp(-t) \end{bmatrix}$	✗	✗	✗
9	$\begin{bmatrix} t y_2 \sin(y_1) \\ t \sin(y_1) \end{bmatrix}$	✗	✗	✗
10	$\begin{bmatrix} t \ln(y_2) \\ t y_1^2 \end{bmatrix}$	✗	✗	✗

Table 1: Performance, indicated by success (✓) and failure (✗), of SYM on ten representative example problems for which symbolic regression finds simple symmetry generators.

Independent Generators

We have argued that the implicit exclusion of the Hamiltonian symmetry allows searching for independent generators. Here, we demonstrate this feature on the following nonlinear system,

$$f(t, y_1, y_2) = \begin{bmatrix} \sqrt{y_1} t \\ y_1 y_2 t \end{bmatrix},$$

where our approach discovers the two independent generators

$$\eta_1^\star = \begin{bmatrix} 0 \\ y_2 \end{bmatrix} \quad \text{and} \quad \eta_2^\star = \begin{bmatrix} \sqrt{y_1} \\ y_1 y_2 \end{bmatrix}.$$

From the two independent generators we can derive two different coordinate transformations leading to two distinct one-dimensional ODEs,

$$s_1'(r) = \sqrt{s_1} r \quad \text{and} \quad s_2'(r) = 0.$$

When either of these systems can be solved, the solution can be translated back into the original coordinates, thus solving the original problem. The trivial second ODE is solved by $s_2(r) = 0$ and leads to a correct solution,

$$y_1(t) = \frac{t^4}{16} \quad \text{and} \quad y_2(t) = e^{\frac{t^6}{96}},$$

of the original problem. Solving the first ODE with $s_1(r) = \frac{r^4}{16}$ leads to the same solution. A detailed derivation of these results can be found in the full version of the paper.

Running Times

The running time of the proposed method exclusively depends on the underlying symbolic regressor. By design, the running time of the regressor grows with the size of the search space for symbolic symmetry generators, which itself grows exponentially with the size of the covered expression DAGs (Kahlmeyer et al. 2024).

In Figure 5, we show the average runtime for our constructed example problems together with the size of the expression tree of the found generators. As expected, our method is significantly faster when the generator has a simple expression. Nevertheless, even for the more challenging examples, generators are found within a reasonable time of under six minutes, running Python 3.10 on an Intel Xeon Gold 6226R 64-core processor with 128 GB of RAM.

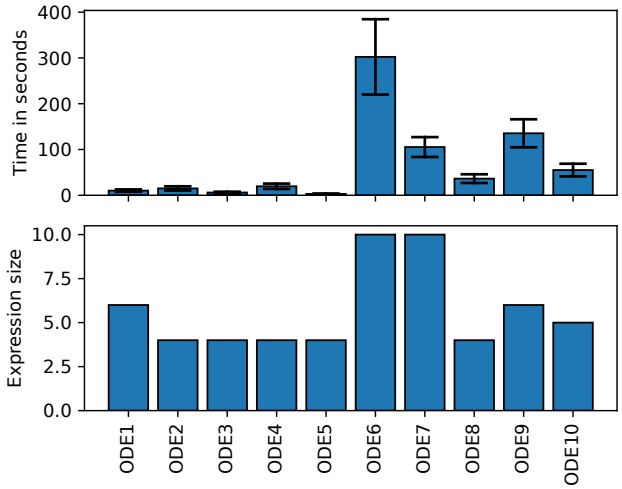

Figure 5: Top: Average running times and 95% confidence intervals over 50 independent runs of the symbolic regression approach on the ten problems that are hard for SYM. Bottom: Tree sizes of the `sympy` expressions (Meurer et al. 2017) for the symmetry generators found by the symbolic regressor.

Conclusion

We have introduced a novel application of symbolic regression, namely, the identification of symmetries of systems of ordinary, differential equations. Our search-based symbolic regression approach can automatically detect symmetries, that are difficult to find for traditional computer algebra systems. As a consequence, the search-based symbolic regression approach enlarges the space of ordinary differential equations, especially nonlinear differential equations, whose analytical solution can be approached using symmetry finding methods.

Acknowledgements

This work was supported by the Carl Zeiss Stiftung within the project "Interactive Inference".

References

Baumann, G. 2002. Symmetry Analysis of Differential Equations Using MATHLIE. *Journal of Mathematical Sciences*, 108(6): 1052–1069.

Bluman, G. W.; and Kumei, S. 1989. Symmetries and Differential Equations, volume 81 of. *Applied Mathematical Sciences Series*.

Bondesan, R.; and Lamacraft, A. 2019. Learning Symmetries of Classical Integrable Systems.

Cheb-Terrab, E.; and von Bülow, K. 1995. A computational approach for the analytical solving of partial differential equations. *Computer Physics Communications*, 90(1): 102–116.

Cheb-Terrab, E. S.; Duarte, L. G. S.; and da Mota, L. A. C. P. 1997. Computer Algebra Solving of First Order ODEs Using Symmetry Methods. *Computer Physics Communications*, 101(3): 254–268.

Cheb-Terrab, E. S.; Duarte, L. G. S.; and da Mota, L. A. C. P. 1998. Computer Algebra Solving of Second Order ODEs Using Symmetry Methods. *Computer Physics Communications*, 108(1): 90–114.

Dimas, S.; and Tsoubelis, D. 2005. SYM: A new symmetry-finding package for Mathematica. *Group Analysis of Differential Equations*, 64–70.

Dormand, J.; and Prince, P. 1980. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1): 19–26.

Guimerà, R.; Reichardt, I.; Aguilar-Mogas, A.; Massucci, F. A.; Miranda, M.; Pallarès, J.; and Sales-Pardo, M. 2020. A Bayesian machine scientist to aid in the solution of challenging scientific problems. *Science Advances*, 6(5): eaav6971.

Hirsch, M.; and Smale, S. 1974. *Differential equations, dynamical systems, and linear algebra*. Number 60 in Pure and applied mathematics. San Diego [u.a.]: Acad. Press.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press. Second edition, 1992.

Jin, Y.; Fu, W.; Kang, J.; Guo, J.; and Guo, J. 2020. Bayesian Symbolic Regression. arXiv:1910.08892.

Kahlmeyer, P.; Giesen, J.; Habeck, M.; and Voigt, H. 2024. Scaling Up Unbiased Search-based Symbolic Regression. In Larson, K., ed., *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, 4264–4272. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Kamienny, P.-A.; d'Ascoli, S.; Lample, G.; and Charton, F. 2022. End-to-end Symbolic Regression with Transformers. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Kommenda, M.; Burlacu, B.; Kronberger, G.; and Affenzeller, M. 2020. Parameter identification for symbolic regression using nonlinear least squares. *Genetic Programming and Evolvable Machines*, 21(3): 471–501.

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

La Cava, W.; Spector, L.; and Danai, K. 2016. Epsilon-Lexicase Selection for Regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, 741–748. New York, NY, USA: Association for Computing Machinery. ISBN 9781450342063.

Landajuela, M.; Lee, C. S.; Yang, J.; Glatt, R.; Santiago, C. P.; Aravena, I.; Mundhenk, T.; Mulcahy, G.; and Petersen, B. K. 2022. A Unified Framework for Deep Symbolic Regression. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 33985–33998. Curran Associates, Inc.

Lie, S. 1888. *Theorie der Transformationsgruppen Abschn. 1*. Teubner.

Liu, Z.; Madhavan, V.; and Tegmark, M. 2022. AI Poincar\'{e} 2.0: Machine Learning Conservation Laws from Differential Equations. *arXiv preprint arXiv:2203.12610*.

Liu, Z.; and Tegmark, M. 2021. Machine Learning Conservation Laws from Trajectories. *Physical Review Letters*, 126(18): 180604.

Liu, Z.; and Tegmark, M. 2022. Machine-Learning Hidden Symmetries. *Physical Review Letters*, 128(18): 180201.

Lorenz, H. 1993. *Nonlinear Dynamical Economics and Chaotic Motion*. Lecture notes in economics and mathematical systems. Springer-Verlag. ISBN 9783540568810.

Maplesoft, a division of Waterloo Maple Inc. 2019. Maple.

Meurer, A.; Smith, C. P.; Paprocki, M.; Certík, O.; Kirpichev, S. B.; Rocklin, M.; Kumar, A.; Ivanov, S.; Moore, J. K.; Singh, S.; Rathnayake, T.; Vig, S.; Granger, B. E.; Muller, R. P.; Bonazzi, F.; Gupta, H.; Vats, S.; Johansson, F.; Pedregosa, F.; Curry, M. J.; Terrel, A. R.; Roucka, S.; Saboo, A.; Fernando, I.; Kulal, S.; Cimrman, R.; and Scopatz, A. M. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3: e103.

Mundhenk, T. N.; Landajuela, M.; Glatt, R.; Santiago, C. P.; faissol, D.; and Petersen, B. K. 2021. Symbolic Regression via Deep Reinforcement Learning Enhanced Genetic Programming Seeding. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Noether, E. 1918. Invariante Variationsprobleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1918: 235–257.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Petersen, B. K.; Larma, M. L.; Mundhenk, T. N.; Santiago, C. P.; Kim, S. K.; and Kim, J. T. 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*.

Richardson, M. J.; and Dale. 2014. *Complex Dynamical Systems in Social and Personality Psychology: Theory, Modeling, and Analysis*. Cambridge University Press.

Schmidt, M.; and Lipson, H. 2009. Distilling Free-Form Natural Laws from Experimental Data. *Science*, 324(5923): 81–85.

Stephani, H. 1993. Differentialgleichungen. *Symmetrien und Lösungsmethoden. Heidelberg: Spektrum*, 48.

Stephens, T. 2016. Genetic Programming in Python, with a scikit-learn inspired API: gplearn.

Strogatz, S. H. 2000. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Westview Press.

Tenenbaum, M.; and Pollard, H. 1985. *Ordinary Differential Equations*. Dover Books on Mathematics. Dover Publications. ISBN 9780486649405.

Udrescu, S.-M.; and Tegmark, M. 2020. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16): eaay2631.

Virgolin, M.; Alderliesten, T.; Witteveen, C.; and Bosman, P. A. 2021. Improving model-based genetic programming for symbolic regression of small expressions. *Evolutionary computation*, 29(2): 211–237.

Wolfram Research Inc. 2024. Mathematica, Version 14.0.