Usability of Symbolic Regression for Hybrid System Identification -System Classes and Parameters

Swantje Plambeck, Maximilian Schmidt, Audine Subias, Louise Travé-Massuyès, Goerschwin Fey

Usability of Symbolic Regression for Hybrid System Identification – System Classes and Parameters

Swantje Plambeck
Institute of Embedded Systems, Hamburg University of Technology, Germany

Maximilian Schmidt
Institute of Embedded Systems, Hamburg University of Technology, Germany

Audine Subias
LAAS-CNRS, Université de Toulouse, INSA, France

Louise Travé-Massuyès
LAAS-CNRS, Université de Toulouse, CNRS, France

Goerschwin Fey
Institute of Embedded Systems, Hamburg University of Technology, Germany

Abstract

Hybrid systems, which combine both continuous and discrete behavior, are used in many fields, including robotics, biological systems, and control systems. However, due to their complexity, finding an accurate model is a challenge. This paper discusses the usage of symbolic regression to learn hybrid systems from data and specifically analyses learning parameters for a recent algorithm. Symbolic regression is a powerful tool that can automatically discover accurate and interpretable mathematical models in the form of symbolic expressions.

Models generated by symbolic regression are a valuable tool for system identification and diagnosis, e.g., to predict future system behavior or detect anomalies. A major opportunity of our approach is the ability to detect transitions between different continuous behaviors of a system directly based on the dynamics. From a diagnosis perspective, this can advantageously be used to detect the system entering fault modes and identify their models. This paper presents a parameter study for a symbolic regression based identification algorithm.

2012 ACM Subject Classification Computer systems organization → Embedded and cyber-physical systems; Computing methodologies → Symbolic and algebraic algorithms; Computing methodologies → Learning paradigms; Computing methodologies → Modeling methodologies

Keywords and phrases Hybrid Systems, Symbolic Regression, System Identification

Digital Object Identifier 10.4230/OASIcs.DX.2024.30

Category Short Paper

Supplementary Material *Software (Source Code)*: `https://github.com/TUHH-IES/SymbolicRegression4HA`

Acknowledgements This work was supported by a fellowship of the German Academic Exchange Service (DAAD) and the ECIU Universities. The research is partially funded by the BMBF project AGenC no. 01IS22047A. It is also supported by ANITI through the French "Investing for the Future – P3IA" program under the Grant agreement n°ANR-19-P3IA-0004. Furthermore, we would like to thank Nicola Zaupa and Luca Zaccarian from LAAS CNRS for their support and comments on the power converter example.

1 Introduction

Hybrid systems are abstract models of systems that exhibit both continuous and discrete behavior. For this, hybrid systems have a finite number of modes, each representing a specific dynamic behavior of the system. They are used to model a wide range of systems, including

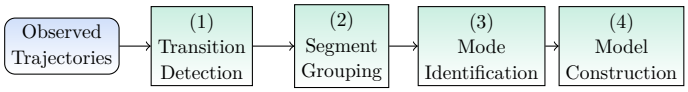


Figure 1 Steps of Hybrid System Identification [22]

cyber-physical systems or manufacturing systems, and systems with normal and faulty modes [2].Due to their inherent combination of continuous and discrete behavior, the identification of hybrid systems is a challenging task. Nevertheless, accurate abstract models are essential for verification, diagnosis, and debugging of these systems.

We recently proposed a novel approach for automatic identification of hybrid system modes from data using Symbolic Regression (SR) [23]. Like most methods for hybrid system identification, we use a general procedure consisting of four steps as shown in Fig. 1. We use SR for the steps (1) to (3) of the identification process. In step (1), we detect transitions between different modes of a hybrid system. Step (2) and (3) are combined in one algorithm. SR is able to identify complex behavior from data [26, 15].Here, the particular opportunity of SR is the ability to detect transitions between different continuous behavior of a system directly based on the dynamics. This goes beyond existing identification strategies, which use similarities of observations to separate and group different modes of a hybrid system in observed data [3, 29, 19]. The goal of this paper is to gain a deeper insight into the capabilities and challenges of using SR for hybrid system identification, leveraging new possibilities for diagnosis on symbolic models in the future.

Like other learning algorithms for system identification, hybrid system identification with SR requires selecting parameters for learning. In this paper, we analyze the impact of the parameters on the identification of hybrid systems and discuss the trade-offs between runtime, accuracy and descriptiveness of the identified models. Finally, we evaluate the approach on a set of selected examples.

In Plambeck et al. [23], we introduced the basic idea of the identification algorithm. In addition to that, we now have the following contributions 1) a discussion of SR in the context of hybrid system identification, 2) the identification of relevant parameters for the identification of hybrid systems using SR, and 3) a structured analysis of learning parameters considering multiple example systems including a physical simulation of a two-state power converter, a two tank system, and a static electrical circuit with multiple power sources.

The paper is structured as follows: in Section 2, we review related work on system identification, specifically for hybrid systems and SR. In Section 3, we introduce the necessary formal definitions. In Section 4, we revisit the algorithm presented in [23] and discuss the impact of parameters on the identification of hybrid systems. In Section 5, we perform an intensive parameter study. The algorithm and the parameter study are open source and available at [21]. Finally, in Section 6, we conclude the paper.

2 Related Work

Symbolic Regression (SR) is a method for regression and system identification that aims to find a symbolic expression that matches a given data set. Contrary to traditional regression methods, SR is not restricted to a specific set of functions or structure of an expression like, e.g., polynomial regression, but uses a set of basic operators to construct complex expressions freely. The development of SR has been supported by the advancement of genetic programming, which is commonly used to implement SR algorithms [15]. In addition to genetic programming, there exist further approaches for SR, e.g., using deep reinforcement

learning [20] or lattices [6].

Genetic programming methods like SR have been applied to a wide range of problems, including the identification of physical concepts from data [26], hybrid dynamical systems [18] mining the expression of diagnosis indicators [11].Thus, SR is one of several methods for system identification together with a broad range of other identification methods such as linear or nonlinear regression, neural networks, or kernel-based methods. Schoukens et al. [27] provides a comprehensive overview on these methods. In Koza [14], general hypotheses about the capabilities and convergence of SR are already discussed. Other and recent works investigate the influence of specific parameters such as the population size and number of generations on the performance of SR [16].

Hybrid systems, exemplified by hybrid automata, are the focus of this paper. Existing approaches for identification of hybrid systems from data, present several different strategies. Among them are clustering methods [3], machine learning with neural networks [19], and linear inequalities [29]. The general procedure, separating the process of learning in multiple steps as shown in Fig. 1, is similar in all of these approaches. Nevertheless, the detection of transitions in these methods is based on the similarity of signals, e.g., based on windows of the observations [29] or on distances between signals [3] or in the frequency domain [19]. Here, our approach using SR offers the opportunity to detect the decision points for transitions directly based on (an estimate of) the dynamics of the observed data.

3 Preliminaries

In this section, we introduce the basics of Symbolic Regression (SR) and formally define hybrid systems and system observations. We follow the presentation given in [23].

3.1 Symbolic Regression

SR is a machine learning technique that aims to find a mathematical expression describing the relationship between multiple input variables and a single output variable, i.e., SR searches a function r with $o = r(\mathbf{i})$, where $\mathbf{i}$ are the input variables and o is the output variable. The function r is represented as a mathematical expression in terms of elementary functions and operators [14]. The search for the best symbolic expression is guided by a fitness metric that evaluates the quality of candidate expressions on a data set. Learning algorithms typically represent the expression as a tree structure, where nodes represent basic operators and leaves represent variables or constants.

▶ **Definition 1** (SR Search & Solution Space). *The search space $\mathcal{E}$ is the set of possible tree structures defined by a set of operators as well as a set of variables and constants. The solution space $\mathcal{S} \subseteq \mathcal{E}$ is the set of expressions, i.e., tree structures, whose fitness is above a predefined threshold.*

The search space is often constrained by limiting the maximum depth of the tree, the number of nodes, or the set of operators [9, 20].

In the scope of this work, we use the framework PySR for SR, which uses Genetic Programming (GP) [14, 9]. Using GP, the search for the best symbolic expression is performed by evolving a population of candidate expressions over so-called *iterations* as shown in Fig. 2. The population is initialized with random expressions. In each iteration, the population is evaluated using a fitness function. New solutions are generated by applying genetic operators, such as mutation and crossover, to the best candidates. The process is repeated until a stopping criterion is reached, e.g., predefined number of iterations.

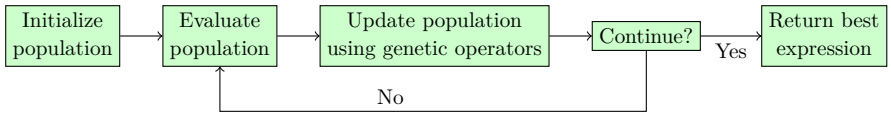


Figure 2 Genetic Programming Algorithm for Symbolic Regression

Nevertheless, for GP-based algorithms, there is no guarantee that the learned expression is the exact one, because the underlying search process is randomized, there exist mathematically or logically equivalent solutions, and learning data might be noisy. Thus, SR is most useful in domains where close approximations to an explicit solution exist and are useful [24].

SR has shown to rapidly approach the close neighborhood of the optimal solution, struggling only in converging to a precise result [14]. Thus, we can expect that the learned expressions represent the most dominant dynamics of the system and are compact. Several studies show that the performance of SR improves by using a suitable parametrization. The population size and the number of iterations are usually considered the most crucial parameters – usually a larger population and number of iterations leads to higher accuracy [10, 17, 16].

Another known problem in the evolution of learned expressions in SR is *bloat*. Bloat describes the growth of the learned expression without a significant improvement in the fitness. Bloat is often addressed by using a parsimony pressure [24]. In this case, SR solves a multi-objective optimization problem, where the fitness of the expression is not only determined by the quality of the approximation but also by the size of the expression. The *parsimony coefficient* regulates the trade-off between the two objectives.

3.2 Hybrid Systems

The literature identifies types of dynamical systems, which are hybrid, i.e., involve discrete and continuous dynamics. These types of systems illustrate different levels of model expressiveness needed to represent the dynamics.

- Jump linear systems are described by stochastic processes [8]. They are often represented by a Markov chain where modes are associated with different linear systems.
- Piecewise affine systems for which flow functions are affine functions [7] and the state space is partitioned into polyhedral regions.
- Switched systems for which the flow functions are general continuous functions.
- Mixed logical dynamical systems include logical variables to model discrete events or conditions in the system description [28].
- Projected dynamical systems extend the expressiveness of flow functions to non-linear expressions [13].

Here, building upon Branicky [5], we use a generalizing definition of hybrid systems which incorporates all the known system types listed above.

▶ **Definition 2** (Hybrid System [5]). *A Hybrid System is defined by a 6-tuple* $(X, Q, \mathcal{F}, \mathcal{T}, \Sigma, R)$ *where*

- $X = \{x_1, x_2, ..., x_n\} = I \cup O \cup S$ *is the set of system variables which consist of input variables* I*, output* O *and state variables* S*. Derivatives of variables may form individual variables in* X*.*
- Q *is the set of modes.*

- *$\mathcal{F}$ is the set of flow functions. Every flow function $f_q \in \mathcal{F}$ defines the change of the state variables S as well as the current output variables O over the continuous time t within the mode q based on the current values of state variables and the external inputs I, i.e., $[O(t), \dot{S}(t)] = f_q(S(t), I(t), t)$.*
- *$\mathcal{T} : Q \times \Sigma \to Q$ defines transitions between modes Q. A transition is triggered if the corresponding event $\sigma \in \Sigma$ is active.*
- *Σ is a set of events leading to transitions between modes. Each event is guarded by a set of conditions on the variables in X. A transition is triggered if the conditions are met.*
- *R is a reset relation $R : Q \times \Sigma \times X \to X$ capturing discontinuous changes of the internal variables.*

A hybrid system, according to this definition, combines discrete and continuous behavior. Discrete behavior is captured by the discrete modes Q and transitions $\mathcal{T}$, while the flow functions $\mathcal{F}$ describe the continuous dynamics. Transitions are usually triggered by conditions on the variables or discrete control signals for mode transitions. In the scope of this paper, the goal is to identify a model of a real system according to Definition 2. We focus on the identification of modes and flow functions while excluding the construction of conditions. This usually implies that transition conditions are defined by external control signals. Within this scope, *dynamics* define the physical behavior of the real system. In the abstraction given by the model, i.e., the hybrid system, the dynamics are represented by the flow functions of the *modes*. Finally, observations of dynamics, i.e., changes in the values of the variables, are considered as *trajectories* defined as an observation in form of a multi-dimensional time series of the variables $X = \{x_0, x_1, \ldots, x_n\}$.

4 Identification of Hybrid Modes

In this section, we first revisit the approach presented in Plambeck et al. [23] and identify the parameters which are used in the parameter study.

4.1 Overview & System Properties

Our approach might model all of the system types encompassed by Definition 2, while we focus here on Jump Linear Systems, Piecewise Affine Systems, and Switched Systems, i.e., systems with continuous flow functions. The most characteristic property of hybrid systems is the combination of *continuous* dynamics (defined by the flow functions $\mathcal{F}$) with *discrete* modes (Q). To learn both parts, identification of hybrid systems includes multiple subproblems as shown in Figure 1 and as similarly introduced in Saberi et al. [25]:

1. detection of discrete transitions between dynamics, separating the trajectories into segments,
2. grouping of segments with identical dynamics, forming discrete modes,
3. identification of the continuous dynamics for each mode, i.e., the flow functions of $\mathcal{F}$,
4. model construction, i.e., accumulation of the results in a single hybrid system.

The modeling strategy covers steps **1.** to **3.** in which steps **2.** and **3.** are combined, both based on SR. Our approach, hence, involves two algorithms: one algorithm to detect the transitions between modes (*segmentation*) and, further on, a second algorithm to group and identify the flow functions of modes using the segmented trajectories (*grouping*). We assume a positive residence time in each of the hybrid modes. This residence time should provide sufficiently many data samples such that the original dynamics can be reconstructed.

4.2 Identification Algorithm

For the segmentation step, we begin with a small window of observed data, learning a symbolic expression. The window is gradually enlarged until the expression's fitness declines, indicating a *decision point* where a mode transition occurs. With each window increase, the expression adapts incrementally to capture changes in the dynamics. Pseudocode for segmentation is shown in Algorithm 1.

In the subsequent grouping step, SR is reused to learn expressions on unions of the previously detected segments. When the loss of combined segments decreases compared to individual segments, they are grouped, identifying the mode. By this segments with the same dynamics, describable by the same flow functions, are grouped. Pseudocode for grouping is in Algorithm 2.

Data: trajectory
Result: $\mathcal{T}, expressions$
1 $i_{start} \leftarrow 0$; $i_{end} \leftarrow l_{init}$; $n \leftarrow n_{init}$;
2 **while** $i_{end} < len(trajectory)$ **do**
3 **while** *segmentationCriterion fulfilled* **do**
4 window $\leftarrow$ trajectory$[i_{start}, \min(i_{end}, len(\text{trajectory}))]$;
5 learnExpression(window,n);
6 $i_{end} \leftarrow i_{end} + l_{step}$;
7 $n \leftarrow n_{update}$;
8 **end**
9 $\mathcal{T} \leftarrow \mathcal{T} \cup \{\text{window}[0, \text{end} - l_{step}]\}$;
10 $i_{start} \leftarrow i_{end} - l_{step}$; $i_{end} \leftarrow i_{start} + l_{init}$; $n \leftarrow n_{init}$;
11 resetSR;
12 **end**

Algorithm 1 Detection of Mode Transitions: given an observed trajectory of the system, we process over this trajectory using a window (Line 4). Initially, this window covers a fixed initial length at the beginning of the trajectory. As long as the segmentation criterion is fulfilled, the window is extended to the right (Line 6). When the segmentation criterion is no longer met, a mode transition is detected, and the window is stored as a segment in the set $\mathcal{T}$. In the inner loop, the symbolic expression is learned incrementally, i.e., n_{update}-many iterations of the SR are performed [23].

From this review of the algorithms, we find the following set of parameters as given in Table 1. The segmentation and grouping criteria as given in Plambeck et al. [23] are used.

Symbol	Occurrence	Description
l_{init}	Segmentation	initial window size when learning an expression
l_{step}	Segmentation	step-width for extending the window
n_{init}	Segmentation	number of iterations of SR when learning an expression
n_{update}	Segmentation	number of iterations of SR for updating the expression on an extension
τ	Segmentation	threshold for the segmentation criterion
φ	Grouping	relaxation parameter for the grouping criterion
n_g	Grouping	number of iterations of SR when learning on grouped data
ρ_s, ρ_g	General SR	Parsimony coefficient (length-accuracy trade-off) for segmentation and grouping
p_s, p_g	General SR	Population size for segmentation and grouping

Table 1 Parameters of the Learning Process.

Data: $\mathcal{T}$
Result: G, expressions
1 $G \leftarrow \{S[0]\}$;
2 **for** $s \in \mathcal{T}$ **do**
3 groupFound $\leftarrow$ False;
4 **for** $g \in G$ **do**
5 exp, fit $\leftarrow$ learnExpression$(s \cup g, n)$;
6 **if** *groupingCriterion fulfilled* **then**
7 $g \leftarrow s \cup g$;
8 expressions$[g] \leftarrow$ exp; groupFound $\leftarrow$ True; break;
9 **end**
10 **end**
11 **if** *not groupFound* **then**
12 G.append(s);
13 **end**
14 **end**

Algorithm 2 Grouping of Modes: the input to the grouping is the set of detected segments $\mathcal{T}$. The first segment is a first candidate group as stated in Line 1. Afterward, we iterate for every segment in $\mathcal{T}$ in Line 2 over all known groups in Line 4. The current segment and the current group are combined to one data set and an expression is learned (see Line 5). If the loss of the learned expression is small, the current segment is included in the current group (see Lines 6 and 7). If no matching group is found for the current segment, the segment forms a new group as stated in Line 11 [23].

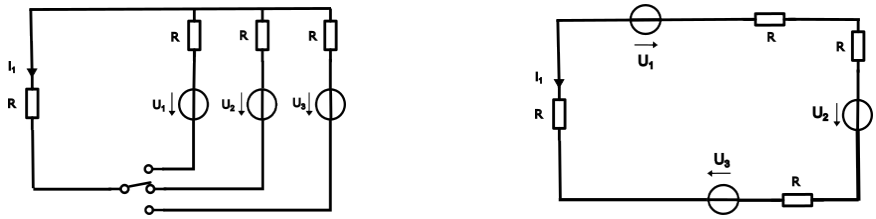
5 Experiments & Parameter Study

In this section, we evaluate the feasibility and capability of SR for hybrid system identification using our proposed algorithm. Thus, the evaluation scenario focuses the learning step. We use a single trace observed on an example system, for which we know both, the ground truth decision points of the trace and the ground truth expressions of the flow functions of the systems. Our analysis focuses the accuracy, that the SR-based learning is able to achieve compared on the learning trace and with respect to the ground truth information. The evaluation of the learned model on an evaluation data set is out of scope here.

In the following, we first showcase the usability on two simple examples which we will use for a further discussion on system and model properties later on. Afterwards, we present an intensive parameter study for the presented algorithm on two real-world examples. This study provides additional insights in the usability of SR for the identification of hybrid systems and aligns with the previous introduction of SR. Furthermore, the study provides indications on how to choose parameters for to be learned systems.

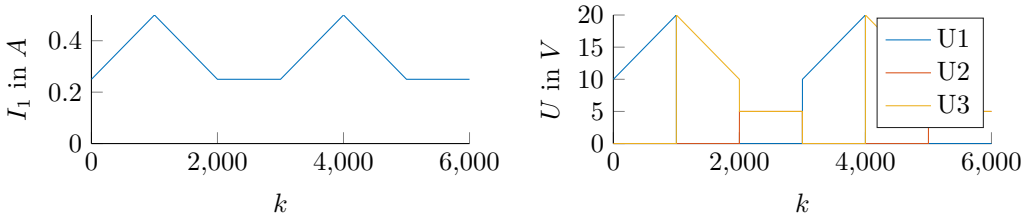
5.1 Usability of SR-Based Hybrid System Identification on Simple Examples

We consider two versions of a passive electrical circuit, which are shown in Figure 3. In both examples, the goal is to learn the flow functions $o(t) = f_q(\mathbf{i}(\mathbf{t}))$, where the inputs $\mathbf{i} = [U_1, U_2, U_3]$ are the voltage levels of the sources and the output $o = I_1$ is the main current in the circuit. The first circuit in Fig. 3a is described by the equation. Depending on the position of the switch, one of the three sources is selected. This is a simple example of a hybrid system with three modes.



(a) Version 1 – Switched sources **(b)** Version 2 – Connected sources

Figure 3 Passive Electrical Circuits



(a) Output current I_1 **(b)** Input voltages U_1, U_2, U_3

Figure 4 Current and Voltage of Circuit 2

$$I_1 = \begin{cases} \frac{U_1}{2 \cdot R}, & \text{switch in 1st position} \\ \frac{U_2}{2 \cdot R}, & \text{switch in 2nd position} \\ \frac{U_3}{2 \cdot R}, & \text{switch in 3rd position} \end{cases} . \tag{1}$$

The second circuit shown in Fig. 3b also contains three sources, but here all of them are connected within the circuit. Fig. 4 shows the output current I_1 and the three voltages over the discrete sampling points k. From visual inspection of I_1, we might assume three operational modes of the system. Nevertheless, the different appearances of I_1 solely result from changes in the input signals. In fact, the system is completely described by the equation

$$I_1 = \frac{U_1 + U_2 + U_3}{4 \cdot R}. \tag{2}$$

Even though this circuit does not show a hybrid behavior, we consider this as an interesting example, as one might assume different modes from visual inspection. Also, this example shows that whether multiple modes are needed or not can be ambiguous as both of the circuits might lead to identical observations. There are multiple approaches that could resolve this issue. One possibility is to choose the most compact representation.

For SR, both circuits use the same set of basic operators which contains addition, subtraction, multiplication, and division operators. The three voltages U_1, U_2, U_3 and the values of the resistance R are given as variables for learning an expression for I_1. Both circuits use $R = 10\Omega$ for all resistors. The parameters for learning as listed in Table 1 are set to $l_{init} = 200$, $l_{step} = 100$, $n_{init} = 20$, $n_{update} = 5$, $l_{hist} = 1$, $\tau = 1 \cdot 10^{-7}$, $n_{group} = 20$, and $\varphi = 1.5$.

Table 2 shows the results of the identification process for the two circuits. For the first circuit all five transitions are detected correctly and no false positive, i.e., additional transitions are detected. For the second circuit, no transitions are detected, as the system is

System	$\|S\|$	TP	FP	$\|G\|$	Learned Expressions
Circuit 1	5	100	0	3	Group 1 – $U_1/(2 \cdot R)$ Group 2 – $U_2/(2 \cdot R)$ Group 3 – $U_3/(2 \cdot R)$
Circuit 2	0	-	-	1	Group 1 – $(U_1 + U_2 + U_3)/(4 \cdot R)$

Table 2 Identification Results for the Simple Examples, True Positives (TP) and False Positives (FP) are given in percent, $|S|$ is the number of detected transitions, $|G|$ is the number of detected groups

actually not hybrid. This shows that the approach is able to identify identical dynamics even though the visual inspection of the trajectories may suggest different modes as discussed for Figure 4.

The learned expressions for both circuits are equivalent to the ground truth expressions. Thus, leading to a mean-square error loss of zero for the predicted trajectories. The results show that the approach is able to identify the structure of the hybrid systems from data perfectly for simple examples.

5.2 Parameter Study

Having shown the usability of the SR-based algorithm on simple examples, we now present a parameter study on two real-world examples introduced in the following. The first example is the two tank system, which is a known benchmark system for hybrid systems. The second example is a power converter, which is a real-world system with a complex behavior.

5.2.1 Two Tank System

The two tank system [4] is a benchmark system for hybrid systems, which consists of two tanks with a pump pumping water into the first tank. A valve V_b regulates whether water flows from the first to the second tank. The system is described by the following differential equation:

$$\dot{h}_1 = \begin{cases} \frac{Q_p - C_{vb} \cdot \sqrt{h_1 - h_2}}{A}, & \text{if } h_1 > h_2, V_b \text{ open} \\ \frac{Q_p + C_{vb} \cdot \sqrt{h_2 - h_1}}{A}, & \text{if } h_1 \leq h_2, V_b \text{ open} \\ \frac{Q_p}{A}, & \text{if } V_b \text{ closed,} \end{cases} \tag{3}$$

where h_1 and h_2 are the heights of the water in the first and second tank, respectively, $\dot{h}_1$ is the derivative, i.e., the change in the water level, of the first tank. Q_p is the flow rate of the pump, C_{vb} is the valve conductance, and A is the cross-sectional area of the first tank. The simulation involves noise and a realistic controller which applies a zero-order hold to the observed height of the tank. The noise is additive white noise with a maximum amplitude of 10^{-6} for all sensors.

In the modeling process, we consider two different versions of the two tank:

- Version 1 (with substitution): the pre-calculated term $\sqrt{|h_1 - h_2|}$, is given as an additional variable. The operators for learning involve addition, subtraction, multiplication, and division.
- Version 2 (without substitution): the pre-calculated term is not given as a variable. The operators for learning involve addition, subtraction, multiplication, division, and, additionally the square-root.

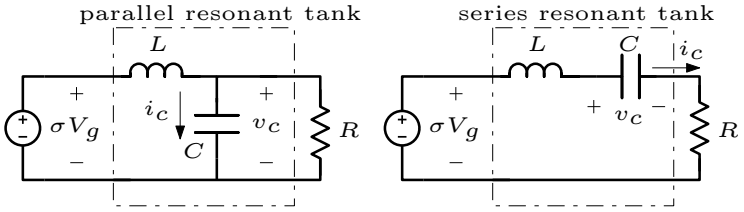


Figure 5 Series and Parallel Representation of the Power Converter [30]

For both versions, variables for SR are the inflow Q_p and the height of the two tanks h_1 and h_2 as well as the constants C_b and A. The goal is to learn the derivative $\dot{h}_1$, i.e., the flow function which defines the state change $\dot{s}(t) = f_q(s(t), \mathbf{i}(t))$, with $s = h_1$ and $\mathbf{i} = [h_2, Q_p, C_b, A]$.

5.2.2 Power Converter

The power converter [30] is a real-world system that has a controlled input voltage $v_s = \sigma V_g$ where V_g is a constant input and σ is a switching variable which can be either 1 or -1. In addition to the voltage source, the circuit consists of a capacitor C, an inductance L, and a resistor R. The circuit can be either a parallel or a series configuration, as shown in Figure 5. As presented in [30], we use a transformed coordinate system to describe both configurations with the same equations. The system is described by the following differential equation:

$$\dot{w} = \begin{pmatrix} 0 & \alpha \\ -\alpha & -\beta \end{pmatrix} w + \begin{pmatrix} 0 \\ \alpha \end{pmatrix} \sigma, \tag{4}$$

where $\alpha = \frac{1}{\sqrt{LC}}$ and $\beta = \frac{1}{RC}$ (parallel case) or $\beta = \frac{R}{L}$ (serial case). The transformed coordinates $w = [w_1, w_2]^T$ are constructed from the quantities in Figure 5 as

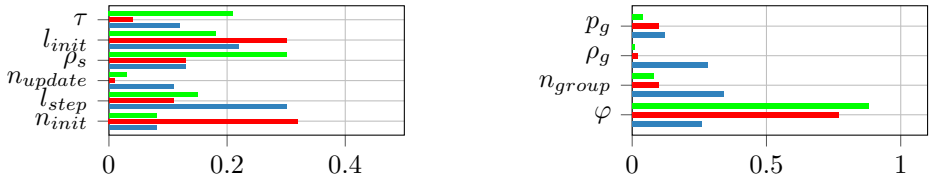
$$w_1 = \frac{v_c}{V_g} \quad \text{and} \quad w_2 = \frac{1}{V_g}\sqrt{\frac{L}{C}} \cdot i_c.$$

Our goal is to model the state variable w_2. The inductance, capacity, and resistor have the values $R = 400\Omega, L = 8\mu H, C = 10.5nF$ and $V_g = 20V$.

For SR, the power converter uses addition, subtraction, multiplication, and division as well as the square-root as basic operators. Variables for learning are w_1, w_2, and the continuous time t of a sampling point. Constants are not given as variables, but are estimated by the learner. The goal is to learn an expression for $\dot{w}_2$, where the derivation is numerically performed. Thus, we learn the flow function which defines the state change $\dot{s}(t) = f_q(s(t), \mathbf{i}(t), t)$, with $s = w_2$ and $\mathbf{i} = [w_1]$.

5.2.3 Experimental Results

For both systems, we analyze the parameters l_{init}, l_{step}, n_{init}, n_{update}, and τ for the segmentation step and n_{group}, φ, and p_g for the grouping step. Additionally, the parsimony coefficients ρ_s and ρ_g for SR during the segmentation and grouping step, respectively, are analyzed. The parameter study is executed with the hyperparameter optimization library Optuna [1] with at least 100 trials for every study. The optimization uses an *objective function*. Parameter importances are found with the fANOVA approach [12] of Optuna.

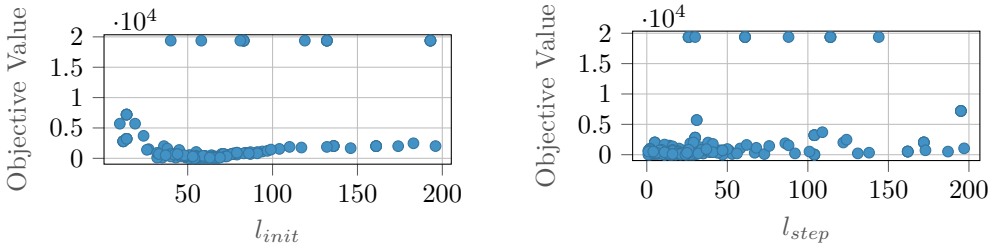


(a) Segmentation **(b)** Grouping

Figure 6 Parameter importances, Two Tank System with Substitution (green), Two Tank System without Substitution (red), Power Converter (blue)

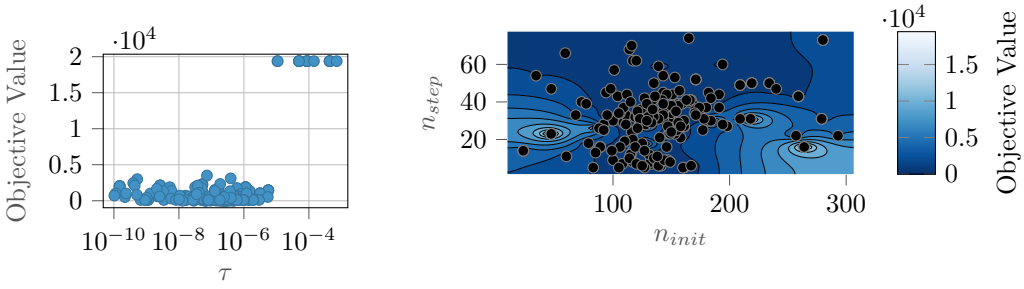
More information on optuna, the objective functions and fANOVA can be found in the project's repository [21].

The parameter importances, as calculated with Optuna using the fANOVA approach [12], show that for most examples, the initial window width l_{init} as well as the length for extending the window l_{step} are relevant. Figure 7 shows these two parameters against the objective value. Note, that this and the following plots always show all runs, i.e., also other than the presented parameters are varied over the runs. We observe, that l_{init} against the objective value has a sweet spot around 60, which is about the mean number of samples, that the system stays within a mode. These observations are intuitive as the best initial window size would be the one that captures exactly one occurrence of a mode. This implies that prior knowledge or a good assumption on the expected time spent in a mode improves the model learning procedure. For l_{step}, the plot indicates that smaller values usually lead to smaller objective values. Thus, a small step size when increasing the window identifies the decision points more accurately. Still, a smaller step width leads to longer runtime.



(a) Start width l_{init} against objective value **(b)** Step width l_{step} against objective value

Figure 7 Segmentation: parameters against objective value for Two Tank without substitution



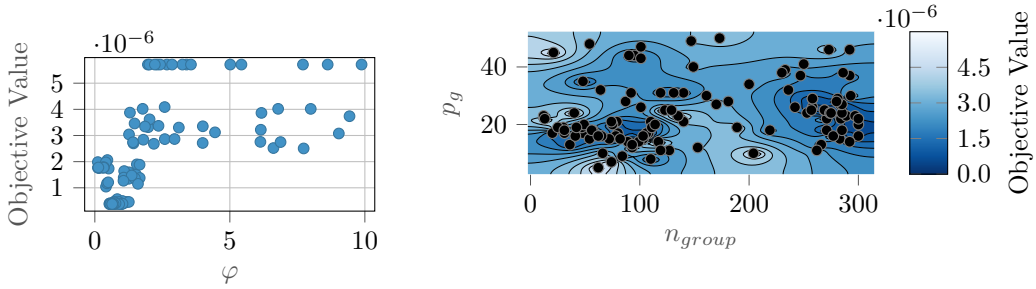
(a) Saturation τ against objective value **(b)** Contour of objective value against n_{init}, n_{step}

Figure 8 Segmentation: parameters against objective value for Two Tank with substitution

The number of generations n_{init} and n_{step} are also relevant parameters. Figure ref-fig:twotanksubsegiter shows the two parameters in a contour plot of the objective value, where dark colors indicate good, i.e., small objective values. For this example, the number of initial iterations n_{init} is best around 100 to 150. This number of iterations allows to pre-learn the dynamics without overfitting such that the inclusion of new data is possible on window extension. The number of generations for updating the population n_{step} has low influence on the objective value. To reduce runtime, we could, thus, choose a small value for n_{step}.

The saturation threshold for the segmentation criterion τ shows a clear impact as shown in Figure 8a. As introduced earlier, this threshold assures that the extension of the window is continued as long as the loss stays below the threshold. Thus, the value of τ has to be below a certain level to correctly identify decision points.

For the grouping step, the grouping factor φ is the most important parameter, because the choice of this parameter clearly separates experiments with a low and a high objective value. We also have a clear dependency of the objective in other parameters such as the number of iterations n_{group}. Figure 9b shows the number of iterations and the size of the population against the objective value. A larger number of iterations is preferable, because a higher number of generations allows for a better exploration of the solution space. The size of the population has less influence on the objective value. Figure 9a shows the factor φ for the grouping criterion against the objective value. Here, we find a clear minimum around $\varphi = 1$, i.e., where we require a strict decrease in the loss when adding a segment to a group.



(a) Factor φ against objective value

(b) Contour plot of objective value against n_{group},p_g

Figure 9 Grouping: parameters against objective value for Converter

5.3 Accuracy of Predicted Trajectories

In the last step, we showcase the accuracy of the learned models with good parametrization with respect to our objective functions. The results are shown in Table 3.

We observe that the learned expressions are not identical to the ground truth expressions, but represent the most dominant behavior. This aligns with the known property of SR which tends to converge to solutions close to the optimum, but matching the exact correct expression is difficult especially if the solution space is large, i.e., the expression to be learned is complex. A deeper discussion and comparison of the predicted and the original trajectories can be found in [23].

6 Conclusion

In this paper, we provide a deep discussion of symbolic regression for hybrid system identification. Revisiting a proposed method for hybrid system identification with symbolic

System	$\|S\|$	TP	FP	$\|G\|$	Learned Expressions		Loss
Converter	4	100	0	2	Group 1	$3.37 \cdot 10^{-3} - 4.71 \cdot 10^{-3} \cdot w1$	$4.76 \cdot 10^{-7}$
					Group 2	$-6.28 \cdot 10^{-3} \cdot \sqrt{w1 + 0.396}$	
Two Tank 1	27	92.9	3.7	3	Group 1	Q_p/A	$5.26 \cdot 10^{-6}$
					Group 2	$(-Cv_{vb} \cdot \sqrt{\|h_1 - h_2\|} + Q_p)/A$	
					Group 3	$(-C_{vb} + Q_p/h_1)/A$	
Two Tank 2	28	92.9	0	3	Group 1	$Q_p/A - \sqrt{Q_p} \cdot h_1$	$4.68 \cdot 10^{-6}$
					Group 2	Q_p/A	
					Group 3	$\sqrt{C_{vb}} - A$	

Table 3 Identification Results, True Positives (TP) and False Positives (FP) are given in percent, $|S|$ is the number of detected transitions, $|G|$ is the number of detected groups

regression, separated in two algorithms, we cover three major aspects. First, we discuss known properties of symbolic regression regarding accuracy and convergence and put them in the context of hybrid system identification. Furthermore, we provide an intensive parameter study of the two identification steps. We see that a higher number of generations leads to more accurate models. Furthermore, prior knowledge on the system behavior can support the learning process. The last part of the paper is dedicated to a discussion of system types in the regime of hybrid systems and within the context of symbolic regression. We argue that the complexity in the expression of dynamics and the number of modes of a model form a solution space where large models with simple expressions form similarly accurate models as small models with complex expressions. The choice of model size and expression complexity, thus, can be seen as a design decision during model learning.

References

1 Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019. `arXiv:1907.10902`.

2 Rajeev Alur. Principles of Cyber-Physical Systems. Technical report, MIT Press, 2015.

3 Nathalie Barbosa Roa, Louise Travé-Massuyès, and Victor H. Grisales-Palacio. Dyclee: Dynamic clustering for tracking evolving environments. *Pattern Recognition*, 94:162–186, 2019.

4 B. Ould Bouamama, R. Mrani Alaoui, P. Taillibert, and M. Staroswiecki. Diagnosis of a two-tank system. Technical report, Intern Report of CHEM-project, USTL, 2001.

5 Michael S. Branicky. *Introduction to Hybrid Systems*, pages 91–116. Birkhäuser, Boston, 2005.

6 Kevin René Broløs, Meera Vieira Machado, Chris Cave, Jaan Kasak, Valdemar Stentoft-Hansen, Victor Galindo Batanero, Tom Jelen, and Casper Wilstrup. An approach to symbolic regression using feyn. *arXiv*, 2021. `arXiv:2104.05417`.

7 Frank J. Christophersen. *Piecewise Affine Systems*, pages 39–42. Springer, Berlin Heidelberg, 2007.

8 Oswaldo Luiz Valle Costa, Ricardo Paulino Marques, and Marcelo Dutra Fragoso. *Markov Jump Linear Systems*, pages 1–14. Springer, London, 2005.

9 Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression.jl, 2023. `arXiv:2305.01582`.

10 Robert Feldt and Peter Nordin. Using factorial experiments to evaluate the effect of genetic programming parameters. In *Genetic Programming*, 2000.

11 Louis Goupil, Elodie Chanthery, Louise Travé-Massuyès, and Sébastien Delautier. Tree based diagnosis enhanced with meta knowledge. In *International Workshop on Principles of Diagnosis (DX)*, 2023.

406 **12** F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter
407 importance. In *International Conference on Machine Learning (ICML)*, 2014.

408 **13** Hassan K. Khalil. *Nonlinear systems*. Pearson Education International, 3. edition, 2000.

409 **14** John R. Koza. *Genetic Programming: On the programming of computers by means of natural*
410 *selection*. MIT Press, 1994.

411 **15** Gabriel Kronberger, Lukas Kammerer, and Michael Kommenda. Identification of dynamical
412 systems using symbolic regression. *arXiv*, 2021. `arXiv:2107.06131`.

413 **16** William B. Langdon. Genetic programming convergence. In *Genetic and Evolutionary*
414 *Computation Conference Companion*, 2022.

415 **17** Thomas Loveard and Vic Ciesielski. Genetic programming for classification: An analysis of
416 convergence behaviour. *Lecture Notes in Artificial Intelligence*, 2557:309–320, 2002.

417 **18** Daniel L. Ly and Hod Lipson. Learning symbolic representations of hybrid dynamical systems.
418 *Journal of Machine Learning Research*, 13(115):3585–3618, 2012.

419 **19** Oliver Niggemann, Benno Stein, Asmir Vodencarevic, Alexander Maier, and Hans Kleine
420 Büning. Learning behavior models for hybrid timed systems. In *AAAI Conference on Artificial*
421 *Intelligence*, 2012.

422 **20** Brenden K Petersen, Mikel Landajuela Larma, Terrell N. Mundhenk, Claudio Prata Santiago,
423 Soo Kyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical
424 expressions from data via risk-seeking policy gradients. In *International Conference on Learning*
425 *Representations*, 2021.

426 **21** Swantje Plambeck. Symbolic regression 4 hybrid systems, 2024. URL: `https://github.com/`
427 `TUHH-IES/SymbolicRegression4HA`.

428 **22** Swantje Plambeck, Aaron Bracht, Nemanja Hranisavljevic, and Goerschwin Fey. Famos–
429 fast model learning for hybrid cyber-physical systems using decision trees. In *International*
430 *Conference on Hybrid Systems: Computation and Control*, 2024.

431 **23** Swantje Plambeck, Maximilian Schmidt, Audine Subias, Louise Travé-Massuyès, and Goer-
432 schwin Fey. Dynamics-based identification of hybrid systems using symbolic regression. In
433 *Software Engineering and Advanced Applications (SEAA)*, 2024.

434 **24** Riccardo Poli, William B Langdon, and Nicholas F McPhee. *A Field Guide to Genetic*
435 *Programming*, volume 10. Springer, 2008.

436 **25** Iman Saberi, Fathiyeh Faghih, and Farzad Sobhi Bavil. A passive online technique for learning
437 hybrid automata from input/output traces. *ACM Transactions on Embedded Computing*
438 *Systems*, 22(1), oct 2022.

439 **26** Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data.
440 *Science*, 324(5923):81–85, 2009.

441 **27** Johan Schoukens and Lennart Ljung. Nonlinear system identification: A user-oriented road
442 map. *IEEE Control Systems Magazine*, 39(6):28–99, 2019.

443 **28** E. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on*
444 *Automatic Control*, 26(2):346–358, 1981.

445 **29** Xiaodong Yang, Omar Ali Beg, Matthew Kenigsberg, and Taylor T. Johnson. A framework
446 for identification and validation of affine hybrid automata from input-output traces. *ACM*
447 *Transactions on Cyber-Physical Systems*, 6(2):1–24, 2022.

448 **30** Nicola Zaupa, Luis Martínez-Salamero, Carlos Olalla, and Luca Zaccarian. Hybrid control
449 of self-oscillating resonant converters. *IEEE Transactions on Control Systems Technology*,
450 31(2):881–888, 2023.