Identification of hybrid systems with dynamics-based modeling through symbolic regression

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

Identification of Hybrid Systems with Dynamics-Based Modeling through Symbolic Regression*

Swantje Plambeck[a,*], Maximilian Schmidt[a], Audine Subias[b], Louise Travé-Massuyès[b], Goerschwin Fey[a]

[a]*Hamburg University of Technology, Hamburg, Germany*
[b]*LAAS-CNRS, CNRS, INSA, University of Toulouse, Toulouse, France*

Abstract

Hybrid systems combine both continuous and discrete behavior. These systems serve as models in many fields, including control systems, robotics, and industrial processes. However, due to their complexity, finding an accurate model is a challenge. This paper presents a holistic approach to learning models of hybrid systems using symbolic regression. Our method leverages symbolic regression to automatically discover accurate and interpretable mathematical models in the form of hybrid systems from observed data. An advantage of our algorithm is that it detects transitions between different behavioral modes of a system based on the inherent dynamics.

From learned expressions for the dynamical behavior of a system, we form a hybrid system by combining the learned expressions with a decision tree determining the current behavioral mode from data. This hybrid decision tree serves regression, prediction, and further related tasks. Our results

*Corresponding author
Email address: swantje.plambeck@tuhh.de (Swantje Plambeck)

demonstrate that symbolic regression can effectively identify the underlying dynamics of a real hybrid system and predict output signals on new input data with high accuracy.
Keywords: Hybrid Systems, Symbolic Regression, Decision Trees, Model Inference, Machine Learning

---

1. Introduction

Hybrid systems are abstract models of systems that exhibit both continuous and discrete behavior. They are used to model a wide range of systems, including cyber-physical systems, manufacturing systems, and many more [1, 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, e.g., diagnosis, verification, and control. Hybrid systems operate with a finite set of modes, each representing a distinct dynamic behavior of the system. Within a mode, the continuous system dynamics are defined by flow functions over the system variables.

We propose a holistic approach for the identification of an inferable hybrid system. The identification of continuous dynamics uses Symbolic Regression (SR) based on our previous work [3]. Existing hybrid system identification methods introduce a general procedure that identifies modes and their flow functions to create a model as shown in Figure 1 [4]. In Plambeck et al. [3], novel approaches for improving the steps (a) to (c) for detecting transitions, grouping of observations with identical dynamics into modes, and identifying modes through symbolic expressions have been proposed. Here, in addition

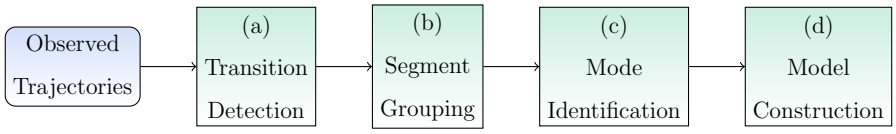


Figure 1: Steps of Hybrid System Identification

to consolidate the work in Plambeck et al. [3], we extend to a holistic hybrid system identification algorithm by adding the final model construction step. This enables the inference of the model on new unseen data. The model construction step uses decision tree learning. The idea of using decision trees for model construction has already been successfully implemented in [4]. The work in [4] aims at fast identification mechanisms and has a restricted representation of dynamics as well as limited segmentation mechanisms. Here, we leverage the capabilities of SR to overcome these limitations by identifying symbolic expressions that capture the dynamic behavior of the system across consecutive segments. SR provides human-readable mathematical expressions for the system dynamics, which is essential for the interpretation of the identified models. The usage of decision trees for model construction also provides an interpretable representation.

Our approach detects mode transitions directly based on the dynamics of the observed data. This goes beyond existing identification strategies, which focus on similarities of observations to detect whether observations belong to the same or different modes of a hybrid system [5, 6, 7]. However, with the same underlying dynamics, the observed trajectories can be completely different, for example if the initial conditions differ. The similarity-based approach can hence lead to many unnecessary transitions. The dynamics-based transition detection is a key contribution of our work, which puts the

focus on the underlying physical laws and detects transitions only if needed. This leads to a simplified model model and a more efficient learning process. Further, we leverage the ability of SR to find symbolic expressions for the system dynamics, which is essential for the interpretation of the identified models.

An illustrative example for the relevance of detecting transitions based on the dynamics of the observed data is shown in Figure 2. The figure shows a trajectory of a bouncing ball, where the height of the ball is observed over time. Considering the similarity of signals, transitions would be detected at each bounce. Further, the height values between the bounces are different, and thus a similarity-based approach would consider them as different modes. However, the bouncing ball has one single mode and its dynamics can be described by a single function, which is:

$$\begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} + \begin{pmatrix} 0 \\ -g \end{pmatrix}, \tag{1}$$

where x is the height of the ball, v is the velocity, and g is the gravitational acceleration. Thus, every singularity does not identify a transition and a new dynamic mode of the system. Our approach automatically determines that the dynamics of the bouncing ball follows a single expression.

Our algorithm works as follows. We start from a small window of the observed data, learn a symbolic expression using SR, and then iteratively enlarge this window until a transition point is detected. A transition point is detected as soon as the fitness of the learned expression drops. The segments between detected transition points are then grouped to identify the modes of the hybrid system. For this grouping step, we reuse SR to learn expressions

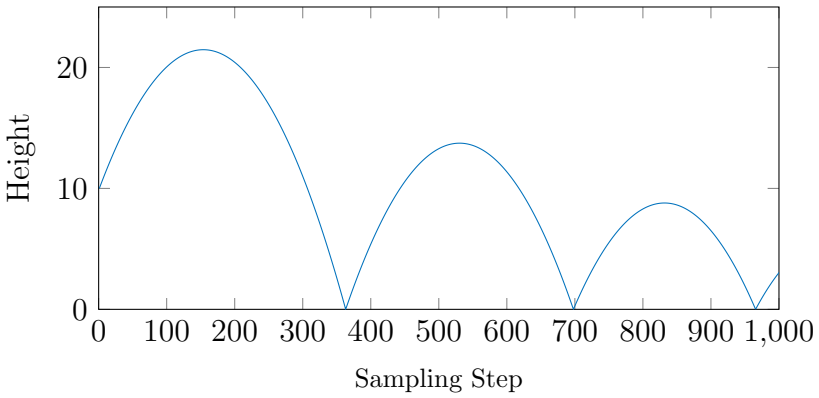


Figure 2: Example of an observed trajectory of a bouncing ball. The height of the ball is measured at each sampling step.

on unions of previously detected segments. Segments with a small loss on the same symbolic expression form a group. The expressions found for each group identify the modes. Finally, the groups are mapped to a learning set for decision tree learning. The learned decision tree represents the transitions by determining the current mode of the system. The final hybrid decision tree consists of a decision tree and a map from system mode to symbolic expression.

Novelties compared to Plambeck et al. [3] as well as Plambeck et al. [4] and Plambeck et al. [8], come from the following contributions of this paper:

- the completion to a holistic identification approach by adding a model construction step with a decision tree,
- the analysis of the parameter importances for the presented algorithms,
- a complexity analysis of the proposed learning algorithms, which in nominal cases is linear in the number of samples,

- the refinement and extension of the empirical evaluation. This includes a metric for the identification of transition points as introduced in [8]. One of the examples is further extended by a second scenario with limited system information compared to [3], and
- the comparison of our approach with two state-of-the-art approaches for hybrid system identification, FaMoS [4] and HAutLearn [6], on the same example systems,
- the open-source publication of an implementation in Python [9].

With respect to the state-of-the-art in hybrid system identification as a whole, this paper contrasts the other works in that:

- the identification is based on the dynamics of the system, focusing on the physical laws underlying the data rather than on the similarity of data,
- the detection of transitions and grouping segments with similar dynamics are both achieved utilizing SR,
- the last step of model learning constructs the hybrid model with a new type of decision tree integrating the learned symbolic expressions.

The paper is structured as follows. In Section 2, we review related work on system identification, specifically for hybrid systems and symbolic regression. In Section 3, we introduce the necessary formal definitions. In Section 4, we present our approach for the identification of hybrid systems using SR, the individual steps as well as the final model construction. In Section 6, we

evaluate our approach on two representative examples. Finally, in Section 7, we conclude the paper.

2. Related Work

In the following, we summarize and relate literature from three main aspects of this work, which are SR in the context of system identification, modern advances of decision trees, and existing algorithms for hybrid system identification.

2.1. Symbolic Regression for System Identification

SR is a method for regression and system identification to find symbolic expressions that match a learning data set. Contrary to traditional regression methods, SR is not restricted to a specific class 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 [10]. Several studies show that the performance of SR based on genetic programming improves by using a suitable parametrization. The population size and the number of iterations are considered the most crucial parameters – usually a larger population and number of iterations lead to higher accuracy [11, 12, 13]. In addition to genetic programming, there exist other approaches for SR, e.g., using deep reinforcement learning [14], lattices [15], or sparse regression methods [16].

SR has been applied to a wide range of problems, including the identification of physical concepts from data [17], mining the expression of diagnosis indicators [18], identify properties in material sciences [19], and even for the

identification of a machine learning algorithm [20]. The works closest to ours are:

- the system identification procedure of Gaucel et al. [21], which focuses on learning matrix ordinary differential equations with SR but does not consider hybrid dynamics.
- the learning process of symbolic representations of hybrid dynamical systems with multi-modal SR of [22], which uses SR with an expectation-maximization metric to learn symbolic expressions on noise data and does not integrate learned functions into a model such as we do with a decision tree.

Our work advances beyond previous approaches by integrating all necessary components to address the hybrid nature of systems, specifically detecting transitions and grouping segments with similar dynamics, incorporating benefits of SR.

2.2. Advancements of Decision Trees

Decision trees are a well-established and widely used method in machine learning, known for their interpretability and effectiveness. Some research is orthogonal to ours and uses well-established decision trees in an innovative manner. For example, Fong and Motani [23] proposes another combination of symbolic regression and decision trees, which results in a more effective structure of learned decision trees, but applies only for classification tasks. Goupil et al. [18] combines decision trees with symbolic classification for diagnosis purposes. Further, Plambeck et al. [24] and Jimenez et al. [25] use decision

trees to learn discrete models of cyber-physical systems, where [24] showcases theoretical limitations of decision trees in representing time-discrete behavior.

Our work uses decision trees for the hybrid model construction step, thus, generating a hybrid decision tree, which enhances a traditional decision tree with symbolic expressions.

2.3. Identification of Hybrid Systems

In this paper, we focus on the identification of hybrid systems. Existing approaches use a variety of methods, but follow the procedure of Figure 1 consisting of steps for the detection of transitions, grouping of segments, mode identification, and model construction. In Yang et al. [6], the authors use a sliding window approach to detect transitions, considering the similarity of data points to detect transitions. This procedure is unaware of the underlying dynamics and potentially reports transitions that do not exist. In contrast, our approach detects transitions based on the dynamics of the observed data, thus, detecting transitions only if necessary.

Similarly, [7] uses a wavelet transform to detect transitions, which focuses on similarities in the frequency domain. Thus, this approach also does not consider the underlying dynamics.

There are clustering-based approaches [5, 26] which use spatial proximity of observations to segment and cluster trajectories into groups. Nevertheless, similar dynamics do not necessarily lead to similar observations, which is a problem solved by our approach.

For the model identification step, other approaches use, e.g., feed forward neural networks [7]. In contrast, we use SR to identify explicit symbolic

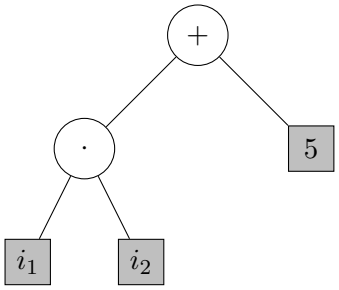


Figure 3: Example of a Symbolic Regression Tree

expressions for the dynamics of the system. This finds human-readable expressions for the dynamics, which is essential for the interpretation of the identified models.

3. Preliminaries

In this section, we introduce the basics of SR and decision trees and formally define hybrid systems and system observations.

3.1. Symbolic Regression

SR is a machine learning technique that finds symbolic expressions that describe the relationship between input and output variables. The goal is to find a function r that maps input variables $\mathbf{i}$ to an output variable o such that $o = r(\mathbf{i})$. The function r is represented as a symbolic expression in terms of a set $\mathcal{B}$ of basic functions and operators applied to the input variables [27]. To represent symbolic expressions, often a tree structure is used, where the leaves are variables or constants and the inner nodes are operators from $\mathcal{B}$. An example of a symbolic regression tree is shown in Figure 3 for the expression $o = (i_1 \cdot i_2) + 5$.

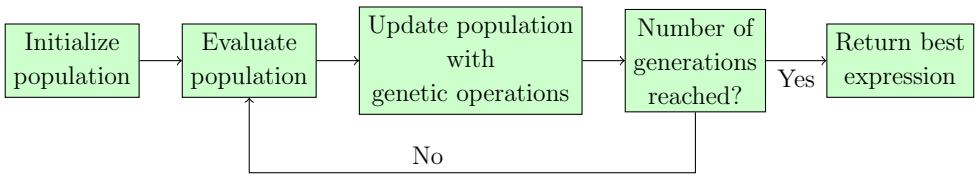


Figure 4: Genetic Programming Algorithm for Symbolic Regression

The search for the best symbolic expression is guided by a fitness metric that evaluates the quality of candidate symbolic expressions on a learning data set. Often, a trade-off between the loss of the expression on the learning data and the length of the candidate expression is used as a fitness metric. This strategy avoids a bloat of the expressions [28]. A parsimony coefficient ρ works as a regularization parameter in this trade-off.

In the scope of this work, we use the framework PySR, which uses genetic programming [27, 29]. Using genetic programming, the search for the best symbolic expression is performed by evolving a population of candidate expressions over so-called *generations*. Figure 4 illustrates this process. A population of size p is initialized with random expressions. In each generation, the population is evaluated using a fitness metric. New candidate solutions are generated by applying genetic operators, such as mutation and crossover, to the best candidates. The process continues for a predefined number n of generations.

3.2. Decision Trees

A decision tree [30] is a tree $T = (V, E)$ with nodes V and edges E that represents a classifier $d : \mathcal{X} \to \mathcal{C}$. The set $\mathcal{X}$ consists of vectors of feature values $\mathbf{f} = [f_1, ..., f_m]$, while $\mathcal{C}$ is a set of classes. A decision tree learner constructs a decision tree based on a learning set $\mathcal{L}$, where each element

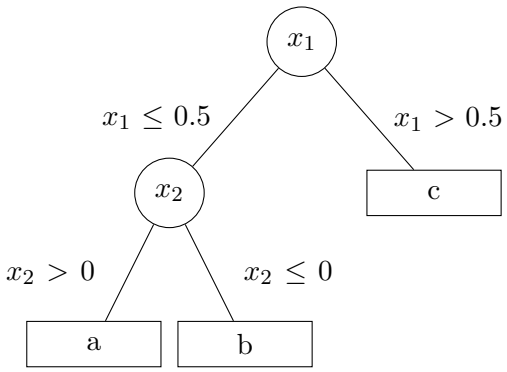


Figure 5: Example of a Decision Tree

$(\mathbf{f}, c) \in \mathcal{L}$ is a tuple of a feature vector $\mathbf{f} \in \mathcal{X}$ and a class label $c \in \mathcal{C}$. The decision rule d adapts to these tuples. The set of nodes with no outgoing edges V_L are the leaf nodes of the tree. All nodes $V \setminus V_L$ are *inner nodes* of the tree. Each node v in T associates to a subset L_v of $\mathcal{L}$, denoted by $v \sim L_v$ and splits this set into subsets forming child nodes. The splitting rule bases on one of the features in $\mathbf{f}$. Each leaf $v_l \sim L_{v_L}$ has a label $c_{v_l} \in \mathcal{C}$ given by the majority of class labels in the associated set, i.e.,

$$c_{v_l} = d(\mathbf{f}) = \underset{c}{\operatorname{argmax}} \#c, \text{ where} \tag{2}$$
$$\#c = |\{\mathbf{f} : (\mathbf{f}, c) \in L_{v_l}\}|.$$

Figure 5 shows an example of a decision tree. The inner nodes represent decisions based on the values of the feature variables x_1 and x_2, while the leaf nodes represent the classes $C = \{a, b, c\}$. The edges between the nodes represent the conditions for the decisions.

A common decision tree learning algorithm is the CART algorithm, which creates a binary decision tree [31].

Definition 1. *A Hybrid System [32] is a 5-tuple* $(X, Q, \mathcal{F}, \mathcal{T}, \Sigma)$*, where*

- $X = \{x_0, x_1, x_2, ..., x_n\} = I \cup O \cup S$ *is the set of system variables which consist of input variables* I*, output variables* O *and state variables* S*. The state variables may include the continuous time* t*. Also, derivatives of variables may form individual variables in* X*.*
- Q *is the set of modes.*
- $\mathcal{F}$ *is the set of flow functions. A flow function* $f_q \in \mathcal{F}$ *defines the change of the state variables* S *and the current output variables* O *within the mode* $q \in Q$ *based on the current values.*
- $\mathcal{T} : Q \times \Sigma \rightarrow Q$ *defines transitions between modes. A transition is triggered if the corresponding guard* $\sigma \in \Sigma$ *is active.*
- Σ *is a set of guards leading to transitions between modes. Each guard is a set of conditions on the variables in* X*. A transition is triggered if the conditions are met.*

A hybrid system combines discrete and continuous behavior and is usually an abstraction of a complex real-world system. Discrete behavior is captured by the discrete modes Q and the transitions $\mathcal{T}$. The continuous dynamics are represented by the flow functions $\mathcal{F}$ of the modes. Observations of dynamics, i.e., changes in the values of the variables, are considered as *trajectories* which are introduced in the following.

Figure 6 shows an example of a hybrid system, which is a heating system. The system has two modes: heating and cooling. The flow functions

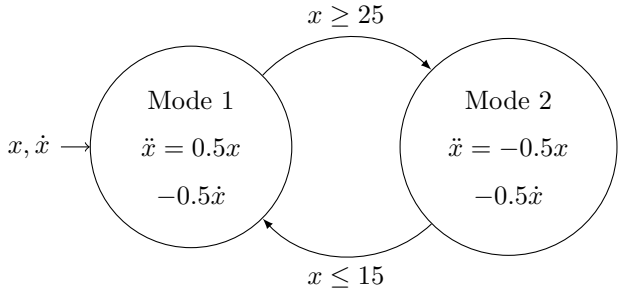


Figure 6: Example of a Hybrid System: Heating System

for the modes are given by the differential equations for the temperature change in the system. The transitions between the modes are triggered by the temperature reaching a certain threshold.

3.4. Observations

We define a *trajectory* as an observation in form of a multidimensional time series. Every sample in this series contains values for the system variables X.

The input to the identification algorithm is a set of trajectories. Each trajectory covers multiple transitions between the modes and stems from observing the real-world system for a finite time interval.

4. Identification of a Hybrid System

In this section, we provide our identification strategy for hybrid systems. We name our method HyDRA (Hybrid systems via Dynamics and Regression-based Analysis) inspired by the multi-headed Hydra of mythology which reflects the multi-modal nature of hybrid systems. First, we give

an overview of the approach and, then, describe the segmentation of trajectories, the grouping of segments, and the model construction step in detail. Finally, we describe how the hybrid system is inferred.

4.1. Overview & System Properties

Definition 1 specifies hybrid systems in a general form. 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:

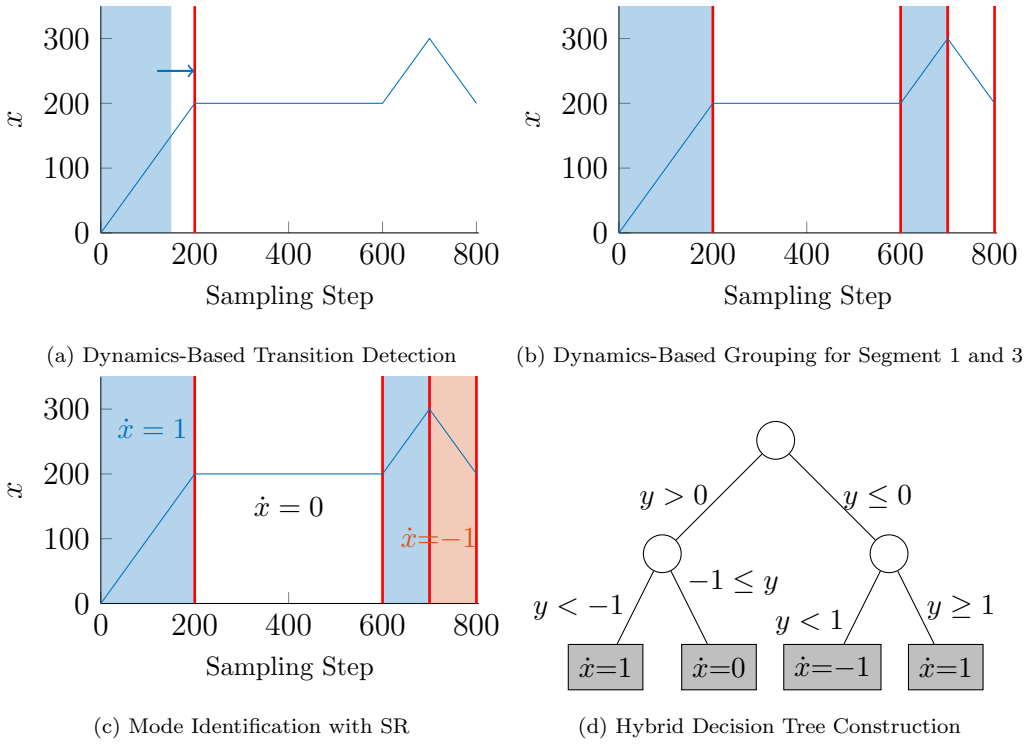
(a) detection of transition points between dynamics, separating the trajectories into segments,

(b) grouping of segments with identical dynamics, forming discrete modes,

(c) identification of the continuous dynamics for each mode, i.e., the flow functions of $\mathcal{F}$,

(d) model construction, i.e., creation of an inferable hybrid system from the results of the previous steps.

We cover all of these steps, where the main idea is illustrated in Figure 7. The first step (a) detects transition points by extending a window over the trajectory until the dynamics do not follow the same flow function anymore. Step (b) groups segments with identical dynamics. Step (c) includes the mode identification of Step (b) as each flow function learned by SR corresponds to a mode of the hybrid system. Finally, Step (d) constructs a hybrid decision tree model from the results of the previous steps.



(a) Dynamics-Based Transition Detection

(b) Dynamics-Based Grouping for Segment 1 and 3

(c) Mode Identification with SR

(d) Hybrid Decision Tree Construction

Figure 7: Illustration of the Proposed Learning Approach.

We, hence, present a complete identification procedure. The procedure contains an algorithm using SR to detect the transition points of trajectories observed on a hybrid system. A second algorithm groups and identifies the flow functions of modes with SR using the segmented trajectories. Lastly, the procedure has a strategy to form a hybrid system involving decision tree learning. This final step combines the results of the previous step to a model that can be inferred and predicts system behavior on new, unseen data.

Our approach considers one-dimensional flow functions. The extension to higher dimensions is straightforward by processing dimensions individually, but requires additional consideration for the detection of segments, as we have to combine detected transition points for all dimensions. Further, we assume that the training data covers the dynamical modes sufficiently to extract flow functions of the system.

4.2. Detection of Transition Points

Our approach for detecting transition points between different modes of a hybrid system uses SR. We define *transition points* as the first samples in the observed trajectory after the transition between two dynamical modes. Algorithm 1 describes the procedure to detect these points. Given an observed trajectory of the system, we process the trajectory using a window (Line 5) that initially covers a fixed segment at the beginning of the trajectory. We propose a *segmentation criterion* which determines whether the loss of the learned expression drops, when updated with additional data. As long as the segmentation criterion is fulfilled, the window is extended to the right by l_{step} (Line 9). As soon as the segmentation criterion is not fulfilled anymore, we detect a transition point between two modes. The identified window is

Data: trajectory

Result: $\mathcal{S}, expressions$

Hyperparameter: $l_{init}, l_{step}, n_{init}, n_{update}, segmentationCriterion, \rho_S, p_S, \mathcal{B}_S$

loss $\leftarrow$ [];
$i_{start} \leftarrow 0$; $i_{end} \leftarrow l_{init}$; $n \leftarrow n_{init}$;
while $i_{end} < len(trajectory)$ **do**
    while *loss is empty or segmentCriterion*(*loss*) **do**
        window $\leftarrow$ trajectory$[i_{start}, \min(i_{end}, len(\text{trajectory}))]$;
        exp, fit $\leftarrow$ learnExpression(window,n);
        loss[0] $\leftarrow$ loss[1];
        loss[1] $\leftarrow$ fit;
        $i_{end} \leftarrow i_{end} + l_{step}$;
        $n \leftarrow n_{update}$;
    end
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{\text{window}[0, \text{end} - l_{step}]\}$;
    expressions.append(exp);
    loss $\leftarrow$ [];
    $i_{start} \leftarrow i_{end} - l_{step}$; $i_{end} \leftarrow i_{start} + l_{init}$; $n \leftarrow n_{init}$;
    resetSR;
end

Algorithm 1: Detection of Transition Points

stored as a segment in the set $\mathcal{S}$. In the inner loop, we learn the symbolic expression incrementally, i.e., by performing n_{update}-many SR iterations to adapt the current expression to the extended window. Whenever a transition point is found, learning restarts (Line 16) and creates a new window starting at the end of the previously identified window. For each segment, a new expression is learned from scratch.

The loss function in Algorithm 1 determines the deviation between the estimation of the learned expression and the learning data. Here, the mean-square error is used. The algorithm is parametrized by:

- l_{init}: the initial window size when learning an expression from scratch,
- l_{step}: the step-width for extending the window in the inner loop,
- n_{init}: the number of iterations of SR when learning an expression from scratch,
- n_{update}: the number of iterations of SR for updating the learned expression on an extended window,
- *segmentationCriterion*: the criterion to determine whether a transition point is reached. Here, the segmentation criterion executes the following evaluation:

$$\begin{aligned} &\mathit{segmentationCriterion}(\text{loss}) = \\ &(\text{loss}[1] < \tau) \quad \text{or} \quad (\text{loss}[1] \leq \text{loss}[0]), \end{aligned} \tag{3}$$

  which evaluates to true, if the loss of the current expression is smaller than a stagnation threshold τ or if the loss does not decrease anymore compared to the previous iteration.

- τ: A stagnation-threshold for the segmentation criterion.

Additionally, SR has the following parameters:

- ρ_S: the parsimony coefficient of SR,
- p_S: the number of individuals in the population of SR,
- $\mathcal{B}_S$: the set of basic operators and functions for SR.

4.3. Grouping and Identification of Modes

Segments whose dynamics are described by the same symbolic expression refer to the same mode of the system. Thus, the next step groups segments with identical dynamics. Algorithm 2 shows the pseudocode for segment grouping using SR. The input to the grouping is the set of detected segments $\mathcal{S}$ from the previous step. Output of the algorithm are two maps G and *expressions*, which map a group ID γ to a conjunction of segments and a symbolic expression, respectively.

The first segment forms a first candidate group (Line 1). Afterward, we iterate for every segment in $\mathcal{S}$ (Line 2) over all known groups (Line 4). The current segment and the current group are combined to one data set to learn an expression with SR (Line 5). If the loss of the learned expression is small, the current segment is included in the current group (Lines 6 and 7). If no matching group is found for the current segment, the segment forms a new group (Line 13).

The hyperparameters of the algorithm are:

- n_{group}: the number of iterations of SR for learning an expression on combined segments,

- *groupingCriterion*: a criterion to determine whether a segment is joined with a group. Here, we use the following evaluation:

$$groupingCriterion(s, g) = \\ \text{loss}(\{s\} \cup g) \leq \varphi \cdot \text{loss}(g), \tag{4}$$

  where $\text{loss}(g)$ gives the loss of the group observed before combining with the segment s and $\text{loss}(\{s\} \cup g)$ is the loss after combination – the smaller, the better. Here, the mean-square error is used as the loss metric. The grouping criterion evaluates to true if the loss of the combined segments is smaller than φ times the loss of the group before combination. The parameter φ is a relaxation parameter, which allows for some flexibility in the grouping.

- φ: the relaxation parameter of the grouping criterion,

Additionally, SR has the following parameters:

- ρ_G: the parsimony coefficient of SR,
- p_G: the number of individuals in the population of SR,
- $\mathcal{B}_G$: the set of basic operators and functions for SR.

4.4. Model Construction

In our previous work [4], we propose the usage of decision trees for hybrid system construction. We construct a learning set for decision tree learning from the results of the previous steps. Let each group g of segments s have a unique group ID γ. The set G of groups holds all tuples (γ, g). Each sample

Data: $\mathcal{S}$

Result: G, expressions

Hyperparameter: $n_{group}, groupingCriterion, \rho_G, p_G, \mathcal{B}_G$

$G \leftarrow \{(0, \mathcal{S}.pop())\}$;
for $s \in \mathcal{S}$ **do**
    groupFound $\leftarrow$ False;
    for $(\gamma, g) \in G$ **do**
        exp $\leftarrow$ learnExpression$(\{s\} \cup g, n_{group})$;
        if *groupingCriterion(s, g)* **then**
            $g \leftarrow \{s\} \cup g$;
            expressions$[\gamma] \leftarrow$ exp;
            groupFound $\leftarrow$ True;
            break;
        end
    end
    if *not groupFound* **then**
        G.append$((|G|, s))$;
    end
end

Algorithm 2: Grouping of Segments

$s[k]$ is a feature vector $\mathbf{f}$ and the respective group ID a class label c. Thus, the learning set $\mathcal{L}$ is defined as follows:

$$\mathcal{L} = \{(s[k], \gamma), \quad \forall s \in g, \gamma \in \mathbb{N}^0, k \in \{0, \ldots, |s| - 1\} \text{ with } (\gamma, g) \in G\}, \quad (5)$$

where $s[k]$ gives the values of all variables X of the system for sample k of segment s in group g. These values are used as the feature vector $\mathbf{f}$, while the group ID γ is used as the class label c. On this learning set, a decision tree T is learned.

We finally define a hybrid decision tree M_{HDT}. The hybrid decision tree model consists of the learned decision tree T as well as the map *expressions*, which is a result of the grouping step. Figure 8 gives an example of such a hybrid decision tree for a system with three variables $X = \{x_0, x_1, x_2\}$. Every inner node of the decision tree splits on one of the variables. The labels of the edges of the decision tree are the splitting rules. For example, the root node in Figure 8 decides on the value of the variable x_1 and the left edge is valid for $x_1 > 3.3$, while the right edge is valid for $x_1 \leq 3.3$. The leaf nodes classify one of the group IDs $\gamma \in \{0, 1, 2\}$. Finally, the expressions map these group IDs to symbolic expressions as shown on the left of Figure 8. The hybrid decision tree forms a variant of a hybrid system as defined in Definition 1 by representing the modes Q and flow functions F by the expressions while the transitions $\mathcal{T}$ and the guards Σ are encoded in the decision tree.

4.5. Hybrid System Inference

With the hybrid decision tree, a representation of the guards and transitions of the hybrid system is found. Additionally, the model is inferable and can perform predictions on new samples. The inference of the model contains

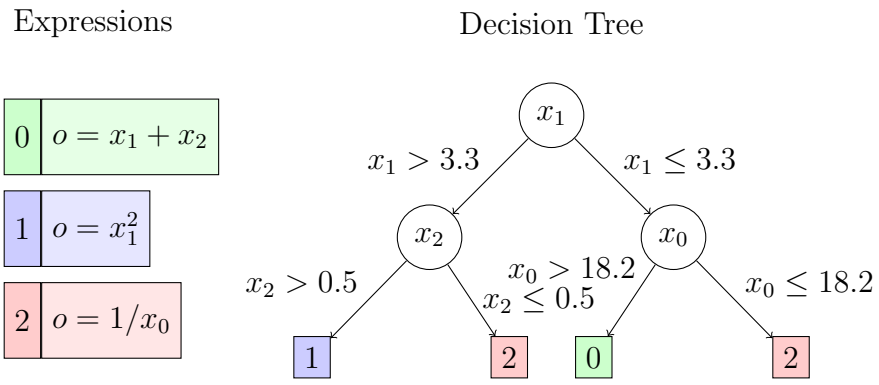


Figure 8: Hybrid Decision Tree consisting of decision tree and expression

three steps as shown in Algorithm 3. Using the hybrid decision tree M_{HDT}, we first apply the decision function d of the decision tree to the sample $\mathbf{x}$ to get the current group ID γ. Using this group ID, we identify the correct $flow$-function. This function is used to evaluate the system dynamics using the variable values in $\mathbf{x}$.

Data: $\mathbf{x}, M_{HDT}$

Result: O

1 **begin**

2 $\quad \gamma \leftarrow M_{HDT}.tree.d(\mathbf{x})$;

3 $\quad flow \leftarrow M.expressions[\gamma]$;

4 $\quad O \leftarrow flow(\mathbf{x})$;

5 **end**

Algorithm 3: Inference of Hybrid Decision Tree

5. Complexity Analysis

We analyze the computational complexity of our approach by examining each step of the identification procedure. First, we present the complexity with respect to model size, i.e., the number of modes d_m, treating SR as a black-box algorithm with complexity K. Second, we analyze the complexity with respect to the number of learning samples. Here, we assume a genetic programming-based SR algorithm (Figure 4). The complexity per iteration is $O(p \cdot N_s)$, where p is the population size and N_s is the number of samples in the learning data set. Over n iterations, the total SR complexity is $K = O(p \cdot n \cdot N_s)$.

5.1. Complexity in Terms of Model Size

The complexity of the identification procedure is determined by the complexity of Algorithm 1 for transition point detection, Algorithm 2 for segment grouping, and the decision tree construction step.

Transition Point Detection. The transition point detection depends on the number of segments d_s, the number of window extensions per segment d_e, and has a complexity of $O(d_s \cdot d_e \cdot K)$. The number of segments is linear in the number of modes d_m, i.e., $d_s = O(d_m)$, and the maximum number of window extensions per segment is constant, i.e., $d_e = O(1)$. Thus, the complexity of transition point detection is:

$$O\left(d_s \cdot d_e \cdot K\right) = O\left(d_m \cdot 1 \cdot K\right) = O(d_m \cdot K). \tag{6}$$

Segment Grouping & Mode Identification. The segment grouping step has a complexity of $O(d_s \cdot d_g \cdot K)$, where d_g is the number of groups. Again, the

number of segments is linear in the number of modes, i.e., $d_s = O(d_m)$ and the number of groups is equivalent to the number of modes, i.e., $d_g = d_m$. Thus, the complexity of segment grouping is:

$$O\left(d_s \cdot d_g \cdot K\right) = O\left(d_m \cdot d_m \cdot K\right) = O(d_m^2 \cdot K). \tag{7}$$

Model Construction & Overall Complexity. The complexity of the decision tree construction depends on the number of learning samples and is, thus, considered as $O(1)$ in terms of model size.

The overall complexity of the identification procedure is dominated by the transition point detection and segment grouping steps. As the phases are executed sequentially, the complexity sums up to:

$$O\left(d_s \cdot d_e \cdot K + d_s \cdot d_g \cdot K\right) = O\left(d_m \cdot K + d_m^2 \cdot K\right) \tag{8}$$

$$= O(d_m^2 \cdot K). \tag{9}$$

In summary, the complexity of the identification procedure in terms of model size is $O(d_m^2 \cdot K)$, where d_m is the number of modes and K is the complexity of the SR algorithm.

5.2. Complexity in Terms of Learning Samples

Next, we analyze the complexity of the identification procedure with respect to the number of learning samples N_t in the trajectory. Here, we also assume a genetic programming-based SR algorithm (Figure 4) with complexity $K = O(p \cdot n \cdot N_s)$, where p is the population size, n is the number of iterations, and N_s is the number of samples in the learning data set. The parameters p and n are constant for the SR algorithm.

Transition Point Detection. Algorithm 1 has a complexity of

$$O\left(d_s \cdot d_e \cdot K\right), \tag{10}$$

where d_s is the number of detected segments (outer loop iterations), d_e is the maximum number of window extensions per segment (inner loop iterations), and K is the SR complexity. The algorithm processes the trajectory using two nested loops, calling SR in each iteration to learn an expression for the current window.

The outer loop advances by l_{init} samples per iteration, while the inner loop extends the window by l_{step} samples. For a trajectory of length N_t, the total number of SR calls is bounded by:

$$d_s \cdot d_e \leq \frac{N_t}{\min(l_{init}, l_{step})}. \tag{11}$$

With $l_{init}, l_{step} \sim N_s$, we have $d_e = O(1)$. Assuming many segments, the segment size is small, while for a few segments, the segment size is large, i.e., $d_s \cdot N_s = O(N_t)$. Thus, the complexity simplifies to:

$$O(d_s \cdot d_e \cdot p \cdot n \cdot N_s) = O(p \cdot n \cdot N_t), \tag{12}$$

where we use $K = O(p \cdot n \cdot N_s)$. However, if initial window and step sizes are small and segments are large, i.e., $l_{init}, l_{step} \ll N_t$, we have $d_s \cdot d_e = O(N_t), N_s = O(N_t)$, and the complexity approaches

$$O(p \cdot n \cdot N_t^2). \tag{13}$$

The choice of l_{init} and l_{step} influences whether a complexity of Equation (12) or Equation (13) applies. Thus, the choice of these parameters is

a trade-off: larger values reduce computational cost but may miss transition points, while smaller values increase accuracy at higher computational expense.

Segment Grouping & Mode Identification. Algorithm 2 compares each segment against existing groups, requiring $O(d_s \cdot d_g)$ comparisons in the worst case. Thus, the algorithm has complexity

$$O\left(d_s \cdot d_g \cdot K\right) = O\left(d_s \cdot d_g \cdot p \cdot n \cdot N_s\right), \tag{14}$$

where d_s is the number of segments, d_g is the number of groups, and K is the SR complexity.

In the worst case, every segment forms its own group, i.e., $d_g = d_s$. Further, the length of a segment could approach the trajectory length N_t, i.e., $N_s = O(N_t)$. This yields complexity:

$$O\left(d_s^2 \cdot p \cdot n \cdot N_s\right). \tag{15}$$

When $d_s \ll N_t$, this is effectively $O(p \cdot n \cdot N_t)$, i.e., linear in trajectory length.

In practice, the inner loop terminates early when a matching group is found, significantly reducing the average number of iterations and improving practical performance.

Model Construction & Overall Complexity. The decision tree construction step has complexity $O(|\mathcal{L}| \log |\mathcal{L}|)$, where $|\mathcal{L}| = N_t$ is the size of the training set [30]. This step is typically dominated by the SR-based components.

The total complexity is dominated by the transition detection and grouping phases. As the phases are executed sequentially, the complexity sums up

to:

$$O\left((d_s \cdot d_e + d_s \cdot d_g) \cdot K\right) = O\left((d_s \cdot d_e + d_s \cdot d_g) \cdot p \cdot n \cdot N_s\right) \tag{16}$$

$$= O(p \cdot n \cdot N_t \cdot d_s \cdot \max(d_e, d_g)). \tag{17}$$

With a suitable choice of l_{init} and l_{step}, this simplifies to $O(p \cdot n \cdot N_t)$, which is linear in the trajectory length.

Table 1: Example Systems

System	l_{init}	l_{step}	n_{init}	n_{update}	τ	n_{group}	φ	ρ_G	ρ_S	p_G
Circuit 1	200	100	20	5	$1 \cdot 10^{-7}$	20	1.5	$1 \cdot 10^{-6}$	$1 \cdot 10^{-6}$	15
Circuit 2	200	100	20	5	$1 \cdot 10^{-7}$	50	1.5	$1 \cdot 10^{-6}$	$1 \cdot 10^{-3}$	15
Bouncing Ball	50	50	20	5	$1 \cdot 10^{-4}$	20	1.5	$1 \cdot 10^{-6}$	$1 \cdot 10^{-6}$	15
Power Converter	100	20	100	0.003	$1 \cdot 10^{-6}$	270	0.65	0.003	0.865	20
Two-tank Sys. *sub*	40	1	150	25	$1.28 \cdot 10^{-8}$	200	0.79	0.450	0.760	15
Two-tank Sys. *w/o sub*	50	10	200	5	$1 \cdot 10^{-6}$	100	1.25	0.850	0.110	40

6. Experiments

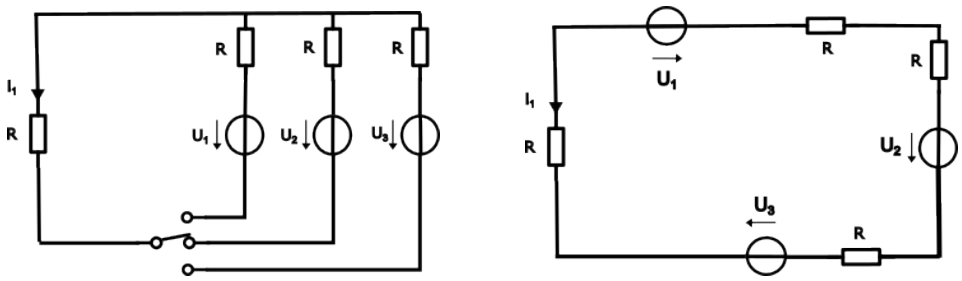
In this section, we present empirical results on two example systems. We analyze all steps, i.e., the trajectory segmentation, the segment grouping and the model construction individually, considering suitable metrics. The assessment of the final model construction also covers the prediction capabilities of the learned hybrid decision tree.

6.1. Example Systems

We present empirical results on four example systems. The examples are selected to cover different aspects of the approach and include well-known benchmark systems as well as a real-world system.

6.1.1. Electrical Circuits

We consider two versions of a passive electrical circuit, namely Circuit 1 and Circuit 2, where the goal is to learn an expression for the output current I_1. The circuits are shown in Figure 9. The first circuit in Figure 9a is



(a) Circuit 1 – Switched sources

(b) Circuit 2 – Connected sources

Figure 9: Passive Electrical Circuits

described by

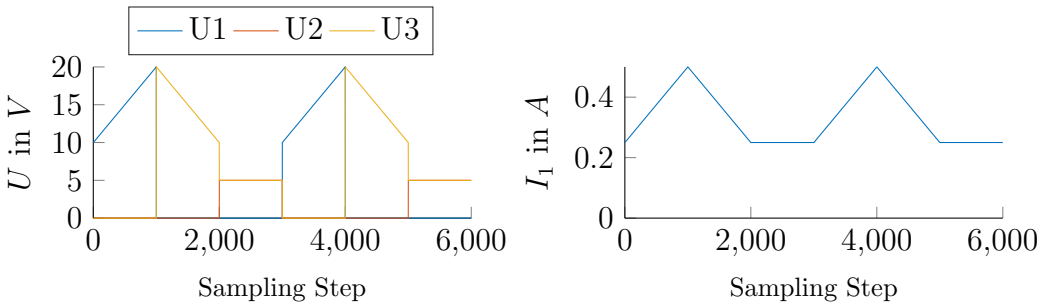
$$I_1 = \begin{cases} \frac{U_1}{2 \cdot R}, & \text{switch in 1st position} \\ \frac{U_2}{2 \cdot R}, & \text{switch in 2nd position} \\ \frac{U_3}{2 \cdot R}, & \text{switch in 3rd position} \end{cases} . \tag{18}$$

Depending on the position of the switch, one of the three sources is selected.

Circuit 2, shown in Figure 9b, contains three sources as well, but all of them are connected within the circuit. The system is described by the equation

$$I_1 = \frac{U_1 + U_2 + U_3}{4 \cdot R}. \tag{19}$$

Figure 10 shows the three voltages and the output current I_1 over the discrete sampling points k for Circuit 2. This example is interesting to showcase the capabilities of HyDRA. The visual inspection of the current and voltage signals suggests that the system has multiple modes. Nevertheless, the system is described by a single equation. A dynamics-based approach identifies that a single dynamic mode is sufficient to cover the behavior of the system.



(a) Input voltages U_1, U_2, U_3 of Circuit 2

(b) Output current I_1 of Circuit 2

Figure 10: Current and Voltages of Circuit 2

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 value of the resistance R are given as variables for learning an expression for I_1. Both circuits use $R = 10\Omega$ for all resistors.

6.1.2. Bouncing Ball

The bouncing ball is a well-known example for system identification, which we use as a motivating example in Figure 2. The system consists of a ball dropped from an initial height, bouncing on the ground. We use a simulation of the bouncing ball in Matlab [33] with a sampling time of $\Delta t = 0.01s$.

The system dynamics are governed by the matrix differential equation:

$$\begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} + \begin{pmatrix} 0 \\ -g \end{pmatrix}, \tag{20}$$

where g denotes the gravitational acceleration and v is the velocity of the ball.

We observe the system with a fixed sampling rate and, thus, learn a one

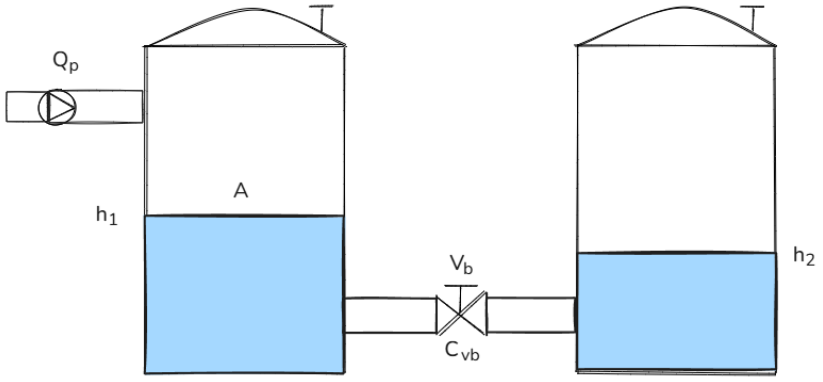


Figure 11: Illustration of the Two-tank System

dimensional discrete approximation for the height of the ball given by

$$x(k) = 2 \cdot x(k-1) - x(k-2) - \frac{g}{2} \cdot \Delta t. \tag{21}$$

For SR, the features are the previous two height values $x(k-1)$ and $x(k-2)$ and the set of operators contains addition, subtraction, multiplication, and division.

6.1.3. Two-tank System

The Two-tank system [34] is a benchmark system for hybrid systems as shown in Figure 11. It consists of two tanks, with a pump pumping water into the first tank and a valve V_b that can be opened to let water flow from the first to the second tank. The system has a controller to regulate the height of the water in the first tank. A differential equation describes the physical behavior of the water level in the first tank depending on the height of the second tank and the valve state. The system has three modes: two modes describe the behavior when the valve between the two tanks is open. In this case, the height of the water changes with the constant inflow and

based on the pressure difference between the two tanks, which is described by the square root of the height difference. The third mode describes the behavior when the valve is closed, in which case the height of the first tank only changes based on the inflow of the pump. The following differential equation describes the system:

$$\dot{h}_1 = \begin{cases} \dfrac{Q_p - C_{vb} \cdot \sqrt{h_1 - h_2}}{A}, & \text{if } h_1 > h_2, V_b \text{ open} \\ \dfrac{Q_p + C_{vb} \cdot \sqrt{h_2 - h_1}}{A}, & \text{if } h_1 \leq h_2, V_b \text{ open} \\ \dfrac{Q_p}{A}, & \text{if } V_b \text{ closed,} \end{cases} \tag{22}$$

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 noise with a maximum amplitude of 10^{-6} for all sensors.

The operators for SR in the segmentation and grouping step are addition, subtraction, multiplication, and division as well as the square-root. The variables used for learning are the inflow Q_p, the height of the two tanks h_1 and h_2 as well as the constants C_b and A. For decision tree learning, the variable V_b is used as an additional feature. Additionally, we have a second scenario with this example where the term $h_{sqr} = \sqrt{|h_1 - h_2|}$ is pre-calculated on the data and used as an additional variable to learn the derivative $\dot{h}_1$. Practically, providing such pre-calculated terms can be interpreted as partial knowledge about the physical system and the operators and terms that are relevant for

its behavior.

6.1.4. Power Converter

The power converter [35] 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 12. As presented in [36], we use a transformed coordinate system to describe both configurations with the same equations. The following differential equation describes the system:

$$\dot{w} = \begin{pmatrix} 0 & \alpha \\ -\alpha & -\beta \end{pmatrix} w + \begin{pmatrix} 0 \\ \alpha \end{pmatrix} \sigma, \tag{23}$$

where $\alpha = (\sqrt{LC})^{-1}$ and $\beta = (RC)^{-1}$ (parallel case) or $\beta = R \cdot L^{-1}$ (serial case). The transformed coordinates $w = [w_1, w_2]^T$ are constructed from the quantities in Figure 12 as

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

Equation (23) describes the system dynamics in the transformed coordinate system, which is the behavior of a resonant circuit consisting of an inductance, a capacitor, and a resistor. The system has two modes, determined by the switching variable σ, which switches the direction of the input voltage V_g.

Our goal is to model the state variable $\dot{w}_2$. The inductance, capacity, and resistor have the values $R = 400\,\Omega, L = 8\,\mu\text{H}, C = 10.5\,\text{nF}$, and $V_g = 20\,\text{V}$.

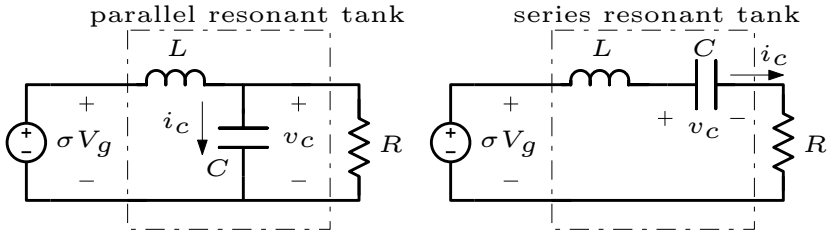


Figure 12: Series and Parallel Representation of the Power Converter [35]

In the learning steps for segmentation and grouping, we use the operators addition, subtraction, multiplication, and division as well as the square-root as basic operators for SR. The to be learned derivative $\dot{w}_2$ is numerically calculated from the data. Feature variables are w_1, w_2, and the continuous time t of a sampling point. Constants are estimated by the learner. Different from the Two-tank example, the simulated data does not involve noise.

6.1.5. General Setup

For both examples, SR uses a fitness metric with regularization and the mean-square error loss as the loss metric. The parameters for the learning algorithm for the examples and scenarios are shown in Table 1. For the Two-tank example and the power converter, the parametrization is done with support of Optuna [37]. We use objective functions Ω_{seg} and Ω_{group} for segmentation and grouping, respectively. The metric Ω_{seg} gives a mean deviation between the detected and ground truth transition points. The metric Ω_{group} gives a mean loss of the found groups compared to the ground truth groups. We determine the loss of the found groups weighted by their size, i.e., the number of samples in the group. The segmentation metric is defined as follows:

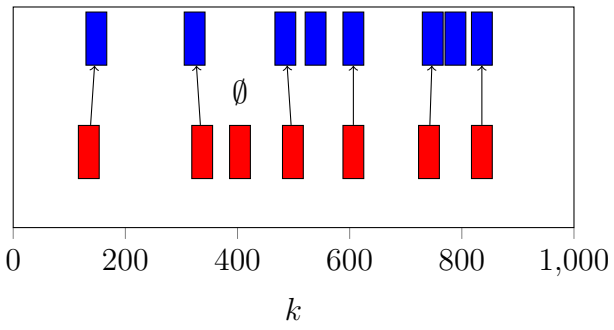
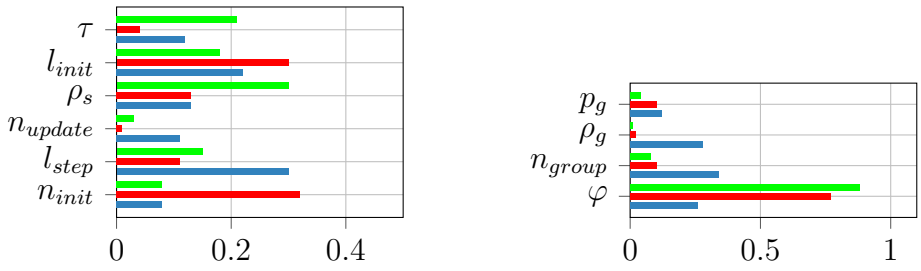


Figure 13: Illustration of mapping m between ground truth (red) and found (blue) transition points for the segmentation objective Ω_{seg}. The mapping for this example is $m = \{(1,1),(2,2),(3,\emptyset),(4,3),(5,5),(6,6),(7,8)\}$.

$$\Omega_{seg}(m, \mathcal{T}_{gt}, \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \left(\omega \cdot ||\mathcal{T}_f| - |\mathcal{T}_{gt}|| + \sum_{(i,j)\in m, j\neq\emptyset} |\mathcal{T}_{gt}(i) - \mathcal{T}_f(j)| \right), \tag{24}$$

where m is a map between ground truth transition points and their closest found transition points or $\emptyset$ if no close transition point is found as illustrated by Figure 13. $\mathcal{T}_{gt}, \mathcal{T}_f$ are vectors of ground truth and found transition points, respectively, while ω is a penalty factor for missed or additionally found transition points. The first term penalizes a divergence in the number of found and ground truth transition points by the factor ω. The second term penalizes a deviation of the found transition points from the ground truth transition points. The grouping metric Ω_{group} is defined as follows:

$$\Omega_{group}(G_{gt}, G_f) = \frac{1 + ||G_{gt}| - |G_f||}{\sum_{g\in G_f} size(g)} \cdot \sum_{g\in G_f} loss(g) \cdot size(g), \tag{25}$$



(a) Segmentation

(b) Grouping

Figure 14: Parameter importances, Two-tank System with Substitution (green), Two-tank System without Substitution (red), Power Converter (blue)

where G_{gt} and G_f are the ground truth and found groups, respectively. Note, that grouping penalizes a divergence in the number of found and ground truth groups by the factor $1 + ||G_{gt}| - |G_f||$.

Optuna also identifies parameter importances using the fANOVA approach [38]. These are useful for the parametrization of future examples. The parameter importances given in Figure 14 show that for most examples, the initial window width l_{init} as well as the length for extending the window l_{step} are relevant for the segmentation while for the grouping especially the relaxation parameter φ is of relevance. A more detailed analysis of parameter influences can be found in Plambeck et al. [8].

The learning approach is implemented in python. The source code including the Optuna parametrization experiments is available on GitHub [9]. All experiments are executed on the same personal computer (Intel i7-9750H 2.6GHz, 16 GB RAM, charging).

Table 2: Detected Transition Points

System	Detected Points	Ground Truth Points	Ω_{seg}	Runtime
Circuit 1	5	5	0.0	448 s
Circuit 2	0	0	0.0	413 s
Bouncing Ball	0	0	0.0	161 s
Power Converter	4	4	2.8	887 s
Two-tank System *sub.*	28	28	1.15	7891 s
Two-tank System *w/o sub.*	27	28	4.64	1919 s

6.2. Detection of Transition Points

In a first step, we analyze the detection properties of our approach. For this, we compare the detected transition points with the ground truth transition points. We use the metric of Equation (24) which determines a mean deviation between the detected and ground truth transition points. Undetected or falsely detected transition points are penalized with a weight $\omega = 10$. Table 2 shows the results for the example systems. The columns provide the number of detected transition points, the number of ground truth transition points, the average deviation metric Ω_{seg}, and the runtime of the segmentation step. The results show that the approach detects the transition points in both systems with high precision. Nearly all transition points are detected perfectly. For the converter, the detected transition points are shown on the left of Figure 15. For the two-tank example, one transition point was missed which is a transition happening after 8 samples at the beginning of the learning trajectory. These samples, thus, are likely to be included in a

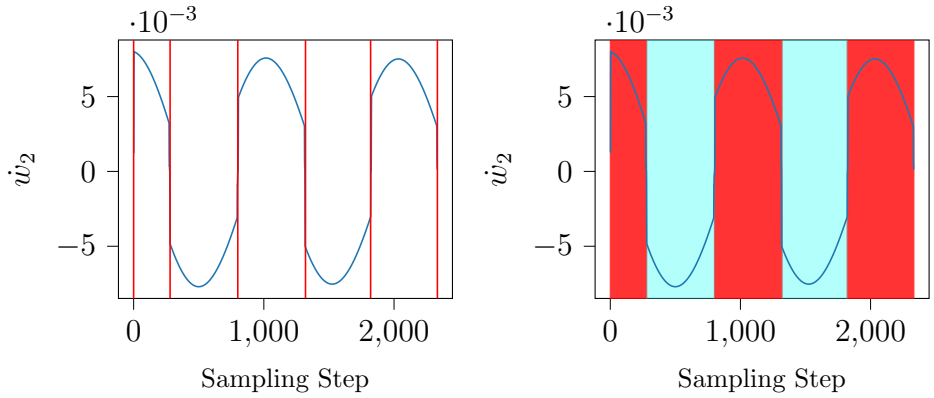


Figure 15: Results for the Detection of Segments and Grouping for the Power Converter

first segment.

6.3. Identification of Modes

In the next step, we analyze the identified groups and expressions. Table 3 shows the results for the example systems. The columns show the detected expressions, thus, providing the number of identified modes, too. Furthermore, the table provides the Mean-Squared-Error (MSE) loss of SR for the learned expressions on the segments of the respective group.

For all examples, the MSE indicates that the learned expressions are close to or exactly match the ground truth expressions. An exact matching is achieved for the two electrical circuits and the Two-tank system with substitution. Still, some differences to the ground truth expressions as introduced in the beginning of the section are present. For example, the bouncing ball example has an additional term $0.006/x(k-1)$ which is not present in the ground truth expression, but due to the small factor 0.006, this term is negligible for the value range of the variable $x(k-1)$. The power converter uses the time t in the second expression which is not used in the ground

truth expression. The sampling steps are very small, i.e., t lies within the range of 10^{-5} and, thus, is negligible compared to the constant 0.01. For the converter, the grouping is correct, which can be seen on the right of Figure 15 by the two colors indicating the two groups.

In general, the identified expressions of the examples involve the most dominant terms of the ground truth expression, but may not exactly represent the less dominant parts. This is visible for the Two-tank system, where the approach is able to identify the dominant influence of the pump defined by the term Q_p/A, while the influence of the water transfer between the two tanks is identified precisely only for the scenario with substitution of the term $\sqrt{|h_1 - h_2|}$. For the scenario without substitution, this is approximated by a linear influence of h_1 which is a useful approximation for the value range of the variables. For the scenario with substitution, we observe that both dynamics are accurately identified. Even though initially three groups are formed, the grouping step finds two of these groups to represent the same dynamics.

The runtime of both, segmentation and grouping steps, as given in Table 2 and Table 4 is moderate. The runtime scales especially with the number of evolutions of the symbolic regression, while a single evolution scales with the amount of learning data. Further, the runtime depends on the number of detected segments and the number of found groups. Thus, the runtime of the grouping step for Circuit 2 and the Bouncing Ball is low, because only a single segment and group exists. The grouping step is often faster than the segmentation step, because during segmentation, the step width of window extensions is small compared to the average segment length. Thus, many

Table 3: Identified Modes

System	Learned Expressions		MSE
Circuit 1 *switched sources*	Group 1 –	$0.5 \cdot U_1$	$3.52 \cdot 10^{-17}$
	Group 2 –	$0.5 \cdot U_2$	0.0
	Group 3 –	$0.5 \cdot U_3$	$3.52 \cdot 10^{-17}$
Circuit 2 *connected sources*	Group 1 –	$0.25 \cdot (U_1 + U_2 + U_3)$	$1.65 \cdot 10^{-16}$
Bouncing Ball	Group 1 –	$2 \cdot x(k-1) - x(k-2) + 0.006/x(k-1)$	$2.11 \cdot 10^{-5}$
Power Converter	Group 1 –	$0.003 - 0.005 \cdot w1$	$6.48 \cdot 10^{-7}$
	Group 2 –	$(t + 0.01) \cdot (w_1 + 0.11 \cdot w_2 + 1.14) \cdot (-0.47)$	$3.63 \cdot 10^{-8}$
Two-tank System *sub.*	Group 1 –	$(-Cv_{vb} \cdot \sqrt{\lvert h_1 - h_2 \rvert} + Q_p)/A$	$6.03 \cdot 10^{-8}$
	Group 2 –	Q_p/A	$1.41 \cdot 10^{-9}$
	Group 3 –	$(-Cv_b \cdot \sqrt{\lvert h_1 - h_2 \rvert} + Q_p)/A$	$1.39 \cdot 10^{-9}$
Two-tank System *w/o sub.*	Group 1 –	$Q_p/A - \sqrt{Q_p} \cdot h_1$	$5.03 \cdot 10^{-6}$
	Group 2 –	Q_p/A	$6.12 \cdot 10^{-6}$
	Group 3 –	$\sqrt{Cv_b} - A$	$3.08 \cdot 10^{-6}$

Table 4: Runtime of the Grouping Step

System	Runtime
Circuit 1	342 s
Circuit 2	85 s
Bouncing Ball	27 s
Power Converter	988 s
Two-tank System *sub.*	3116 s
Two-tank System *w/o sub.*	5749 s

iterations of the internal loop in Algorithm 1 are needed. The parameter l_{step}, thus, is a trade-off between runtime and detection accuracy for transition points.

6.4. Model Construction & Accuracy of Predicted Trajectories

In the last step, we construct a hybrid system in form of a hybrid decision tree. To evaluate the performance of the model, we use the model to predict the output on an evaluation trajectory and compare the prediction to the ground truth trajectory. The evaluation data consists of 1000 to 6000 samples, depending on the example. Table 5 shows the runtime of the model construction for all examples. We observe that the runtime is small as decision tree learning is a lightweight algorithm. In our implementation, we use the classification tree from the scikit-learn library [39].

The prediction accuracy is shown in Table 6 in the column HyDRA, which provides the MSE on the evaluation set using the inference process described in Section 4 as well as the runtime of the model construction, i.e., decision tree learning step.

The MSE on the evaluation set is as expected a bit higher than the MSE on the learning set of the groups in the previous step. Nevertheless, the overall accuracy for all examples is good. The MSE of the Two-tank with substitution is better than the MSE for the Two-tank without substitution and, thus, shows that prior knowledge on relevant sub-dynamics supports the learning process and leads to more accurate models. This effect follows directly from the previous analysis of the identified expressions.

Further, Figure 16 shows the ground truth and the predicted trajectories of the evaluation sets for the converter and the Two-tank example. For the

Table 5: Runtime of the Model Construction

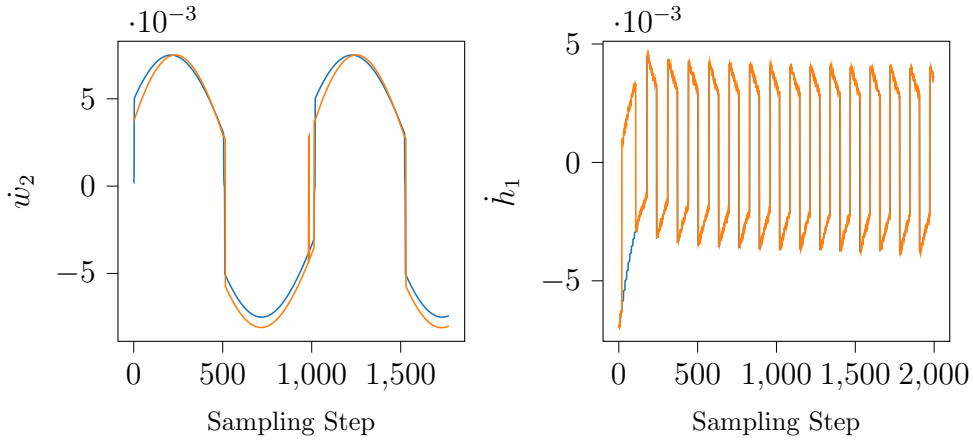
System	Runtime
Circuit 1	0.57 s
Circuit 2	0.14 s
Bouncing Ball	0.51 s
Power Converter	0.28 s
Two-tank System *sub.*	0.42 s
Two-tank System *w/o sub.*	0.68 s

converter, the trajectories match well. There is a single miss-prediction of a mode at around $k = 1000$. For all other samples, the mode is predicted correctly by the hybrid decision tree.

For the Two-tank example, several miss-predictions occur at the beginning of the trajectory. The reason, here, is that due to different initial values of the evaluation and the learning set, the initial phase of the evaluation set differs from the initial phase of the learning set and, thus, can be considered as unseen data during learning. At approximately $k = 150$ the orange trajectory has a discontinuity, realigning with the ground truth trajectory. The rest of the trajectory is predicted accurately.

6.5. Comparison to State-of-the-Art

We compare HyDRA to the state-of-the-art approaches FaMoS [4] and HAutLearn [6]. Both approaches follow the scheme of Figure 1. The transition detection of both approaches executes a peak detection on windowed trajectories. For segment grouping, FaMoS uses a grouping based on Dy-



(a) Power Converter

(b) Two-tank with substitution

Figure 16: Predicted and Ground Truth Evaluation Trajectories

namic Time Warping (DTW), while HAutLearn uses a grouping based on linear matrix inequalities, assuming that system dynamics follow linear differential equations. Consequently, the mode identification of both approaches is based on linear differential equations. Finally, model construction of FaMoS uses decision trees similar to our approach, while HAutLearn uses automata learning based on prefix tree acceptors (PTA).

Both approaches have a set of parameters that need to be tuned. The relevant parameters which are used for this evaluation are shown in Table 7. The parameter w is the window size for the peak detection. The parameter σ is the tolerance for linear matrix inequalities in the grouping step of HAutLearn. The parameters α, ρ_{min}, and ρ_{max} are used for the grouping step of FaMoS. They determine the factor, minimum, and maximum threshold, which is used to determine whether the DTW distance between two segments

Table 6: Prediction Accuracy and Comparison of HyDRA to State-of-the-Art

System	MSE		
	HyDRA	FaMoS	HAutLearn
Circuit 1	$1.07 \cdot 10^{-10}$	-	-
Circuit 2	$2.95 \cdot 10^{-31}$	-	-
Bouncing Ball	$5.15 \cdot 10^{-5}$	1.09	$6.27 \cdot 10^{-1}$
Power Converter	$6.72 \cdot 10^{-7}$	$3.97 \cdot 10^{-6}$	$3.92 \cdot 10^{-6}$
Two-tank System *sub.*	$2.31 \cdot 10^{-6}$	$1.04 \cdot 10^{-7}$	$4.83 \cdot 10^{-6}$
Two-tank System *w/o sub.*	$8.58 \cdot 10^{-6}$	$1.38 \cdot 10^{-7}$	$1.07 \cdot 10^{-7}$

Table 7: Parameters of the State-of-the-Art Approaches

Example	σ	w	α	ρ_{min}/ρ_{max}
Bouncing Ball	$2 \cdot 10^{-5}$	10	2.5	$10^{-3}/10$
Power Converter	$5 \cdot 10^{-3}$	10	2.5	$3 \cdot 10^{-2}/1$
Two-tank System *sub.*	$2 \cdot 10^{-5}$	30	4	$10^{-4}/10$
Two-tank System *w/o sub.*	$2 \cdot 10^{-4}$	30	5	$10^{-4}/10$

is small enough to be grouped together. For additional parameters, we base on the parameters presented in [4].

Table 6 shows the MSE of the prediction on the evaluation set for both approaches and our proposed approach. For the two circuits, FaMoS and HAutLearn are not applicable — indicated by a dash — because the systems are not described by linear differential equations. This shows a limitation of both approaches, which apply only to systems whose dynamics are described by linear differential equations. For the power converter and the Two-tank

Table 8: Segmentation Metric & Number of Identified Modes

System	Identified Modes			Ω_{seg}		
	FaM	HAut	HyDRA	FaM	HAut	HyDRA
Circuit 1	-	-	3	-	-	0.0
Circuit 2	-	-	1	-	-	0.0
Bouncing Ball	4	4	1	10.0	10.0	0.0
Power Converter	2	1	2	9.80	9.80	2.80
Two-tank System *sub.*	3	10	3	6.26	6.26	1.15
Two-tank System *w/o sub.*	2	3	3	3.06	3.06	4.64

systems, HyDRA achieves similar performance in terms of MSE as the state-of-the-art approaches. For the bouncing ball, HyDRA achieves a significantly better MSE than both approaches. This is possible, because HyDRA is able to identify the single mode of the bouncing ball, while both state-of-the-art approaches identify multiple modes.

Table 8 shows the number of identified modes and the segmentation metric Ω_{seg} for the state-of-the-art approaches and HyDRA for the example systems. Our approach achieves a perfect segmentation metric for the first three metrics and a lower segmentation metric for the power converter and the Two-tank system with substitution. Only for the Two-tank system without substitution, HyDRA has a higher segmentation metric than the state-of-the-art approaches, which is the only example, where our approach misses a transition point.

HyDRA consistently finds a small number of modes for the system dynamics, while the state-of-the-art approaches might identify more modes

than necessary. In all examples, the number of modes is determined by the parametrization of the grouping step. While FaMoS relies solely on segment similarity, HAutLearn additionally assesses the compatibility of segments based on linear differential equations. Our approach groups segments based on the underlying dynamics and applies to systems with linear and nonlinear dynamics.

7. Conclusion

In this paper, we present a novel approach to hybrid system identification using symbolic regression and decision trees. We show that symbolic regression can be used to identify the structure of hybrid systems from data. Symbolic regression iteratively adapts to observed dynamics in a trajectory segmentation step and a segment grouping step. The final model construction is done with decision tree learning resulting in a hybrid decision tree. The proposed method, thus, provides a holistic and straightforward approach for the identification of hybrid systems using symbolic regression that opens the scope to a new dynamics-based paradigm for learning models of hybrid systems.

We evaluate our method on four example systems, two of them with two versions: an electrical circuit with two versions, a bouncing ball, a two tank system where there are two different learning scenarios with and without inclusion of prior knowledge, and a power converter. The results show that our method can accurately identify the structure of hybrid systems from data and predict the behavior of the system on new data. Further, the output of the identification process is an explainable and interpretable model which

makes the system understandable to other stakeholders. The model, thus, serves for prediction of future system behavior and can be used for control, fault detection, and other applications.

Acknowledgment

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^{o}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.

During the preparation of this work the authors used GitHub Copilot in order to improve the writing style and identify alternative formulations. After using this tool, the authors reviewed and edited the content as needed and takes full responsibility for the content of the publication.

References

[1] R. Alur, Principles of Cyber-Physical Systems, Technical Report, MIT Press, 2015.

[2] J. Lunze, F. Lamnabhi-Lagarrigue, Handbook of Hybrid Systems Control: Theory, Tools, Applications, Cambridge University Press, Cambridge, 2009.

[3] S. Plambeck, M. Schmidt, G. Fey, A. Subias, L. Travé-Massuyès, Dynamics-based identification of hybrid systems using symbolic regression, in: Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2024, pp. 64–71. doi:10.1109/SEAA64295.2024.00019.

[4] S. Plambeck, A. Bracht, N. Hranisavljevic, G. Fey, Famos– fast model learning for hybrid cyber-physical systems using decision trees, in: International Conference on Hybrid Systems: Computation and Control, Association for Computing Machinery, 2024, pp. 1–10. doi:10.1145/3641513.3650131.

[5] N. Barbosa Roa, L. Travé-Massuyès, V. H. Grisales-Palacio, Dyclee: Dynamic clustering for tracking evolving environments, Pattern Recognition 94 (2019) 162–186.

[6] X. Yang, O. A. Beg, M. Kenigsberg, T. T. Johnson, A framework for identification and validation of affine hybrid automata from input-output traces, ACM Transactions on Cyber-Physical Systems 6 (2022) 1–24. doi:10.1145/3470455.

[7] O. Niggemann, B. Stein, A. Vodencarevic, A. Maier, H. Kleine Büning, Learning behavior models for hybrid timed systems, AAAI Conference on Artificial Intelligence 26 (2012) 1083–1090. doi:10.1609/aaai.v26i1.8296.

[8] S. Plambeck, M. Schmidt, A. Subias, L. Travé-Massuyès, G. Fey, Usability of Symbolic Regression for Hybrid System Identification - System Classes and Parameters, in: International Conference on Principles of Diagnosis and Resilient Systems (DX), 2024, pp. 30:1–30:14. doi:10.4230/OASIcs.DX.2024.30.

[9] S. Plambeck, Symbolic regression 4 hybrid systems, 2024. URL: https://github.com/TUHH-IES/SymbolicRegression4HA.

[10] G. Kronberger, L. Kammerer, M. Kommenda, Identification of dynamical systems using symbolic regression, CoRR abs/2107.06131 (2021). arXiv:2107.06131.

[11] R. Feldt, P. Nordin, Using factorial experiments to evaluate the effect of genetic programming parameters, in: Genetic Programming, 2000, pp. 271–282.

[12] T. Loveard, V. Ciesielski, Genetic programming for classification: An analysis of convergence behaviour, Lecture Notes in Artificial Intelligence 2557 (2002) 309–320.

[13] W. B. Langdon, Genetic programming convergence, in: Genetic and Evolutionary Computation Conference Companion, 2022, pp. 27–28.

[14] B. K. Petersen, M. L. Larma, T. N. Mundhenk, C. P. Santiago, S. K. Kim, J. T. Kim, Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients, in: International Conference on Learning Representations, 2021, pp. 1–26.

[15] K. R. Broløs, M. V. Machado, C. Cave, J. Kasak, V. Stentoft-Hansen, V. G. Batanero, T. Jelen, C. Wilstrup, An approach to symbolic regression using feyn, CoRR abs/2104.05417 (2021). arXiv:2104.05417.

[16] M. R. Muthyala, F. Sorourifar, Y. Peng, J. A. Paulson, Symantic: An efficient symbolic regression method for interpretable and parsimonious model discovery in science and beyond, Industrial & Engineering Chemistry Research 64 (2025) 3354–3369. doi:10.1021/acs.iecr.4c03503.

[17] M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, Science 324 (2009) 81–85. doi:10.1126/science.1165893.

[18] L. Goupil, L. Travé-Massuyès, E. Chanthery, T. Kohler, S. Delautier, Tree based diagnosis enhanced with meta knowledge applied to dynamic systems, IFAC-PapersOnLine 58 (2024) 1–6.

[19] G. Wang, E. Wang, Z. Li, J. Zhou, Z. Sun, Exploring the mathematic equations behind the materials science data using interpretable symbolic regression, Interdisciplinary Materials 3 (2024) 637–657. doi:https://doi.org/10.1002/idm2.12180.

[20] E. Real, C. Liang, D. So, Q. Le, AutoML-zero: Evolving machine learning algorithms from scratch, in: International Conference on Machine Learning, volume 119, Pmlr, 2020, pp. 8007–8019.

[21] S. Gaucel, M. Keijzer, E. Lutton, A. Tonda, Learning dynamical systems using standard symbolic regression, in: M. Nicolau, K. Krawiec, M. I. Heywood, M. Castelli, P. García-Sánchez, J. J. Merelo, V. M. Rivas Santos, K. Sim (Eds.), Genetic Programming, Springer Berlin Heidelberg, 2014, pp. 25–36.

[22] D. L. Ly, H. Lipson, Learning symbolic representations of hybrid dynamical systems, Journal of Machine Learning Research 13 (2012) 3585–3618.

[23] K. S. Fong, M. Motani, Symbolic regression enhanced decision trees for classification tasks, Proceedings of the AAAI Conference on Artificial Intelligence 38 (2024) 12033–12042. doi:10.1609/aaai.v38i11.29091.

[24] S. Plambeck, L. Schammer, G. Fey, On the viability of decision trees for learning models of systems, in: Asia and South Pacific Design Automation Conference (ASP-DAC), 2022, pp. 696–701. doi:10.1109/ASP-DAC52403.2022.9712579.

[25] F. Jimenez, J. Kaminska, E. Lucena-Sanchez, J. Palma, G. Sciavicco, Multi-Objective Evolutionary Optimization for Time Series Lag Regression, in: International Conference on Time Series and Forecasting (ITISE), Universdad de Granada, 2019, pp. 373–384.

[26] M. Tappler, E. Muškardin, B. K. Aichernig, B. Könighofer, Learning environment models with continuous stochastic dynamics, 2023. arXiv:2306.17204.

[27] J. R. Koza, On the programming of computers by means of natural selection, Koza, John R. Genetic programming., MIT Press, Cambridge, Mass. u.a., 1992.

[28] R. Poli, W. B. Langdon, N. F. McPhee, A Field Guide to Genetic Programming, volume 10, Springer, 2008.

[29] M. Cranmer, Interpretable machine learning for science with pysr and symbolicregression.jl, 2023. arXiv:2305.01582.

[30] S. Shalev-Shwartz, S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, Cambridge, 2014. doi:10.1017/CBO9781107298019.

[31] W.-Y. Loh, Classification and regression trees, WIREs Data Mining and Knowledge Discovery 1 (2011) 14–23. doi:10.1002/widm.8.

[32] M. S. Branicky, Introduction to Hybrid Systems, Birkhäuser Boston, Boston, MA, 2005, pp. 91–116.

[33] I. The MathWorks, Model a bouncing ball in continuous time, 2025. URL: \url{https://de.mathworks.com/help/stateflow/ug/modeling-a-bouncing-ball-in-continuous-time.html}.

[34] B. O. Bouamama, R. M. Alaoui, P. Taillibert, M. Staroswiecki, Diagnosis of a two-tank system, Technical Report, Intern Report of CHEM-project, USTL, 2001.

[35] N. Zaupa, L. Martínez-Salamero, C. Olalla, L. Zaccarian, Results on hybrid control of self-oscillating resonant converters, in: IFAC Conference on Analysis and Design of Hybrid Systems (ADHS), volume 54, 2021, pp. 211–216.

[36] N. Zaupa, L. Martínez-Salamero, C. Olalla, L. Zaccarian, Hybrid control of self-oscillating resonant converters, IEEE Transactions on Control Systems Technology 31 (2023) 881–888. doi:10.1109/tcst.2022.3179948.

[37] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, 2019. arXiv:1907.10902.

[38] F. Hutter, H. Hoos, K. Leyton-Brown, An efficient approach for assessing hyperparameter importance, in: International Conference on Machine Learning (ICML), 2014, pp. 754–762.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.