One-Shot Learning in Hybrid System Identification: A New Modular Paradigm

Swantje Plambeck
Hamburg University of Technology, Germany

Maximilian Schmidt
Hamburg University of Technology, Germany

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

Goerschwin Fey
Hamburg University of Technology, Germany

Abstract

Identification of hybrid systems requires learning models that capture both discrete transitions and continuous dynamics from observational data. Traditional approaches follow a stepwise process, separating trace segmentation and mode-specific regression, which often leads to inconsistencies due to unmodeled interdependencies. In this paper, we propose a new iterative learning paradigm that jointly optimizes segmentation and flow function identification. The method incrementally constructs a hybrid model by evaluating and expanding candidate flow functions over observed traces, introducing new modes only when existing ones fail to explain the data. The approach is modular and agnostic to the choice of the regression technique, allowing the identification of hybrid systems with varying levels of complexity. Empirical results on benchmark examples demonstrate that the proposed method produces more compact models compared to traditional techniques, while supporting flexible integration of different regression methods. By favoring fewer, more generalizable modes, the resulting models are not only likely to reduce complexity but also simplify diagnostic reasoning, improve fault isolation, and enhance robustness by avoiding overfitting to spurious mode changes.

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 System, Model Learning, Symbolic Regression

Digital Object Identifier 10.4230/OASIcs.DX.2025.7

Supplementary Material
Software (Source Code): `https://github.com/TUHH-IES/SymbolicRegression4HA` [13]
archived at `swh:1:dir:00ff8c08db289ab78af743dd7ce6b71dceccb37c`

Funding This work is funded by BMBF project AGenC no. 01IS22047A.

1 Introduction

Hybrid systems integrate continuous-time dynamics with discrete transitions and serve as a fundamental modeling paradigm for a broad class of cyber-physical, manufacturing, and embedded systems [1, 8]. The identification of hybrid systems from data is particularly challenging due to the inherent coupling between discrete mode transitions and continuous behavior. However, accurate and compact abstract models are essential for tasks such as system verification, control synthesis, and fault diagnosis. Compactness is especially beneficial for diagnosis, as it can streamline fault isolation, lower the computational load of diagnostic reasoning, and mitigate the risk of ambiguity or overfitting to spurious or non-informative mode changes.

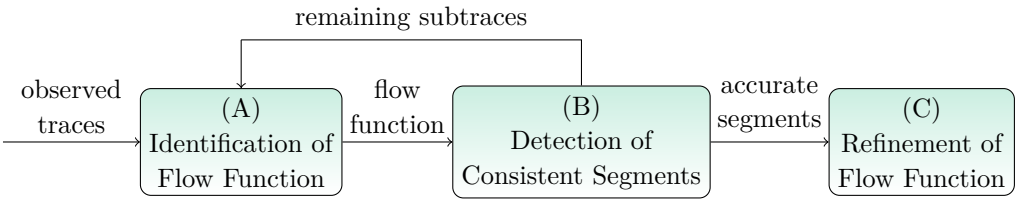


Figure 1 Illustration of the Proposed Learning Paradigm.

A hybrid system comprises a finite set of discrete modes, each associated with a distinct continuous flow function. Transitions between modes are governed by discrete events, defined through guard conditions, which depend on the state of the system or external inputs [3].

Data-driven identification aims to reconstruct the underlying hybrid model from observed traces of the system. Traditional approaches to hybrid system identification typically adopt a stepwise procedure [14]. First, the observed traces are segmented and grouped into regions presumed to be generated by the same dynamic mode. Subsequently, regression techniques are applied to fit a continuous flow function to each region representing the grouped segments. However, this decoupled approach suffers from several limitations. Independent optimization of the segmentation and regression often leads to inconsistencies, as the learned components do not account for their mutual dependencies. In particular, inaccurate segmentation can degrade the quality of the estimated dynamics, resulting in inaccurate models and poor performance.

In our previous work, we propose a learning approach using symbolic regression [15, 16]. This approach identifies segments directly based on changes in the system dynamics. For this purpose, symbolic regression is used, which allows adapting to given dynamics in a flexible way by iterating on a flow function. Now, we present a new procedure that elaborates on this idea comprising the overall learning paradigm of hybrid system identification. A major advantage of our previous work is the detection of transition points based on a first estimate of the flow function for the system dynamics. This encourages the development of a new learning paradigm that, instead of resetting an identification, leverages a flow function identified on a given segment to segment the remaining observed traces in one shot by finding the consistent segments. In contrast to the previous work, this paradigm is independent of the regression method used to identify the flow functions, i.e., the method does not restrict to symbolic regression as presented in Plambeck et al. [15, 16].

We present a discussion on the learning process for hybrid systems facing interdependencies in the learning process. With that, we provide a formalization of the underlying optimization problem of data-driven identification for hybrid systems and revisit the traditional idea of stepwise learning in this context. In addition, we propose an iterative learning paradigm for hybrid systems that takes account the original optimization problem. Figure 1 illustrates the proposed procedure. The approach begins by identifying a continuous flow function from an initial segment of a trace (A). The flow function from Step (A) is then evaluated against the remaining observed traces (B). Segments where the flow function exhibits sufficient predictive accuracy are consistent with respect to the flow function and they are grouped into a mode (C), while the remaining subtraces are treated as candidates for further mode identification (A). This procedure is repeated until the complete trace is adequately explained by one of the modes in the model. Step (C) finally refines the found flow function on all consistent segments of the dynamic mode. To determine an appropriate initial segment, we

employ a windowing strategy that increases the length of the segment until the error of the prediction by the corresponding flow function is above a specified threshold. This strategy maximizes segment size under a fixed accuracy constraint.

The proposed method offers several advantages over traditional techniques. By jointly optimizing segmentation and model identification in an iterative loop, the algorithm exploits the interdependencies among the segments of traces and flow functions to guide the learning process. The model is constructed incrementally, with new modes introduced only when existing ones fail to capture the observed dynamics. This results in compact models characterized by a minimal number of modes as well as large and coherent segments.

Our empirical results demonstrate that our approach has a flexible and modular design for the integration of regression methods in the identification of flow functions. In addition, our learning strategy identifies models with a smaller number of modes compared to a traditional learning strategy, which is very well illustrated on a bouncing ball example.

The remainder of the paper is structured as follows: Section 2 reviews related work on hybrid system identification. In Section 3, we introduce the necessary preliminary concepts and notation. In Section 4, we present our learning approach in detail. Section 5 presents an empirical evaluation of the proposed approach. We conclude the paper in Section 6.

2 Related Work

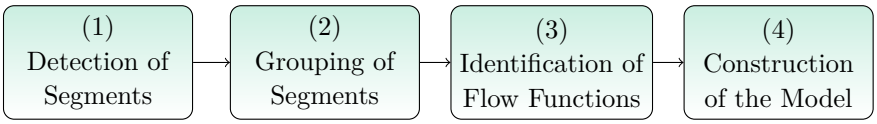


Figure 2 Traditional Approach to Hybrid Systems Identification.

Most existing approaches to hybrid system identification adopt a sequential pipeline, as illustrated in Figure 2. In contrast to our approach, these methods defer mode identification – i.e., the estimation of flow functions – after the segmentation of traces and the grouping of segments. This introduces two intermediate representations (segments and segment groups) that are constructed without directly considering the final goal of mode identification.

In this traditional process, the first step is the detection of transition points (1), which partition the observed traces into segments. For the detection of transition points, existing methods use a similarity measure, based on the distance between samples [2, 19], in the frequency domain [11], or based on slices of observations [21]. However, these methods often overlook the fact that complex trajectories generated by the same dynamics can occupy different regions of the state space. In contrast, our approach avoids explicit transition detection and instead identifies segments based on the consistency with a belief of the underlying flow function.

The second step (2) involves grouping the segments that are presumed to belong to the same mode. Common techniques for this step include clustering [2] and similarity-based grouping [18]. Ly et al. [9] propose a clustering approach based on symbolic regression to infer the structure of modes.

Finally, in step (3), flow functions are then identified for each group of segments. This is typically done using regression techniques, including linear regression [21], polynomial regression [18], or neural networks [11]. Often, existing methods restrict to a specific class of functions, e.g., linear differential equations [21] or polynomials [18], or types of regression models such as neural networks [11]. Several approaches are further limited to a single

regression method, such as linear regression or symbolic expressions [9]. Our method, on the contrary, is agnostic to the choice of the regressor and can incorporate various regression models, as demonstrated in our evaluation.

The final step (4) is the construction of the complete hybrid system. This is done by combining the identified flow functions into modes and forming transitions between them. Transition conditions are typically learned in a separate step [18] or identified during the construction of the hybrid system [11, 5].

3 Preliminaries

This section introduces the foundational concepts used throughout the paper. We start with the definition of hybrid systems, which are the target of our learning approach. We then introduce the system under learning and the traces, that we observe on this system.

3.1 Hybrid Systems

Hybrid systems combine continuous dynamics with discrete transitions, representing complex behaviors that are common in many real-world systems.

▶ **Definition 1** (Hybrid System [3]). *A Hybrid System* Γ *is defined by a 5-tuple* $(X, Q, \mathcal{F}, \Sigma, \mathcal{T})$ *where:*

- $X = \{x_1, x_2, \ldots, x_n\} = I \cup S$ *is the set of system variables, consisting of input variables* I *and state variables* S*.*
- Q *is the finite set of discrete modes.*
- $\mathcal{F}$ *is the set of flow functions. Each flow function* $f_q \in \mathcal{F}$ *defines the continuous flow, i.e., dynamics within mode* $q \in Q$ *by specifying the outputs as a function of the current state and inputs, i.e.,* $O = f_q(S, I)$*, where* $O \subseteq S$ *are the output variables of the system.*
- Σ *is a set of events. Each event is guarded by a condition* $c : X \to \{true, false\}$ *on the variables in* X*. An event is active if the condition evaluates to true.*
- $\mathcal{T} : Q \times \Sigma \to Q$ *defines transitions between modes* Q*. A transition is triggered if the corresponding event* $\sigma \in \Sigma$ *is active.*

This definition combines continuous behavior with discrete transitions. The continuous behavior is governed by mode-specific flow functions, whereas transitions between modes are triggered by the guard conditions on system variables, including external control signals.

Throughout this work, we assume that flow functions are time-invariant, i.e., the functions in $\mathcal{F}$ depend only on the current state and input values, not explicitly on the time. However, time-dependent behavior can be modeled by introducing time-shifted versions of state variables as an additional input to the flow function.

3.2 System & Observations

The system under learning is observed through a finite collection of execution traces $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_m\}$. Each trace $\mathcal{O}_j$ consists of a time series of sampled variable vectors from X. Formally, $\mathcal{O}_j[i] = [x_1, x_2, \ldots, x_n]$ denotes the ith sample in trace $\mathcal{O}_j$, where $j = 1, \ldots, m$ and $i = 1, \ldots, k_j$ with k_j being the length of the jth trace.

The goal is to identify a hybrid system Γ that explains the observed traces, capturing both the continuous dynamics and the discrete mode transitions of the underlying system behavior.

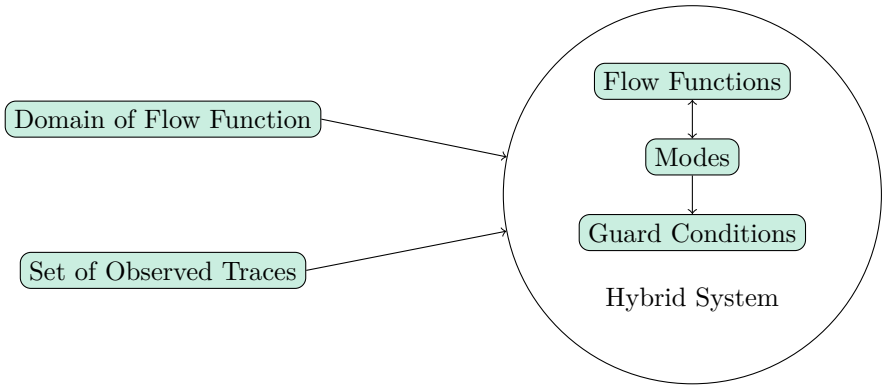


Figure 3 Illustration for the Problem of Hybrid System Identification.

4 A New Paradigm for Hybrid System Identification

Identifying hybrid systems from data is challenging due to the inherently complex structure. A complete model must capture discrete modes, continuous dynamics (flow functions), and the conditional transitions that govern mode switching. We analyze the goal of hybrid system identification and propose a formalization of the learning problem. Afterward, we discuss the limitations of traditional approaches to hybrid system identification with respect to the original learning problem. Finally, we present our new paradigm for hybrid system identification that integrates the learning of flow functions and segmentation into a unified framework.

4.1 Problem Definition and Objectives for Hybrid System Identification

The learning process for hybrid system identification starts with a set of observed traces. Additionally, a domain Φ of candidate flow functions is given at the beginning of the learning process. The set of candidate flow functions Φ contains all flow functions that can be used to model the continuous dynamics of the system and is, e.g., defined by the application. The goal is to identify a hybrid system Γ that explains the observed traces using a minimal number of flow functions. As illustrated in Figure 3, the task is framed as a multi-objective optimization problem: minimizing the model complexity (e.g., the number of modes or flow functions) while maximizing the accuracy of the learned system on the observed data. We define a required accuracy by a threshold ϵ for the deviation between the observed traces and the identified hybrid model. This reduces the learning problem to a minimization of the number of flow functions under this accuracy constraint:

$$\min_{\substack{\mathcal{F}\subset\Phi,\\ \mathcal{S}\in\mathcal{P}(\mathcal{O})}} |\mathcal{F}| \quad \text{s.t.} \quad E(\mathcal{F},\mathcal{S}) \leq \epsilon, \tag{1}$$

where E denotes an error function that evaluates the deviation between the behavior found from the flow functions $\mathcal{F}$ on the segments $\mathcal{S}$ composing a segmentation from the set $\mathcal{P}(\mathcal{O})$ of all possible segmentations of the observed traces $\mathcal{O}$. In this formalization, a minimal exact solution, i.e., $\epsilon = 0$ exists only if the observed traces are free of noise and the domain of candidate flow functions Φ covers the original system dynamics.

4.2 Traditional Identification of Hybrid Systems

The traditional approach to learning hybrid systems uses a sequential procedure as shown in Figure 2:

1. Transition detection: Identify potential transition points in each trace.
2. Segment grouping: Cluster segments between transitions into groups that are assumed to represent the same mode.
3. Flow function identification: Learn one flow function per segment group.
4. Model construction: Construct the hybrid system by combining the learned functions and define transitions.

In this approach, the individual steps are independent of each other and can be formulated as follows. First, the set $\mathbf{t}$ of transition points is identified as

$$\mathbf{t} = \{(j, i) \mid \varphi(\mathcal{O}_j[i]) \text{ is true}\}, \tag{2}$$

where φ is a predicate for transition detection. Every transition point is represented by a tuple (j, i) with j identifying the trace O_j and i giving an index i in this trace. Using the transition points, all traces are cut into segments $\mathcal{S}$:

$$\mathcal{S} = \bigcup_{j \in [1,\dots,m]} \{\mathcal{O}_j[i_k : i_{k+1}] \; \forall \; [(j, i_k), (j, i_{k+1})] \in \mathbf{t}\}, \tag{3}$$

where $\mathbf{t}$ is sorted such that $i_0 < i_1 < \cdots < i_{k_j}$. $\mathcal{S}$ is one possible segmentation of the traces, i.e., $S \in \mathcal{P}(\mathcal{O})$, based on the identified transition points. Segments form a group g, if they are sufficiently similar, i.e.,

$$g = \bigcup \{\hat{\mathcal{S}} \subseteq \mathcal{S} \mid \sigma(\hat{\mathcal{S}}) \geq \tau\}, \tag{4}$$

where σ is a similarity function used to compare segments and τ is a similarity threshold. Note that similar dynamics do not necessarily create similar traces, which is one issue that our approach addresses. Every segment is assigned to a single group only, i.e., $g_i \cap g_j = \emptyset \; \forall i \neq j$. The groups form the set

$$G = \{g_1, \dots, g_{|\mathcal{F}|}\}, \tag{5}$$

such that every group contains the learning data for one flow function. The set of flow functions $\mathcal{F}$ is found as follows:

$$\mathcal{F} = \{\arg\min_{f \in \Phi} e(f, g) \; \forall g \in G\}, \tag{6}$$

where $e(f, g)$ is the approximation error of function f on group g, e.g., the mean squared error between the output of the flow function and the observed output on the learning data.

The process defined by Equation (2) to Equation (6) is problematic as the identification of flow functions is postponed to the last step. Segments and groups form intermediate learning results that are not informed by the final goal of identifying accurate and compact flow functions, which is not inline with the original learning problem in Equation (1). This separation of concerns introduces strong interdependencies: transition points are only meaningful in light of changes in dynamics and accurate flow functions can only be learned from meaningful segments.

4.3 Proposed Identification Process for Hybrid Systems

To address these limitations, we propose a new learning framework that integrates flow function identification into the segmentation and grouping process. Rather than treating the components of the hybrid system as separate inference steps, our method, as shown in Figure 1, alternates between the identification of flow functions (A) and the detection of segments (B) using partial model information to guide the refinement of each component. This iterative approach directly targets the optimization problem in Equation (1) by prioritizing minimality: new modes are introduced only when the existing set of flow functions cannot explain the data with sufficient accuracy. This design aligns the learning process with the goal of minimality in model structure. Our approach does not guarantee finding an optimal solution for Equation (1). Instead, we optimize the length of individual segments under the constraint that a single flow function represents the complete segment. With this procedure, the number of segments needed to cover all traces is reduced. Finally, a smaller number of segments facilitates a smaller number of flow functions.

Currently, our framework focuses on the identification of flow functions and their associated modes. The inference of transition conditions is treated as a separate post-processing step, to be learned independently once the mode structure is established.

Our iterative learning approach is described as follows:

A Identification of Flow Function: Find a segment of maximal length starting from a first trace such that a flow function exists, that achieves a sufficient accuracy on this segment. Identify the flow function for the found segment.
B Detection of Accurate Segments: Assess the accuracy of the identified flow function on all observed traces.
If accuracy sufficient: **Stop**, and finalize the modes.
If accuracy insufficient:
C Refinement of Flow Function: Refine the flow function on segments with sufficient accuracy and keep this flow function for the mode.
Back to A Continue with the remaining subtraces, returning to Step A.

At each iteration on the set of traces, a single flow function is learned and then used to identify segments where the flow function is valid, i.e., achieves sufficient accuracy and subtraces where the accuracy of the flow function is insufficient. The data for all segments with sufficient accuracy is used to refine the flow function for this mode. The process repeats on the remaining data until all segments have been assigned to a mode.

In relation to Figure 1, we formulate one iteration of the approach. The identification of a flow function (A) is given as

$$\tilde{f} = \arg\min_{f \in \Phi} e(f, \mathcal{O}_j[0, l^*]), \tag{7}$$

where $e(f, \mathcal{O}_j[0, l^*])$ is the approximation error which is determined for the function $\tilde{f}$ on the segment $\mathcal{O}_j[0, l^*]$ of length l^* of an arbitrary trace $\mathcal{O}_j \in \mathcal{O}$. The length l^* is maximized in an iterative manner together with the identification of the flow function f:

$$l^{(k)} = \begin{cases} l_{\text{init}} & \text{if } k = 0, \\ \max\{n \leq k_j \mid e(f^{(k-1)}, \mathcal{O}_j[0, n]) \leq \epsilon\} & \text{if } k > 0, \end{cases} \tag{8}$$

$$f^{(k)} = \arg\min_{f \in \Phi} e(f, \mathcal{O}_j[0, l^{(k)}]), k \geq 0, \tag{9}$$

where given a flow function $f^{(k-1)}$ from the previous iteration, the length $l^{(k)}$ is determined as the maximum length of a segment where the flow function $f^{(k-1)}$ achieves sufficient accuracy. The flow function $f^{(k)}$ is then learned on this segment of length $l^{(k)}$. The parameter l_{init} is the initial length of the segment, and k_j is the length of the trace $\mathcal{O}_j$. The segment length l^* is the fixed point of Equation (8) and Equation (9), i.e., where $l^{(k)} = l^{(k+1)}$. This is the maximum length of a segment that achieves a sufficient accuracy on the trace $\mathcal{O}_j$.

The detection of accurate segments (B) finds $\mathcal{S}_A$ as a set of segments from the traces in $\mathcal{O}$ where the function $\tilde{f}$ achieves sufficient accuracy:

$$\mathcal{S}_A = \bigcup_{j \in m} \{\mathcal{O}_j[i] \mid e(\tilde{f}, \mathcal{O}_j[i]) < \epsilon \text{ with } i \in [1, ..., k_j]\}. \tag{10}$$

Finally, we use $\mathcal{S}_A$ to refine a final flow function f (C) which represents a new dynamic for the segments of $\mathcal{S}_A$:

$$f = \arg\min_{f \in \Phi} e(f, \mathcal{S}_A). \tag{11}$$

The next iteration of the algorithm starts on an updated set of observations

$$\mathcal{O} = \bigcup_{j \in m} \{\mathcal{O}_j[i] \mid e(f, \mathcal{O}_j[i]) \geq \epsilon \text{ with } i \in [1, ..., k_j]\}. \tag{12}$$

In this set, all subtraces of the traces in $\mathcal{O}$ where the flow function f does not achieve sufficient accuracy, form input traces for the next iteration. These are the subtraces that are not included in the set of segments $\mathcal{S}_A$.

The iterative process continues until all traces in $\mathcal{O}$ have been covered, i.e., $\mathcal{O} \equiv \emptyset$.

Compared to the traditional approach (Equation (2)–Equation (6)), this formulation eliminates the explicit representation of transition points and segment groups. Instead, transitions emerge naturally from the explanatory limits of each flow function on the data. Mode identification and segmentation are no longer treated as independent pre-processing steps but are tightly coupled through accuracy-driven inference. The condition for the detection of a transition point (formulated as φ in the traditional approach) is based directly on the accuracy of the identified flow function. Consequently, this procedure addresses the original problem in Equation (1) more directly than the traditional approach.

4.4 An Algorithmic Approach to the Proposed Method

We concretize the learning process in Algorithm 1, which describes the steps of the learning process.

The algorithm begins with an arbitrary trajectory from $\mathcal{O}$. For the sake of reproducibility, we choose the first trace $\mathcal{O}_0$ in Line 2. A belief flow function $\tilde{f}$ is learned over a growing time window starting at the beginning of $\mathcal{O}_0$ with an initial length of l_{init} in the loop from Line 4 to Line 8. The window is expanded in increments of size l_{step} until the approximation error exceeds the threshold ϵ. The resulting function $\tilde{f}$ is assumed to characterize a new dynamic. All trace segments where $\tilde{f}$ achieves an error below ϵ are considered additional instances of this dynamic and are used to find a final f for the found mode. This set $\mathcal{S}_A$ of segments is determined by the function *getAccurateSegments* in Line 9. The set $\mathcal{S}_A$ is then used to refine the flow function in Line 10, which identifies the new dynamic. Afterward, all identified segments are removed from the traces in Line 12 which reduces existing traces or splits traces into multiple new traces. Figure 4 visualizes the effect of the function *removeAccurateSegments*. Starting from the traces $\mathcal{O}_1$, $\mathcal{O}_2$, and $\mathcal{O}_3$, the segments where the

Algorithm 1 Identification of Hybrid Systems.

Data: $\mathcal{O}$
Result: $\mathcal{F}$
while $\mathcal{O} \neq \emptyset$ **do**
  $\mathcal{O}_0 \leftarrow \mathcal{O}[0]$;
  $i_{end} \leftarrow l_{init}$; $e \leftarrow 0.0$;
  while $i_{end} < |\mathcal{O}_0| \ \wedge \ e < \epsilon$ **do**
    window $\leftarrow \mathcal{O}_0[0, i_{end}]$;
    $\tilde{f}, e \leftarrow$ learnFlowFunction(window);
    $i_{end} \leftarrow i_{end} + l_{step}$;
  end
  $\mathcal{S}_A =$ getAccurateSegments$(\mathcal{O}, \tilde{f}, \epsilon)$;
  $f \leftarrow$ refineFlowFunction$(\mathcal{S}_A)$;
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{f\}$;
  $\mathcal{O} \leftarrow$ removeAccurateSegments$(\mathcal{O}, \mathcal{S}_A)$;
end

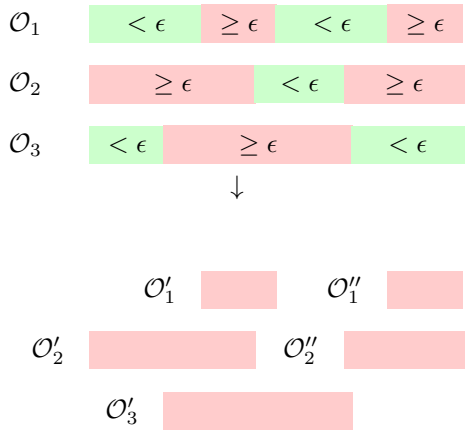


Figure 4 Segmentation using the Belief Flow Function.

error is smaller than ϵ are removed. These segments are covered by the flow function f. The new set of traces is $\mathcal{O} = \{\mathcal{O}'_1, \mathcal{O}''_1, \mathcal{O}'_2, \mathcal{O}''_2, \mathcal{O}'_3\}$. The process repeats until all observed data have been covered, i.e., $\mathcal{O} \equiv \emptyset$.

5 Empirical Analysis

This section presents an empirical evaluation of our learning approach as introduced in Algorithm 1. The approach is independent of the specific regression method used for *learnFlowFunction* and *refineFlowFunction*. To demonstrate this flexibility, we apply our learning approach with three distinct regression methods, each operating over different classes of flow functions. The results illustrate the versatility of the approach across diverse regression methods and hybrid system types.

The structure of this section is as follows: we first describe the learning methods employed, followed by a description of the evaluation examples and, finally, a presentation and discussion of the results.

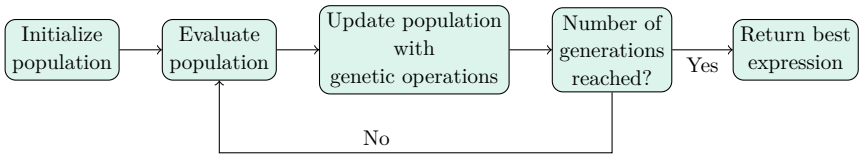


Figure 5 Genetic Programming for Symbolic Regression.

5.1 Learning Methods

The learning approach presented in this paper is modular and allows the integration of different regression methods for the identification of flow functions. We consider three methods during evaluation of our learning approach:

- Linear regression (LR) for learning simple, interpretable linear flow functions.
- Symbolic regression (SR) to identify more expressive symbolic representations.
- Linear matrix (LM) differential equations for systems with multidimensional state spaces.

For symbolic and linear regression, we restrict our evaluation to hybrid systems with a single output variable that corresponds directly to the state variable, i.e., $O = S$ and $|O| = 1$. The third regression method, i.e., linear matrix differential equations, applies to a multidimensional state.

Linear Regression

Linear regression learns flow functions in the form of affine linear mappings: $o = \mathbf{w}^T\mathbf{i} + b$, where $\mathbf{i}$ is the input vector, $\mathbf{w}$ is the learned weight vector, and b is the bias term. This method offers high computational efficiency and interpretability, but is limited in expressiveness due to the restriction to linear functions. We use the implementation provided by the `scikit-learn` library [12], which solves the regression problem using ordinary or non-negative least squares.

Symbolic Regression

Symbolic regression aims to learn expressions of the form $o = r(\mathbf{i})$, where r is a symbolic expression composed of a predefined set $\mathcal{B}$ of basic functions and operators [6]. Candidate expressions are evaluated based on a fitness metric that balances accuracy in the training data and expression complexity. A regularization term – weighted by a parsimony coefficient ρ – is typically included to prevent expression bloat [17].

In our experiments, we use the PySR framework [4], which implements symbolic regression via genetic programming. The algorithm maintains a population of size p of expressions, which evolves over multiple generations through selection, mutation, and crossover. Figure 5 illustrates this evolutionary process.

Linear Matrix Differential Equations

With a third regression method, we model time-continuous, multidimensional linear models, specifically linear matrix differential equations. These are common in system identification, particularly in ARX models [7], and are of the form:

$$\dot{\mathbf{y}}[t] = \mathbf{A}\mathbf{y}[t] + \mathbf{B}\mathbf{u}[t], \tag{13}$$

where $\mathbf{y}[t]$ and $\mathbf{u}[t]$ denote the state and input vectors, respectively, and $\mathbf{A}$ and $\mathbf{B}$ are the system and input matrices.

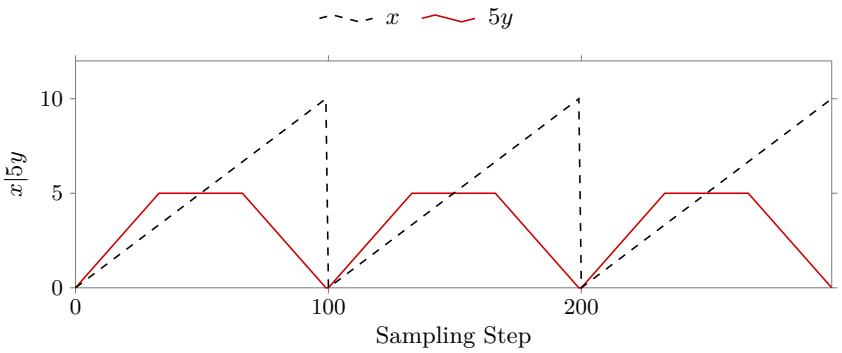


Figure 6 Piecewise Linear Function with Three Segments. The value of y is scaled for better visibility.

Given a set of observed traces $\mathcal{O}$, the goal is to identify the matrices $\mathbf{A}$ and $\mathbf{B}$ that best explain the observed dynamics. This is achieved by solving the following least-squares optimization problem:

$$\min \|\mathbf{Y}[t+1] - \mathcal{A}\mathbf{Y}[t]\|, \tag{14}$$

where $\mathbf{Y}[t]$ is constructed from all observed state and input vectors at time t:

$$\mathbf{Y}_{\mathcal{O}}[t] = \begin{bmatrix} \mathcal{Y}_1[t] & \cdots & \mathcal{Y}_m[t] \\ \mathcal{U}_1[t] & \cdots & \mathcal{U}_m[t] \end{bmatrix}. \tag{15}$$

The least squares solution is obtained using the linear regression implementation of `scikit-learn` [12].

Traditional Baseline Method

In addition to the analysis of the presented learning approach, we compare to the results of hybrid system identification with FaMoS [14]. FaMoS follows the traditional approach to hybrid system identification as given in Figure 2. The approach uses the Euclidean distance of windows of previous samples and upcomig samples of a trace for every sampling step for the segmentation of traces in Step (2). The segments are then grouped based on the similarity of the segments in Step (3) using dynamic time warping [10]. Flow functions are identified as linear matrix difference equations in Step (3).

5.2 Examples

We illustrate our learning approach with two representative examples: a piecewise linear function and a bouncing ball system. The first example serves as a simple baseline that allows a straightforward identification of the flow functions. The second example, the bouncing ball, is a classical benchmark in system identification. Although the bouncing ball exhibits discontinuities in velocity, it can be modeled using a single flow function. In addition, we demonstrate the applicability of different learning methods for both examples and compare identified flow functions with the original system dynamics.

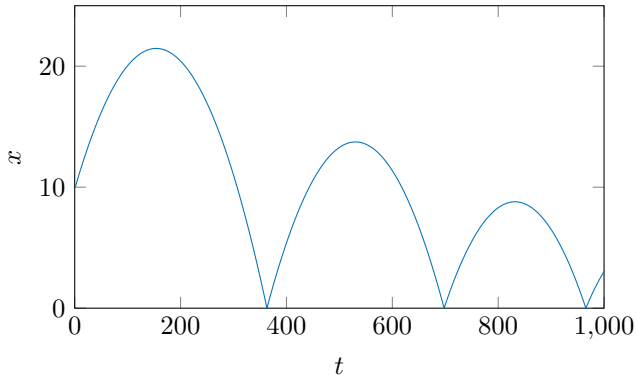


Figure 7 Bouncing Ball with Height over Time.

Piecewise Linear Function

As a simple baseline, we consider a piecewise linear function with three segments. The function is defined as follows:

$$f(x) = y = \begin{cases} 0.3 \cdot x & x < \frac{10}{3} \\ 1 & \frac{10}{3} \leq x < \frac{20}{3} \\ -0.3 \cdot x + 3 & \frac{20}{3} \leq x < 10. \end{cases} \tag{16}$$

The observation of the output variable $y = f(x)$ and the input variable x over time is shown in Figure 6. The system is excited with a repeating input signal x, which is increasing and resetting linearly.

Bouncing Ball

The bouncing ball is a well-known example for system identification. 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 [20] with a sampling time of $\Delta t = 0.01s$. The height x of the ball observed over time is shown in Figure 7.

The system dynamics are governed by the matrix differential equation:

$$\begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} + \begin{pmatrix} 0 \\ -g \end{pmatrix}, \tag{17}$$

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 discrete difference equation as follows:

$$\begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x(t-1) \\ v(t-1) \end{pmatrix} + \begin{pmatrix} 0 \\ -g \end{pmatrix} \cdot \Delta t. \tag{18}$$

For model learning with symbolic regression and linear regression, one dimensional flow functions are learned. Thus, the height of the ball over time is approximated by the height of the ball at the previous sampling step and the height at the second-last sampling step. We provide a time history and approximate the height of the ball as follows:

$$x(t) = 2 \cdot x(t-1) - x(t-2) - \frac{g}{2} \cdot \Delta t. \tag{19}$$

Table 1 Learning Parameters for Hybrid System Identification (Algorithm 1) and Symbolic Regression.

Example	Algorithm 1			Symbolic Regression		
	ϵ	l_{init}	l_{step}	ρ	n	p
Piecewise Linear	0.01	5	1	0.9	10	31
Bouncing Ball	0.5	10	10	0.9	10	31

Table 2 Results of Model Learning.

Example	Method	Flow Functions	MSE	Learning Time in s
Piecewise Linear	LR	$\begin{cases} f_1(x) &= 0.3 \cdot x - 2.2 \cdot 10^{-16} \\ f_2(x) &= 1.0 \\ f_3(x) &= -0.3 \cdot x + 3.0 \end{cases}$	$2.198 \cdot 10^{-32}$	0.159
	SR	$\begin{cases} f_1(x) = & 0.3 \cdot x \\ f_2(x) = & 0.5 + 0.5 \\ f_3(x) = & 3.066 - (-1.264) \cdot \\ & (-0.237) \cdot (x + 0.220) \end{cases}$	$1.192 \cdot 10^{-15}$	568.016
Bouncing Ball	LR	$\begin{cases} f_1(x_1, x_2) = & 1.989 \cdot x_1 - 0.989 \cdot x_2 \\ & +0.003 \end{cases}$	$1.99 \cdot 10^{-4}$	0.268
	SR	$\{ f_1(x_1, x_2) = x_1 + x_1 - x_2$	$2.025 \cdot 10^{-4}$	570.75
	LM	$\begin{cases} f_1(x, v) = & \begin{pmatrix} 1 & 0.001 \\ -0.023 & 0.984 \end{pmatrix} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} \\ & + \begin{pmatrix} 0.001 \\ 0.225 \end{pmatrix} = \begin{pmatrix} x(t+1) \\ v(t+1) \end{pmatrix} \end{cases}$	$\begin{pmatrix} 8.749 \cdot 10^{-3} \\ 4.573 \cdot 10^{-2} \end{pmatrix}$	0.038

Setup & Parameters

Table 1 lists the parameters used in the experiments. In our learning approach from Algorithm 1, ϵ denotes the error threshold, l_{init} the initial segment length, and l_{step} the increment of the segment length. For symbolic regression, ρ is the parsimony coefficient, n the number of generations, and p the population size.

5.3 Identification with Linear Regression

We apply our learning approach with linear regression to both the piecewise-linear function and the bouncing ball system. As shown in Table 2, three distinct flow functions are identified for the piecewise linear function, each corresponding to one mode. The learned functions match the true system except for a minor difference in the first flow function. The maximum length of the found segments is $l^* = 33$, which is equivalent to the sampling step of the ground truth transition point.

Similar results are obtained for the bouncing ball. Here, the error is slightly larger, but also a larger error threshold ϵ is used. The learned flow function differs in the slope and offset from the original function. The linear function based on the heights x_1 and x_2 of the last and second-last sampling step is learned, respectively. This is an approximation of

the actual differential equation of the bouncing ball. Due to the high sampling rate, the approximation is very close to the actual differential equation. This is also shown in Figure 8. Figure 8a shows the learned flow function compared to the ground truth. The learned flow function is close to the ground truth. Figure 8b shows the absolute error of the learned flow function compared to the ground truth. The error is small, but varies over time due to the mismatch of the slope and offset of the learned flow function compared to the ground truth. Larger errors occur at the transition points, where the ball bounces on the ground and the simulation resolution is not sufficient to capture the dynamics of the bouncing ball.

Further, the variant of our approach with linear regression has a very low learning time.

5.4 Identification with Symbolic Regression

Symbolic regression offers greater flexibility compared to linear regression, enabling the identification of a wider class of functions depending on the available operators and basis functions. For both the piecewise-linear function and the bouncing ball, we use addition, subtraction, and multiplication as primitives. In addition, symbolic regression identifies constants for the learned expressions.

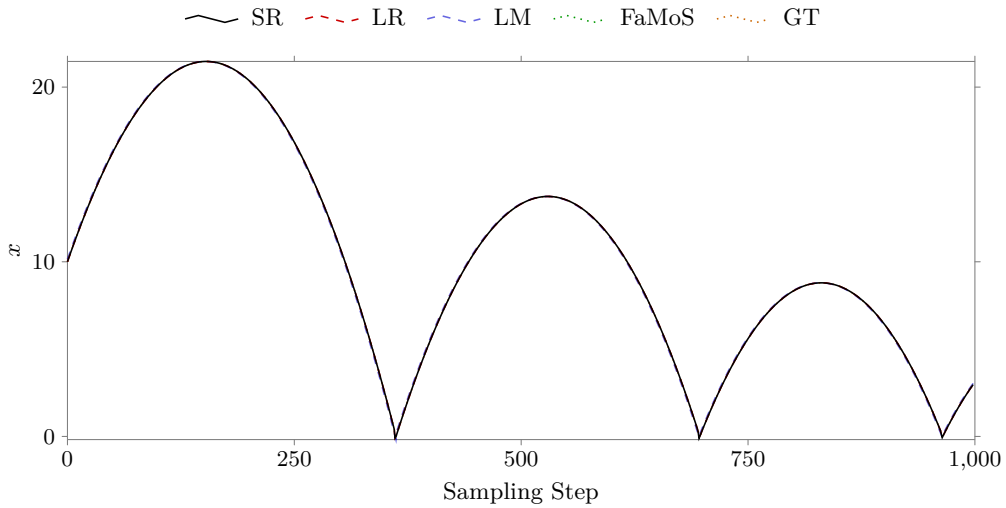
The results of the model learning are shown in Table 2. For the piecewise-linear function, we learn three flow functions, which are the three modes of the piecewise-linear function. The first two flow functions are equivalent to the original function. However, instead of 1 for the second flow function, $0.5 + 0.5$ is learned. Symbolic regression does not simplify expressions internally, as this could hinder exploration of the search space in the genetic algorithm. We simplify the expression after learning, which results in the following set of flow functions:

$$\begin{cases} f_1(x) & = 0.3 \cdot x \\ f_2(x) & = 1.0 \\ f_3(x) & \approx 3.0 - 0.3 \cdot x \end{cases} . \tag{20}$$

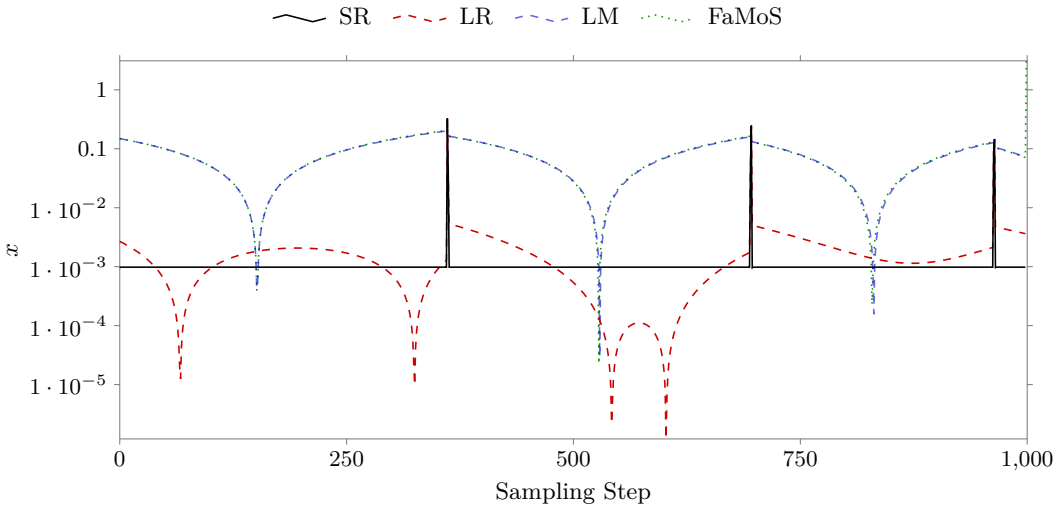
The slope and offset of the third flow function deviate only in the range 10^{-8} from the original function. Again, the maximum length of the found segments is $l^* = 33$, which is equivalent to the ground truth transition point.

Similar results are obtained for the bouncing ball. The coefficients for the features x_1 and x_2 are learned correctly, although $x_1 + x_1$ is learned instead of $2 \cdot x_1$. Compared to the original function, the constant offset is not learned. However, the mean squared error is close to the error of the linear regression. Figure 8a shows the learned flow function compared to the ground truth. The learned flow function is close to the ground truth. Figure 8b shows the absolute error of the learned flow function compared to the ground truth. The error is small for both linear regression and symbolic regression, and the error of linear regression is on average smaller than the error of symbolic regression. The error with symbolic regression is constant because only the constant offset is not learned correctly. Larger errors occur at the transition points, where the ball bounces on the ground and the simulation resolution is not sufficient to capture the dynamics of the bouncing ball.

Overall, linear regression achieves lower errors and significantly faster learning time than symbolic regression. This is due to the larger search space Φ of symbolic regression and its dependence on heuristic optimization (here, genetic programming), in contrast to the closed-form solution used in linear regression.



(a) Bouncing Ball Height x for our Learning Approach with Linear Regression (LR), Symbolic Regression (SR), and Linear Matrix Regression (LM) as well as FaMoS [14]. Additionally, the ground truth (GT) is shown.



(b) Bouncing Ball Absolute Error in Height x for our Learning Approach with Linear Regression (LR) and Symbolic Regression (SR), and Linear Matrix Regression (LM) as well as FaMoS [14] (Note that the FaMos curve coincides with that of LM).

Figure 8 Bouncing Ball Height x and Error for our Learning Approach with Linear Regression (LR), Symbolic Regression (SR), and Linear Matrix Regression (LM) as well as FaMoS [14].

Table 3 Learning Results for the Bouncing Ball with FaMoS [14].

Flow Function	MSE	Learning Time in s
$\begin{cases} f_1(x,v) = \begin{pmatrix} 1.0 & 0.008 \\ 0.0 & 1.0 \end{pmatrix} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} + \begin{pmatrix} 0.0 \\ -0.006 \end{pmatrix} = \begin{pmatrix} x(t+1) \\ v(t+1) \end{pmatrix} \\ f_2(x,v) = \begin{pmatrix} 0.999 & 0.007 \\ -0.082 & 0.975 \end{pmatrix} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} + \begin{pmatrix} 0.0 \\ 0.036 \end{pmatrix} = \begin{pmatrix} x(t+1) \\ v(t+1) \end{pmatrix} \\ f_3(x,v) = \begin{pmatrix} 1.0 & 0.0 \\ -0.177 & 0.947 \end{pmatrix} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} + \begin{pmatrix} 0.0 \\ 0.052 \end{pmatrix} = \begin{pmatrix} x(t+1) \\ v(t+1) \end{pmatrix} \end{cases}$	$\begin{pmatrix} 4.021 \cdot 10^{-5} \\ 2.217 \cdot 10^{-4} \end{pmatrix}$	0.012

5.5 Identification of Linear Matrix Differential Equations

We perform our learning approach with linear matrix regression for the bouncing ball. The goal is to learn the two-dimensional state space of the ball as given in Equation (18). The results are shown in Table 2 and visualized in Figure 8a and Figure 8b.

The first line of the matrix differential equation is learned correctly except for the constant offset. The second line of the matrix differential equation differs from the original function. Instead of a constant reduction of the velocity, a factor smaller than one is learned for the previous velocity.

The MSE of the learned matrix differential equation for the height of the ball is higher compared to the results of one-dimensional linear regression and symbolic regression. The learning problem here is more complex as a two-dimensional state is learned. This regression method enables hybrid system identification for a higher-dimensional state space. The learning time for linear matrix regression is small.

5.6 Comparison to Traditional Learning of Hybrid Automata

Finally, we compare our learning approach with traditional learning of hybrid automata. We use the bouncing ball example and learn a hybrid automaton with the FaMoS tool [14]. The results are shown in Table 3.

The learning time of FaMoS is low and the MSE is small. FaMoS achieves a slightly lower MSE than all variants of our approach, which is due to the fact that FaMoS normalizes the data before learning and, thus, the MSE is evaluated on normalized data. The prediction of the state space with the learned flow function and the ground truth is shown in Figure 8a as a dotted green line and is very close to the ground truth. Figure 8b shows the denormalized absolute error of the learned flow function compared to the ground truth. The error is very similar to the error of our learning approach with linear matrix regression. However, instead of a single flow function, three flow functions are learned with FaMoS. All three flow functions are similar and are close to the ground truth, but not exactly equivalent to the original functions. This example shows that our approach can achieve a smaller number of flow functions and, thus, a smaller model. The traditional approach executed by FaMoS

detects transition points at the bouncing points of the ball, i.e., whenever $x \equiv 0$. For the given learning data, transition points are detected at the sampling steps 361 and 695. The flow functions are learned on the segments individually. The grouping step in FaMoS is not able to group the segments correctly, as they are not similar when comparing the curves of the segments. Thus, FaMoS fails to recognize identical dynamics in the segments. Our learning approach instead groups segments based on the flow functions and, thus, needs only a single flow function to represent the dynamics of the bouncing ball.

6 Discussion & Conclusion

In this work, we present a method to learn the flow functions of hybrid systems from observations integrating where the individual flow functions are valid. Revisiting the traditional sequential learning approach for hybrid systems, we propose a new iterative learning paradigm. This iterative process allows us to use the learned flow functions as intermediate results within the learning process. This ensures that transition points between dynamic modes are detected only if the current flow function is not able to explain the observations. Our learning approach, thus, is closer to the actual optimization problem of hybrid system identification and specifically focuses on the identification of hybrid systems with a small number of modes. We propose a template algorithm which is agnostic to the actual set of flow functions and learning methods used. Our empirical evaluation instantiates the template algorithm with symbolic regression, linear regression, and linear matrix identification. The results show that the approach identifies simple flow functions with high accuracy. Further, we demonstrate that the approach is able to identify hybrid systems with fewer modes compared to a traditional sequential learning approach on a bouncing ball benchmark.

Future work extends the current approach to involve the identification of events and transition conditions as well as reset relations, which enable the reset of system variables at the transition between modes.

References

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

2 Nathalie Barbosa Roa, Louise Travé-Massuyès, and Victor H. Grisales-Palacio. Dyclee: Dynamic clustering for tracking evolving environments. *Pattern Recognition*, 94:162–186, 2019. doi:10.1016/j.patcog.2019.05.024.

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

4 Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression.jl, 2023. doi:10.48550/arXiv.2305.01582.

5 Nemanja Hranisavljevic, Alexander Maier, and Oliver Niggemann. Discretization of hybrid CPPS data into timed automaton using restricted Boltzmann machines. *Engineering Applications of Artificial Intelligence*, 95:103826, 2020. doi:10.1016/j.engappai.2020.103826.

6 John R. Koza. *On the programming of computers by means of natural selection*. Koza, John R. Genetic programming. MIT Press, Cambridge, Mass. u.a., 1992.

7 Lennart Ljung. *System Identification: Theory for the User*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1986.

8 J. Lunze and F. Lamnabhi-Lagarrigue. *Handbook of Hybrid Systems Control: Theory, Tools, Applications*. Cambridge University Press, Cambridge, 2009.

9 Daniel L. Ly and Hod Lipson. Learning Symbolic Representations of Hybrid Dynamical Systems. *Journal of Machine Learning Research*, 13(115):3585–3618, 2012. doi:10.5555/2503308.2503356.

10 Meinard Müller. *Dynamic Time Warping*, chapter 4, pages 69–84. Springer-Verlag, Heidelberg, 2007.

11 Oliver Niggemann, Benno Stein, Asmir Vodencarevic, Alexander Maier, and Hans Kleine Büning. Learning behavior models for hybrid timed systems. *AAAI Conference on Artificial Intelligence*, 26(1):1083–1090, 2012. doi:10.1609/aaai.v26i1.8296.

12 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, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. doi:10.5555/1953048.2078195.

13 Swantje Plambeck. SymbolicRegression4HA. Software, BMBF project AGenC no. 16IS22047A, swhId: swh:1:dir:00ff8c08db289ab78af743dd7ce6b71dceccb37c (visited on 2025-10-21). URL: https://github.com/TUHH-IES/SymbolicRegression4HA, doi:10.4230/artifacts.24971.

14 Swantje Plambeck, Aaron Bracht, Nemanja Hranisavljevic, and Goerschwin Fey. Famos — fast model learning for hybrid cyber-physical systems using decision trees. In *International Conference on Hybrid Systems: Computation and Control (HSCC)*, 2024. doi:10.1145/3641513.3650131.

15 Swantje Plambeck, Maximilian Schmidt, Goerschwin Fey, Audine Subias, and Louise Travé-Massuyès. Dynamics-based identification of hybrid systems using symbolic regression. In *Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2024. doi:10.1109/SEAA64295.2024.00019.

16 Swantje Plambeck, Maximilian Schmidt, Audine Subias, Louise Travé-Massuyès, and Goerschwin Fey. Usability of Symbolic Regression for Hybrid System Identification - System Classes and Parameters. In *35th International Conference on Principles of Diagnosis and Resilient Systems (DX 2024)*, 2024. doi:10.4230/OASIcs.DX.2024.30.

17 Riccardo Poli, William B Langdon, and Nicholas F McPhee. *A Field Guide to Genetic Programming*, volume 10. Springer, 2008. URL: http://www.gp-field-guide.org.uk/.

18 Iman Saberi, Fathiyeh Faghih, and Farzad Sobhi Bavil. A passive online technique for learning hybrid automata from input/output traces. *ACM Transactions on Embedded Computing Systems*, 22(1):1–24, 2021. doi:10.1145/3556543.

19 Martin Tappler, Edi Muškardin, Bernhard K. Aichernig, and Bettina Könighofer. Learning environment models with continuous stochastic dynamics, 2023. doi:10.48550/arXiv.2306.17204.

20 Inc. The MathWorks. Model a bouncing ball in continuous time, 2025. URL: https://de.mathworks.com/help/stateflow/ug/modeling-a-bouncing-ball-in-continuous-time.html.

21 Xiaodong Yang, Omar Ali Beg, Matthew Kenigsberg, and Taylor T. Johnson. A framework for identification and validation of affine hybrid automata from input-output traces. *ACM Transactions on Cyber-Physical Systems*, 6(2):1–24, 2022. doi:10.1145/3470455.