Derivative-Agnostic Inference of Nonlinear Hybrid Systems

HENGZHI YU*, BOHAN MA*, and MINGSHUAI CHEN†, Zhejiang University, China
JIE AN, Institute of Software, Chinese Academy of Sciences, China
BIN GU, Beijing Institute of Control Engineering, China
NAIJUN ZHAN, Peking University, China
JIANWEI YIN, Zhejiang University, China

This paper addresses the problem of inferring a hybrid automaton from a set of input-output traces of a hybrid system exhibiting discrete mode switching between continuously evolving dynamics. Existing approaches mainly adopt a derivative-based method where (i) the occurrence of mode switching is determined by a drastic variation in derivatives and (ii) the clustering of trace segments relies on signal similarity – both subject to user-supplied thresholds. We present a derivative-agnostic approach, named DAINARX, to infer nonlinear hybrid systems where the dynamics are captured by nonlinear autoregressive exogenous (NARX) models. DAINARX employs NARX models as a unified, threshold-free representation through the detection of mode switching and trace-segment clustering. We show that DAINARX suffices to learn models that closely approximate a general class of hybrid systems featuring high-order nonlinear dynamics with exogenous inputs, nonlinear guard conditions, and linear resets. Experimental results on a collection of benchmarks indicate that our approach can effectively and efficiently infer nontrivial hybrid automata with high-order dynamics yielding significantly more accurate approximations than state-of-the-art techniques.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; • **Theory of computation → Timed and hybrid models**.

Additional Key Words and Phrases: hybrid system identification, model/automata learning, nonlinear dynamics

ACM Reference Format:
Hengzhi Yu, Bohan Ma, Mingshuai Chen, Jie An, Bin Gu, Naijun Zhan, and Jianwei Yin. 2025. Derivative-Agnostic Inference of Nonlinear Hybrid Systems. *ACM Trans. Embedd. Comput. Syst.* xx, x, Article xxx (October 2025), 24 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Hybrid systems are a commonly adopted mathematical model for cyber-physical systems (CPS). It describes the interaction between embedded computers and their continuously evolving physical environment or plant via sensors and actuators. Since *model-based design* [26, 29] emerges as a predominant approach to the construction of CPS – where a prerequisite is to have a model of the system under construction – identifying the hybrid system model underneath a potentially black-box system through external observations has become an essential problem in the design of CPS.

*Both authors contributed equally to this research.
†The corresponding author.

Authors' Contact Information: Hengzhi Yu; Bohan Ma; Mingshuai Chen, m.chen@zju.edu.cn, Zhejiang University, Hangzhou, China; Jie An, Institute of Software, Chinese Academy of Sciences, Beijing, China; Bin Gu, Beijing Institute of Control Engineering, Beijing, China; Naijun Zhan, Peking University, Beijing, China; Jianwei Yin, Zhejiang University, Hangzhou, China.

ACM 1558-3465/2025/10-ARTxxx
https://doi.org/XXXXXXX.XXXXXXX

Table 1. Qualitative comparison of different approaches to hybrid system identification.

Method	System Scope	Dynamics Type	High-Order	Resets	Inputs
Jin et al. [27]	Switched Systems	Polynomial ODEs	✗	✗	✗
Dayekh et al. [14]	Switched Systems	Polynomial ODEs	✗	✗	✗
POSEHAD [50]	Hybrid Automata	Polynomial ODEs	✗	✗	✓
LearnHA [23]	Hybrid Automata	Polynomial ODEs	✗	✓	✓
HySynth [54]	Hybrid Automata	Linear ODEs	✗	✗	✗
HAutLearn [66]	Hybrid Automata	Linear ODEs	✓	✗	✓
FaMoS [47]	Hybrid Automata	Linear ARX	✓	✗	✓
Madary et al. [37]	Switched Systems	Nonlinear ARX	✓	✗	✓
DAINARX (ours)	Hybrid Automata	Nonlinear ARX	✓	✓	✓

The general scope of this problem, known as *system identification* [35] or *model learning* [57], has undergone a surge of interest due to prominent applications in, e.g., system analysis [8, 17, 51, 65], model checking [18, 46, 53], falsification [7, 62], and monitoring [21, 28].

Hybrid automata (HAs) [2, 38] are a widely used formalism to characterize the discrete-continuous behaviors in hybrid systems. It extends finite-state machines by associating each discrete location (mode) with a vector of continuously evolving dynamics – typically modeled by a system of ordinary differential equations (ODEs) – subject to potential discrete mode switching upon transitions. Every discrete transition is equipped with a guard to constrain the switching condition and a reset function to update the system state after the switching. See a concrete example HA in Section 2.

This paper addresses the problem of *inferring a hybrid automaton $\overline{\mathcal{H}}$ from a set of observations $\mathcal{D}$ (i.e., discrete-time traces) of a hybrid system $\mathcal{H}$*. In particular, $\overline{\mathcal{H}}$ is expected to closely approximate the behaviors of $\mathcal{H}$ – not only on $\mathcal{D}$, but also on unseen traces. To tackle this problem, extensive results have been established in the literature, yet are confined to different subclasses of hybrid systems due to its inherent complexity; see Table 1 for a qualitative comparison of different approaches. A more thorough review of related work is deferred to Section 7.

We present an inference approach called DAINARX to identify nonlinear hybrid automata where the dynamics are captured by *nonlinear autoregressive exogenous (NARX) models* – a commonly used formalism in hybrid system identification [10, 34]. DAINARX follows the conventional pipeline of trace segmentation – segment clustering – mode characterization – guard and reset learning. However, in contrast to existing approaches that rely heavily on derivative variation and trace similarity for the first two steps, DAINARX works in a *derivative-agnostic* manner by employing NARX models as a unified, threshold-free representation through multiple steps, and thereby unleashes the pipeline for its general applicability to inferring hybrid system models featuring *high-order nonlinear dynamics with exogenous inputs, nonlinear guard conditions, and linear resets* (cf. Table 1). Experimental results on a collection of benchmarks demonstrate DAINARX's distinct superiority in accuracy, applicability, and efficiency compared to state-of-the-art inference tools.

Contributions. The main contributions of this paper are summarized as follows.

- We present the DAINARX framework leveraging NARX model fitting for inferring nonlinear hybrid systems. To the best of our knowledge, DAINARX is the first approach that admits high-order non-polynomial dynamics with non-polynomial inputs/guards, and linear resets.
- We show that, unlike threshold-based approaches, DAINARX provides formal guarantees on the correctness of both mode-switching detection and trace-segment clustering.
- We have implemented DAINARX and conducted an extensive set of experiments to demonstrate its effectiveness and efficiency in comparison to state-of-the-art inference tools.

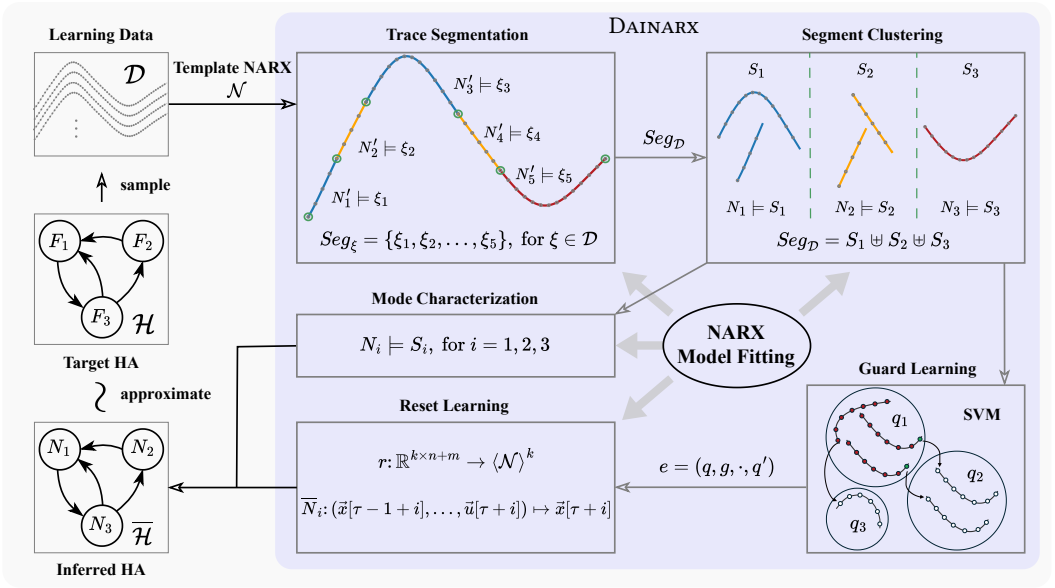


Fig. 1. The general workflow of DAINARX.

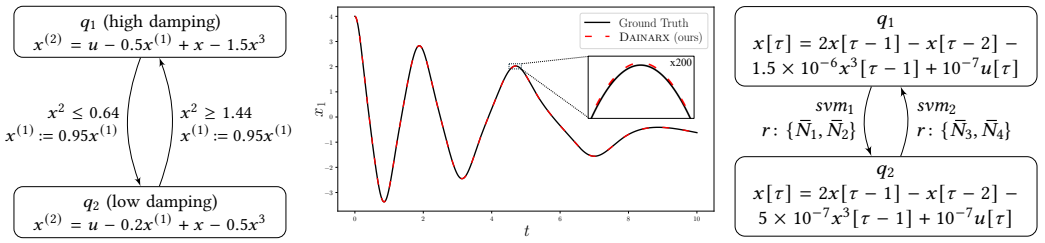
2 A Bird's-Eye Perspective

This section outlines the general workflow of our DAINARX framework as depicted in Fig. 1. Given a possibly high-order nonlinear black-box hybrid system $\mathcal{H}$, DAINARX aims to infer – from sampled discrete-time input-output traces $\mathcal{D}$ of $\mathcal{H}$ – a hybrid automaton $\overline{\mathcal{H}}$ with NARX-modeled dynamics (subject to a template $\mathcal{N}$) that closely approximates $\mathcal{H}$. The inference process in DAINARX adopts a common pipeline of the following steps: (i) *trace segmentation* for partitioning every trace $\xi \in \mathcal{D}$ into a set Seg_ξ of segments pertaining to different system modes; (ii) *segment clustering* for grouping all trace segments that are deemed to follow the same mode dynamics into a cluster S_i; (iii) *mode characterization* for learning the mode dynamics characterized by a NARX model N_i under template $\mathcal{N}$ for each cluster; (iv) *guard learning* and *reset learning* for inferring the guard and reset conditions along discrete transitions, respectively. The main technical novelty of DAINARX is that it employs *NARX model fitting* as an efficient engine throughout these steps and yields threshold-free inference of a high-order nonlinear hybrid automaton $\overline{\mathcal{H}}$ that closely approximates the target system $\mathcal{H}$.

We use the following example to demonstrate the application of DAINARX.

Example 1 (Duffing Oscillator [31]). The Duffing oscillator is a second-order nonlinear ODE commonly used to describe, e.g., magnetoelastic systems and nonlinear circuits. Consider the hybrid automaton depicted in Fig. 2a, which models a mass undergoing a nonlinear, forced vibration and constantly switches between two controlled modes with different damping coefficients. Each guarded transition between modes is accompanied by a loss of speed (via resets). The input signal u is a cosine function of time t modeling an external driving force. To the best of our knowledge, *none of the existing techniques suffices* to identify a relatively accurate and useful model of this system due to its complex nature of high-order nonlinear dynamics, non-polynomial inputs, and resets.

Suppose we have collected a dataset $\mathcal{D}$ of 10 discrete-time traces (9 for training, 1 for testing; each with 10,000 data points) via external observations of $\mathcal{H}$. Then, given a template NARX model with nonlinear terms $x^3[\tau - 1]$ and $x^3[\tau - 2]$ (see details in Section 3.2), DAINARX first segments



(a) Target system $\mathcal{H}$. **(b)** Example trace of $\mathcal{H}$ and $\overline{\mathcal{H}}$. **(c)** Inferred automaton $\overline{\mathcal{H}}$.

Fig. 2. Inference of the Duffing oscillator via DAINARX.

the training traces into 155 segments by detecting mode-switching points via NARX model fitting: A model switching is detected if and only if the the current segment under consideration *cannot* be fitted by a single NARX model; Then, all segments that are fittable by the same NARX model are grouped into the same cluster, yielding 2 clusters for this example. The flow dynamics of these two modes are then inferred, again, by NARX model fitting. Finally, we extract the complete hybrid automaton by learning guard conditions using support vector machines (SVMs) with kernel tricks [12] as well as reset functions expressed by NARX models. DAINARX ultimately infers a hybrid automaton $\overline{\mathcal{H}}$ (Fig. 2c) with NARX-modeled dynamics that closely approximates the behavior of $\mathcal{H}$: As depicted in Fig. 2b, the trace generated by $\overline{\mathcal{H}}$ exhibits a maximum (average, resp.) deviation of 0.0003 (2.8×10^{-5}, resp.) from the testing trace of the original system $\mathcal{H}$. See Section 5 for more details on this example (the `duffing` benchmark). ◁

3 Problem Formulation

This section first recaps the basics of hybrid systems characterized by hybrid automata and NARX models and then formulates our problem of identifying hybrid systems from time-discrete traces.

Notations. We denote by $\mathbb{N}$, $\mathbb{N}_{>0}$, $\mathbb{R}$, and $\mathbb{R}_{>0}$ the set of natural, positive natural, real, and positive real numbers, respectively; let $\mathbb{R}^{\infty}_{>0} \triangleq \mathbb{R}_{>0} \cup \{\infty\}$. $X \uplus Y$ denotes the disjoint union of sets X and Y. Given matrix A, we denote by A_{ij}, $A_{i,:}$, and $A_{:,j}$ respectively the (i, j)-th element, the i-th row vector, and the j-th column vector of A. For $n \in \mathbb{N}_{>0}$ and $T \in \mathbb{R}^{\infty}_{>0}$, an n-dimensional (continuous) *signal* $\sigma\colon [0, T) \to \mathbb{R}^n$ is a function that maps each timepoint in the domain $[0, T)$ to an n-dimensional real-valued vector in $\mathbb{R}^n$; a signal is discrete if its time domain consists of discrete timestamps.

3.1 Hybrid Automata

Hybrid automata [2, 38] are a widely adopted formalism to characterize the discrete-continuous behaviors of hybrid systems. It extends finite-state machines by associating each discrete location with a vector of continuously evolving dynamics subject to discrete jumps upon transitions.

Definition 2 (Hybrid Automaton [2, 38]). *An n-dimensional* hybrid automaton *is a sextuple*

$$\mathcal{H} \triangleq (Q, V, \mathcal{F}, Inv, Init, \mathcal{T}) \quad \textit{where}$$

- $Q = \{q_1, q_2, \ldots, q_j\}$ *is a finite set of* discrete modes *representing the discrete states of* $\mathcal{H}$.
- $V = X \uplus U$ *is a finite set of* continuous variables *consisting of (observable) output variables* $X = \{x_1, x_2, \ldots, x_n\}$ *and (controllable) input variables* $U = \{u_1, u_2, \ldots, u_m\}$. *A real-valued vector* $\vec{v} = (x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_m) \in \mathbb{R}^{n+m}$ *denotes a continuous state. The overall* state space *of* $\mathcal{H}$ *is* $Q \times \mathbb{R}^{n+m}$, *i.e., the Cartesian product of the discrete and continuous state space. We denote by* $\vec{x} = \vec{v}|_X$ *and* $\vec{u} = \vec{v}|_U$ *the projection of* $\vec{v}$ *onto the output and input space, respectively.*

- $\mathcal{F} = \{F_q\}_{q\in Q}$ *assigns to each mode* $q \in Q$ *a* flow function F_q *characterizing the change of output variables* X *over inputs* U*; Detailed forms of* F_q *are specified below.*
- $Inv \subseteq Q \times \mathbb{R}^{n+m}$ *specifies the* mode invariants *representing admissible states of* $\mathcal{H}$*; We denote by* $Inv(q) = \{\vec{v} \mid (q, \vec{v}) \in Inv\}$ *the invariant of mode* $q \in Q$*.*
- $Init \subseteq Inv$ *is the* initial condition *defining admissible initial states of* $\mathcal{H}$*.*
- $\mathcal{T} \subseteq Q \times G \times R \times Q$ *is the set of* transition relations *between modes, where* G *is a set of* guards $g \subseteq \mathbb{R}^{n+m}$ *and* R *is a set of* resets *(aka,* updates*)* $r\colon \mathbb{R}^{n+m} \to \mathbb{R}^n$*. A transition* $e = (q, g, r, q') \in \mathcal{T}$ *is triggered immediately when* g *is active*[1]*, i.e.,* $\vec{v} \in g$ *(à la the synchronous semantics in* [41]*); The output state vector* $\vec{x}$ *is updated to* $r(\vec{v})$ *upon taking this transition.*

The flow dynamics of a hybrid automaton $\mathcal{H}$ within mode q is typically modeled by a system of (potentially high-order) *ordinary differential equations* (ODEs) of the form

$$\vec{x}^{(k)} \;=\; F_q\left(\vec{x}, \vec{x}^{(1)}, \ldots, \vec{x}^{(k-1)}, \vec{u}\right) , \tag{1}$$

where $k \in \mathbb{N}$ is the *order* of the ODE and $\vec{x}^{(i)} = \mathrm{d}^{(i)}\vec{x}/\,\mathrm{d}t^i$ is the i-th order derivative of $\vec{x}$ w.r.t. time t ($0 \le i \le k$). Note that for $k > 1$, the initial condition *Init* is extended with initial valuations for the derivatives $\vec{x}^{(1)}, \ldots, \vec{x}^{(k-1)}$; these derivatives can also be reset in analogous to $\vec{x}$.

The semantics of a hybrid automaton $\mathcal{H}$ is usually captured by the set of continuous-time hybrid traces. Such a trace starts from an initial state in *Init* and evolves according to the flow dynamics $\mathcal{F}$ (while remaining within invariants *Inv*) as well as the discrete jumps per $\mathcal{T}$:

Definition 3 (Continuous-Time (Hybrid) Trace [23]). *A* continuous-time hybrid trace *(aka,* run*) of a hybrid automaton* $\mathcal{H}$ *with first-order*[2] *dynamics is a (possibly infinite) sequence*

$$\rho \;=\; \left\langle (q_0, \vec{v}_0) \xrightarrow{\tau_0} (q_0, \vec{v}_0') \xrightarrow{e_0} (q_1, \vec{v}_1) \xrightarrow{\tau_1} (q_1, \vec{v}_1') \xrightarrow{e_1} \cdots \right\rangle$$

satisfying the following conditions: (i) initialization: $(q_0, \vec{v}_0) \in Init$*; (ii)* continuous evolution: *for any* $i \in \mathbb{N}$*, there is signal* $\sigma_i\colon [0, \tau_i] \to \mathbb{R}^{n+m}$ *such that* $\sigma_i(0) = \vec{v}_i$*,* $\sigma_i(\tau_i) = \vec{v}_i'$*. For any* $t \in [0, \tau_i]$*,* $\sigma_i(t) \in Inv(q_i)$ *and* $\sigma_i(t)$ *satisfies flow dynamics* (1) *(with* $k = 1$*)*[3]*; (iii)* discrete consecution: *for any* $e_i = (q_i, g_i, r_i, q_{i+1})$*,* $\vec{v}_i' \in g_i$ *and* $\vec{v}_{i+1} = (r_i(\vec{v}_i'), \vec{v}_i'|_U)$*. A* continuous-time trace *over run* ρ *is a signal* $\sigma\colon [0, T) \to \mathbb{R}^{n+m}$ *such that,* $\forall i \in \mathbb{N}\colon \sigma(t) = \sigma_i(t - T_i)$ *for* $t \in [T_i, T_{i+1})$ *with* $T_i = \sum_{j=0}^{i-1} \tau_j$ *and* $T_0 = 0$*.*

To alleviate *blocking behaviors* of hybrid automata [6, 36], we assume the most permissive form of invariants $Inv = \{(q, \mathbb{R}^{n+m})\}_{q\in Q}$ along the line of [23, 47]. Moreover, we consider *nonlinear* hybrid automata in the sense that both the *flow dynamics* $\mathcal{F}$ and the *guard conditions* G can potentially be described by nonlinear functions even beyond polynomials.[4] The duffing oscillation system given in Example 1 is a concrete hybrid automaton with second-order ODEs and inputs.

This paper aims to identify hybrid system models in a *passive learning* manner (see, e.g., [23, 34, 37, 47, 66]), i.e., to infer a hybrid automaton $\mathcal{H}$ based on external observations of the system behavior without actively interacting with the system during learning. Such observations are typically given by a set of *discrete-time traces*, i.e., sequences of timestamped vectors:

Definition 4 (Discrete-Time Trace). *A* discrete-time (input-output) trace *of a hybrid automaton* $\mathcal{H}$ *with first-order dynamics is a (possibly infinite) sequence*

$$\xi \;=\; \left\langle (t_0, \vec{v}_0), (t_1, \vec{v}_1), \ldots \right\rangle , \quad \textit{with } 0 \le t_0 < t_1 < \cdots$$

[1] We assume all transitions are *deterministic*, i.e., at most one guard (for a source mode) is active at any time.
[2] We omit the complex yet straightforward extension to the case of high-order dynamics with $k > 1$.
[3] This means that $\mathrm{d}\vec{x}_i/\,\mathrm{d}t = F_{q_i}(\vec{x}_i, \vec{u}_i)$ where $\vec{x}_i$ and $\vec{u}_i$ are the projections of signal σ_i onto X and U, respectively.
[4] We allow *linear resets* only as we use transition-matrix representations for learning resets.

such that there exists a continuous-time trace σ of $\mathcal{H}$ satisfying $\vec{v}_i = \sigma(t_i) \in \mathbb{R}^{n+m}$ for all $i \in \mathbb{N}$. We call such discrete-time trace ξ a discretization *of its continuous-time counterpart σ.*

In practice, given a hybrid system $\mathcal{H}$, one can obtain a set of *finite* discrete-time traces $\mathcal{D} = \{\xi_j\}_{1\le j\le M}$ – called *learning data* – by *sampling* the observable state $\vec{x}$ under control input $\vec{u}$ of $\mathcal{H}$. By assuming an identical, fixed sampling period Δt for all variables in V, we obtain the learning data $\{\xi_j\}_{1\le j\le M}$ where each finite discrete-time trace ξ_j (of length $\ell+1$) is of the form

$$\xi \;=\; \left\langle (t_0, \vec{v}_0), (t_0+\Delta t, \vec{v}_1), \ldots, (t_0+\ell\Delta t, \vec{v}_\ell) \right\rangle \;. \tag{2}$$

We remark that sampling via external measurements or numerical simulations inevitably introduces *errors*. However, as a convention in system identification, we aim to learn a model that best fits the *sampled data* (while exhibiting predictive power along unseen traces) instead of exact traces of the original system $\mathcal{H}$ which are often intractable to obtain, especially for nonlinear dynamics [19].

To this end, we infer hybrid system models where the flow dynamics $\mathcal{F}$ is represented using *nonlinear autoregressive exogenous* (NARX) models – a commonly used formalism in the identification of (nonlinear) hybrid systems [10, 15, 33, 34, 37, 47] – for fitting discrete-time traces.

3.2 NARX Models

We forge the flow dynamics $\mathcal{F} = \{F_q\}_{q\in Q}$ using (a system of) NARX models: The dynamics of each mode is represented by a *nonlinear* ARX model in the form of a k-th order *difference equation*:[5]

$$\vec{x}[\tau] \;=\; F_q\left(\vec{x}[\tau-1], \vec{x}[\tau-2], \ldots, \vec{x}[\tau-k], \vec{u}[\tau]\right) \;, \quad \text{for } \tau \ge k \;, \tag{3}$$

where the current output vector $\vec{x}[\tau] \in \mathbb{R}^n$ is determined by its k recent history states as well as the current input vector $\vec{u}[\tau] \in \mathbb{R}^m$ through a nonlinear function $F_q \colon \mathbb{R}^{k\times n} \times \mathbb{R}^m \to \mathbb{R}^n$.

We observe that, in many practical applications, the form of differential dynamics as well as their difference counterparts can be expressed as a linear combination of simple nonlinear terms and linear terms. This gives rise to an equivalent form of the NARX model (3) as

$$\vec{x}[\tau] \;=\; \sum\nolimits_{i=1}^{\alpha} \vec{a}_i \circ f_i\left(\vec{x}[\tau-1], \ldots, \vec{x}[\tau-k], \vec{u}[\tau]\right) \;+\; \sum\nolimits_{i=1}^{k} B_i \cdot \vec{x}[\tau-i] \;+\; B_{k+1} \cdot \vec{u}[\tau] \;+\; \vec{c} \;, \tag{4}$$

where $f_1, f_2, \ldots, f_\alpha$ are arbitrary vector-valued nonlinear components in $\mathbb{R}^n$ representing the model structure of F_q; $\vec{a}_i, \vec{c} \in \mathbb{R}^n$ are vector coefficients and $B_i \in \mathbb{R}^{n\times n}$, $B_{k+1} \in \mathbb{R}^{n\times m}$ are matrix coefficients; $\cdot$ denotes matrix multiplication and $\circ$ denotes the Hadamard product, i.e., $(A_1 \circ A_2)_{ij} = (A_1)_{ij}(A_2)_{ij}$. For a more compact representation, we maintain a *coefficient matrix* $\mathcal{A} \in \mathbb{R}^{n\times(\alpha+kn+m+1)}$ of (4):

$$\mathcal{A} \;=\; \begin{bmatrix} \vec{a}_1 & \vec{a}_2 & \ldots & \vec{a}_\alpha & B_1 & B_2 & \ldots & B_k & B_{k+1} & \vec{c} \end{bmatrix} \;,$$

and thereby denote by $\mathcal{N}_q = (\mathcal{A}, \{f_i\}_{1\le i\le\alpha})$ the flow dynamics of a hybrid system within mode q.

A *template NARX model* refers to flow dynamics $\mathcal{N}$ whose nonlinear components $\{f_i\}_{1\le i\le\alpha}$ are given yet the coefficient vector $\mathcal{A}$ consists of *unknown parameters*. Let Λ be an instantiation of $\mathcal{A}$. We denote by $N = \mathcal{N}[\Lambda]$ the *instance* of $\mathcal{N}$ under Λ and by $\langle\mathcal{N}\rangle$ the set of all instances of $\mathcal{N}$.

A *descrete-time trace* of a hybrid automaton $\mathcal{H}$ with NARX-modeled dynamics can be defined as a discrete signal $\zeta \colon \mathbb{N} \to \mathbb{R}^{n+m}$ in analogy to (the definition of continuous-time trace in) Definition 3, except that we replace the ODE-modeled dynamics (1) with the NARX dynamics (4).

The identification problem concerned in this paper reads as follows:

[5]For the sake of simplicity, we consider *noise-free* NARX models; see Section 6 for a detailed discussion.

Problem Statement. Given a set of finite discrete-time traces $\{\xi_j\}_{1\le j\le M}$ as per (2) sampled from a hybrid automaton $\mathcal{H}$ and a template NARX model $\mathcal{N} = (\mathcal{A}, \{f_i\}_{1\le i\le \alpha})$, identify a hybrid automaton $\overline{\mathcal{H}}$ where all mode dynamics are instances of $\mathcal{N}$ such that, for any trace ξ_j, there exists a finite discrete-time trace ζ of $\overline{\mathcal{H}}$ satisfying $\zeta(\tau) = \vec{v}_\tau$ for all $0 \le \tau \le \ell$.

The above problem statement specifies the ideal setting where the inferred model $\overline{\mathcal{H}}$ *coincides* with the target model $\mathcal{H}$ over the given learning data. In practice, however, one often aims to learn a model $\overline{\mathcal{H}}$ that *approximates* $\mathcal{H}$ in the sense that $\overline{\mathcal{H}}$ is capable of generating good predictions along both given traces (for training) and unseen traces (for testing) of $\mathcal{H}$.

4 Derivative-Agnostic Inference

This section presents our derivative-agnostic approach DAINARX for inferring hybrid systems with NARX-modeled dynamics from discrete-time input-output traces. As has been exemplified in Section 2, DAINARX follows the common pipeline of hybrid system identification including (I) *Trace segmentation*: It partitions every discrete-time trace in the learning data into trace segments by the detection of changepoints, namely, data points on the trace that signify a change of dynamics (i.e., mode switching) between neighboring segments; (II) *Segment clustering*: It groups the trace segments into several clusters such that each cluster includes all segments that are deemed to be generated by the same mode dynamics; (III) *Mode characterization*: It learns a NARX model adhering to the given template to characterize the mode dynamics for each cluster; (IV) *Guard learning*: It learns a guard condition for each mode switching using SVMs; (V) *Reset learning*: It learns a reset function for each mode switching again using NARX models.

The main technical novelty of DAINARX lies in Steps (I), (II) and (V), and, in particular, we employ *NARX model fitting* as the basis throughout Steps (I), (II), (III), and (V), which aims to fit one or more trace segments using a NARX model. In the rest of this section, we first present NARX model fitting, then elaborate each step of DAINARX, and finally provide an algorithmic complexity analysis.

4.1 NARX Model Fitting

Given a template NARX model $\mathcal{N} = (\mathcal{A}, \{f_i\}_{1\le i\le \alpha})$ and a set $S = \{\xi_j\}_{1\le j\le M}$ of finite discrete-time traces or trace segments (cf. Section 4.2) where each trace (segment) takes the form $\xi = \langle (t_0, \vec{v}_0), (t_0 + \Delta t, \vec{v}_1), \ldots, (t_0 + \ell\Delta t, \vec{v}_\ell)\rangle$, the task of *NARX model fitting* asks to find an instance $N = \mathcal{N}[\Lambda] \in \langle\mathcal{N}\rangle$, if it exists, such that N *fits* S:

Definition 5 (Trace Fitting). *Let ξ be a finite discrete-time trace of hybrid automaton $\mathcal{H}$. We say that a k-th order NARX model N* fits *ξ, denoted by $N \models \xi$, if (4) holds by substituting $\vec{v}_\tau|_X$ and $\vec{v}_\tau|_U$ in ξ respectively for $\vec{x}[\tau]$ and $\vec{u}[\tau]$. N fits a set $S = \{\xi_j\}$ of such trace (segments) if $N \models \xi_j$ for all ξ_j in S. We say S is* fittable *by a template NARX model $\mathcal{N}$ if there exists $N \in \langle\mathcal{N}\rangle$ such that $N \models S$.*

We start with the simple case of fitting a *single* trace (segment) ξ by finding $\Lambda \in \mathcal{A}$ such that $N = \mathcal{N}[\Lambda] \models \xi$. We address this task by the *linear least squares method* (LLSQ) as established in the field of regression analysis [22]: We find the best-fitting NARX coefficients Λ to ξ by minimizing the sum of the squares of the "offsets" of data points on ξ from $\mathcal{N}[\Lambda]$, i.e.,

$$\underset{\Lambda_{i,:}}{\text{minimize}} \quad \left\| O_{i,:} - \Lambda_{i,:} \cdot D_i \right\|_2 \quad \text{for } i = 1, 2, \ldots, n \tag{5}$$

where $\|\cdot\|_2$ is the L^2-norm; $O \in \mathbb{R}^{n\times(\ell-k+1)}$ and $D_i \in \mathbb{R}^{(\alpha+kn+m+1)\times(\ell-k+1)}$ are respectively the *observed value matrix* and the *data matrix* (with respect to the i-th row of Λ), which are constructed

as follows. Let $\vec{x}_\tau = \vec{v}_\tau|_X$ and $\vec{u}_\tau = \vec{v}_\tau|_U$ for $0 \le \tau \le \ell$, then the observed value matrix O is

$$O \;=\; \begin{bmatrix} \vec{x}_\ell & \vec{x}_{\ell-1} & \cdots & \vec{x}_k \end{bmatrix} \;;$$

and the data matrix D_i is constructed as

$$D_i \;=\; \begin{bmatrix} \chi_{i,\ell} & \chi_{i,\ell-1} & \cdots & \chi_{i,k} \end{bmatrix} \quad \text{where}$$

$$\chi_{i,\tau} \;=\; \begin{bmatrix} f_1\left(\vec{y}_\tau\right)_{i,:} & \cdots & f_\alpha\left(\vec{y}_\tau\right)_{i,:} & \vec{y}_\tau^{\mathsf{T}} & 1 \end{bmatrix}^{\mathsf{T}} \quad \text{with} \quad \vec{y}_\tau = (\vec{x}_{\tau-1}, \cdots, \vec{x}_{\tau-k}, \vec{u}_\tau) \;.$$

The linear optimization problem (5) can be discharged using any off-the-shelf solvers by finding its least-squares solution [22]. The following example illustrates the optimization problem:

Example 6 (Trace Fitting via LLSQ). To fit the trace $\xi = \left\langle (t_0, \vec{v}_0), (t_0 + \Delta t, \vec{v}_1), \ldots, (t_0 + 4\Delta t, \vec{v}_4) \right\rangle$ under the second-order *univariate* template NARX model $\mathcal{N} = (\mathcal{A}, \{x^2[\tau-1], x^3[\tau-2]\})$ (where all coefficients in $\mathcal{A}$ are of dimension 1×1), the matrices can be constructed as $O = [\, x_4,\ x_3,\ x_2\,]$ and $D_1 = \left[\, x_3^2,\ x_2^2,\ x_3,\ x_2,\ 1;\ x_2^2,\ x_1^3,\ x_2,\ x_1,\ 1;\ x_1^2,\ x_0^3,\ x_1,\ x_0,\ 1\,\right]^{\mathsf{T}}$. Then, the optimization objective in (5), for the first column, is $x_4 - (a_1 \cdot x_3^2 + a_2 \cdot x_2^3 + B_1 \cdot x_3 + B_2 \cdot x_2 + c \cdot 1)$ (cf. (4)). ◁

The above LLSQ-based fitting can be generalized to the case of multiple trace (segments) with $S = \{\xi_j\}_{1 \le j \le M}$ by extending the matrices O and D_i as

$$O \;=\; \begin{bmatrix} O^{\xi_1} & O^{\xi_2} & \cdots & O^{\xi_M} \end{bmatrix} \quad \text{and} \quad D_i \;=\; \begin{bmatrix} D_i^{\xi_1} & D_i^{\xi_2} & \cdots & D_i^{\xi_M} \end{bmatrix} , \tag{6}$$

where O^{ξ_j} and $D_i^{\xi_j}$ are the corresponding matrices constructed for trace ξ_j.

It follows that the NARX model $\mathcal{N}[\Lambda]$ obtained by discharging the optimization problem (5) with (6) indeed fits the set of trace (segments) S if the optimal value is zero for all $1 \le i \le n$:

Theorem 7 (Correctness of Trace Fitting). *Given a template NARX model $\mathcal{N}$ and a set of finite discrete-time trace (segments) $S = \{\xi_j\}_{1 \le j \le M}$. Let $E_S = \max\{E_1, E_2, \ldots, E_n\}$ where each E_i is the optimal value of (5) with (6) w.r.t. i; let $N = \mathcal{N}[\Lambda] \in \langle \mathcal{N} \rangle$ be the NARX model corresponding to the optimum Λ. Then, S is fittable by $\mathcal{N}$ if and only if $E_S = 0$. In this case, we have $N \models S$.*

4.2 Trace Segmentation

Given the learning data $\mathcal{D} = \{\xi_j\}_{1 \le j \le M}$, the initial step of DAINARX is to partition every discrete-time trace in $\mathcal{D}$ into trace segments such that the neighboring segments are generated by different mode dynamics. Intuitively, a *trace segment* is a finite subsequence of a discrete-time trace:

Definition 8 (Trace Segment). *Given a finite discrete-time trace $\xi = \langle (t_0, \vec{v}_0), (t_0 + \Delta t, \vec{v}_1), \ldots, (t_0 + \ell\Delta t, \vec{v}_\ell) \rangle$ and $l, h \in \mathbb{N}$ satisfying $l < h \le \ell + 1$. The segment of ξ over $[l, h)$ is $\xi_{l,h} \triangleq \langle (t_0', \vec{v}_0'), (t_0' + \Delta t, \vec{v}_1'), \ldots, (t_0' + (h-l-1)\Delta t, \vec{v}_{h-l-1}') \rangle$, where $t_0' = t_0 + l\Delta t$ and for any $\tau < h - l$, $\vec{v}_\tau' = \vec{v}_{\tau+l}$.*

By definition, a trace segment is also a (finite) discrete-time trace. Since the segmentation procedure is identical for every trace in the learning data $\mathcal{D}$, we consider an individual trace $\xi \in \mathcal{D}$. Recall that the semantics of hybrid systems considered in this paper is deterministic and *synchronous* [41]: A transition of a hybrid automaton is triggered *immediately* upon the activation of its guard and thus will be reflected by the corresponding mode switching in the observed discrete-time traces. Existing derivative-based segmentation approaches – e.g., FaMoS [47] which uses linear ARX models for trace fitting – determine mode switching (and thus a segmentation point) based on the presence of large deviations between immediate future and immediate past measured by sliding windows [56] around the point. In DAINARX, we also employ a sliding window along the trace, however, the mechanism for detecting mode switching is essentially different: We identify a set of *timestamps* featuring the following *changepoint property*:

Property 9 (Changepoints). *Given a discrete-time trace* $\xi = \langle (t_0, \vec{v}_0), (t_0 + \Delta t, \vec{v}_1), \ldots, (t_0 + \ell \Delta t, \vec{v}_\ell) \rangle$ *and a template NARX model* $\mathcal{N}$*. Let* $CP = \{p_0, p_1, \ldots, p_s\} \in \mathbb{N}^s$ *be a set of timestamps with* $0 = p_0 < p_1 < p_2 < \cdots < p_s = \ell + 1$*. We say CP is the (unique) set of* changepoints *w.r.t.* ξ *and* $\mathcal{N}$ *if it satisfies*

(i) $\xi_{p_i, p_{i+1}}$ *is* fittable *by* $\mathcal{N}$ *for any* $0 \leq i < s$*; and*
(ii) $\xi_{p_i, (p_{i+1})+1}$ *is* not fittable *by* $\mathcal{N}$ *for any* $0 \leq i < s - 1$.

Based on the identified set *CP* of changepoints, we can obtain the set of segments of trace ξ as

$$Seg_\xi = \left\{ \xi_{p_0, p_1}, \xi_{p_1, p_2}, \ldots, \xi_{p_{s-1}, p_s} \right\} .$$

For the entire learning data $\mathcal{D} = \{\xi_j\}_{1 \leq j \leq M}$, the set of all trace segments is $Seg_\mathcal{D} = \bigcup_{j=1}^{M} Seg_{\xi_j}$. The intuition behind segmentation in DAINARX as well as its difference from derivative-based segmentation is depicted in Fig. 3. We note that, in FaMoS, the presence of "large deviations" depends on a *user-supplied threshold* γ, which is difficult to provide in practice since genuine mode-switching points may be missed (e.g., $\xi(12)$ in Fig. 3) if γ is too large while spurious mode-switching points may be identified (e.g., $\xi(3), \xi(5), \xi(7)$ in Fig. 3) if γ is too small. A concrete scenario is demonstrated in Section 5.2.3.

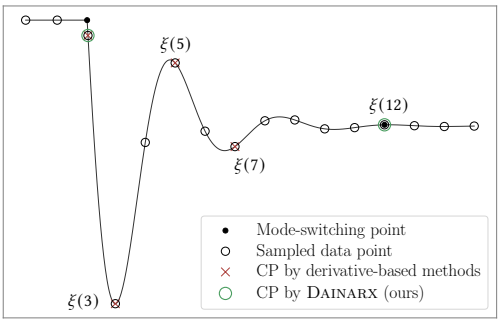


Fig. 3. Illustration of segmentation: DAINARX vs. derivative-based strategies.

In order to identify the set of changepoints as per Property 9, we adopt a *sliding window* (of size $w > k + 1$) along every trace $\xi \in \mathcal{D}$; the details are given in Algorithm 1. We remark that, in Lines 9 to 11 of Algorithm 1, we perform a *posterior screening* to ensure that every trace segment in the eventually obtained set Seg_ξ is fittable by the template NARX model $\mathcal{N}$. Such a screening process is necessary due to the following three *corner cases*: (i) the given template NARX model $\mathcal{N}$ does not suffice to fit some trace segment(s) (within a mode); (ii) the distance between some neighboring mode-switching points falls below the size w of the sliding window; (iii) two segments ξ and ξ' with $w - 1$ overlapped data points are both fittable by $\mathcal{N}$, yet $\{\xi, \xi'\}$ is *not* fittable by $\mathcal{N}$ – in this case, we may miss the changepoint in ξ'. In the absence of these corner cases (see details in the proof of Theorem 10, Appendix A), we can establish the correctness of Algorithm 1 as follows:

Theorem 10 (Correctness of Segmentation in Algorithm 1). *Given a discrete-time trace* ξ *and a template NARX model* $\mathcal{N}$*. The set* Seg_ξ *of trace segments returned by Algorithm 1 – in the absence of the above three corner cases – satisfies the changepoint property as stated in Property 9.*

Remark. One can replace the sliding window-based search of changepoints in Algorithm 1 with a *binary search*. The so-obtained algorithm (see Appendix B) exhibits a (logarithmic) increase in time complexity, yet yields stronger theoretical guarantees (cf. Theorem 16).

4.3 Segment Clustering

The goal of segment clustering is to group the trace segments obtained through segmentation into several clusters such that each cluster includes all segments that are deemed to be generated by the same mode dynamics. The key to this procedure is the criterion for merging a set of trace segments into the same cluster. Existing approaches, such as POSEHAD [50], FaMoS [47], and LearnHA [23], mainly adopt the criterion based on *trace similarity* computed by, e.g., the dynamic time warping (DTW) algorithm [9], which again depend on *user-supplied thresholds* to determine the clustering of segments, and may induce wrong clustering because "similar traces" do not imply the "same

Algorithm 1: Trace Segmentation Based on NARX Model Fitting (via sliding window of size w)

Input: A discrete-time trace ξ of length $\ell + 1$ and a template NARX model $\mathcal{N}$.
Output: A set Seg_{ξ} of trace segments of ξ under $\mathcal{N}$.

$CP \leftarrow \{0\}; l \leftarrow 0$; ▷ initialize CP and l (left bound of sliding window)
while $l \leq \ell + 1 - w$ **do**
  if $\xi_{l,l+w}$ *is not fittable by* $\mathcal{N}$ **then** ▷ detected potential changepoint at $l + w - 1$
    $CP \leftarrow CP \cup \{l + w - 1\}$; ▷ update the changepoints
    $l \leftarrow l + w$; ▷ skip data points in the current window
  else
    $l \leftarrow l + 1$; ▷ slide the window to the right
$Seg_{\xi} \leftarrow \{\xi_{p_0,p_1}, \xi_{p_1,p_2}, \ldots, \xi_{p_{s-1},p_s}\}$ for $CP = \{p_0, p_1, \ldots, p_s\}$; ▷ segment ξ as per CP
for $seg \in Seg$ **do**
  if *seg is not fittable by* $\mathcal{N}$ **then**
    $Seg_{\xi} \leftarrow Seg_{\xi} \setminus \{seg\}$; ▷ drop non-fittable segments in Seg_{ξ}
return Seg_{ξ};

mode dynamics", neither vice versa (for different initial values); see examples in Section 5.2.3. In contrast, DAINARX employs a *threshold-free* method based on NARX model fitting.

We start by defining a simple criterion called *mergeable*:

Definition 11 (Mergeable Traces). *A set S of discrete-time trace segments is* mergeable *w.r.t. the template NARX model $\mathcal{N}$ if there exists $N \in \langle\mathcal{N}\rangle$ such that $N \models S$.*

Determining whether two trace segments are mergeable or not boils down to fitting multiple trace segments a NARX model as established in Section 4.1. Based on this criterion, one can simply cluster all the trace segments $Seg_{\mathcal{D}} = \bigcup_{j=1}^{M} Seg_{\xi_j}$ obtained via segmentation by finding a *maximal partitioning* $Seg_{\mathcal{D}} = S_1 \uplus S_2 \uplus \cdots \uplus S_c$ with $c \in \mathbb{N}_{>0}$ clusters such that all segments within the same cluster S_i are mergeable yet segments across different clusters are not mergeable.

The notion of mergeable is simple; it guarantees that all trace segments generated by the same system mode are mergeable. However, the reverse does not hold: The set of mergeable segments may actually be generated by *different* mode dynamics, as demonstrated by the following example.

Example 12 (Aggressive Merging). Consider four trace segments with a single output variable x:

$$\xi_1 = \langle(0,1),(1,2),(2,3),(3,4),(4,5)\rangle\,, \qquad \xi_3 = \langle(0,1),(1,\mathrm{e}),(2,\mathrm{e}^2),(3,\mathrm{e}^3),(4,\mathrm{e}^4)\rangle,$$
$$\xi_2 = \langle(0,1),(1,3),(2,5),(3,7),(4,9)\rangle\,, \qquad \xi_4 = \langle(0,0),(1,0),(2,0),(3,0),(4,0)\rangle\,.$$

Observe that ξ_1 and ξ_2 may be generated (i) by the same second-order mode dynamics $x^{(2)} = 0$; or (ii) by different first-order mode dynamics of $x^{(1)} = 1$ and $x^{(1)} = 2$, respectively. However, the mergeable criterion à la Definition 11 applies only to the former case: ξ_1 and ξ_2 are mergeable because they are fittable by the same second-order NARX model $x[\tau] = 2x[\tau-1] - x[\tau-2]$. The same argument holds for trace segments ξ_3 and ξ_4: they may be generated (i) by the same first-order mode dynamics $x^{(1)} = x$; or (ii) by different mode dynamics of $x^{(1)} = x$ (of order 1) and $x^{(0)} = 0$ (of order 0), respectively. However, the mergeable criterion applies only to the former case: ξ_3 and ξ_4 are mergeable because they are fittable by the same first-order NARX model $x[\tau] = \mathrm{e}x[\tau-1]$. ◁

DAINARX provides a less aggressive merging criterion called *minimally mergeable*:

Definition 13 (Minimally Mergeable Traces). *Given a set S of discrete-time trace segments. Let $\hat{k}_S \triangleq \min\{k \in \mathbb{N} \mid k$ is the order of $N \in \langle\mathcal{N}\rangle$ such that $N \models S\}$ be the minimal order of NARX models that fit S under template $\mathcal{N}$. Then, S is* minimally mergeable *w.r.t. $\mathcal{N}$ if $\hat{k}_S = \hat{k}_{\{\xi\}}$ for any $\xi \in S$.*

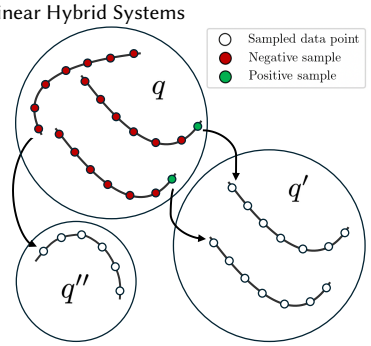


Fig. 4. Samples in $(q, q')^+, (q, q')^-$.

Determining whether two trace segments are minimally mergeable or not is analogous to that for the mergeable criterion, except that we need to incrementally enumerate the order from 0 to k.

By applying the minimally mergeable criterion, ξ_1 and ξ_2 in Example 12 can no longer be merged, nor can ξ_3 and ξ_4. However, we emphasize that, due to the limited information carried by the learning data, we can often not determine whether two traces should be merged or not (recall Example 12). Therefore, DAINARX provides the two different merging strategies (mergeable and minimally mergeable) as an option controlled by the users. In practice, we observe that the simple mergeable criterion à la Definition 11 suffices to yield good clustering results.

4.4 Mode Characterization

Given the clustered trace segments $Seg_{\mathcal{D}} = S_1 \uplus S_2 \uplus \cdots \uplus S_c$, the process of mode characterization aims to construct the corresponding set of system modes $Q_{\mathcal{D}} = \{q_1, q_2, \ldots, q_c\}$ by learning a NARX model N_i adhering to the given template $\mathcal{N}$ as the dynamics for each cluster S_i in mode q_i $(1 \leq i \leq c)$. We achieve this by an immediate application of our LLSQ-based trace fitting technique established in Section 4.1: We find $N_i \in \langle \mathcal{N} \rangle$ such that $N_i \models S_i$ by solving the linear optimization problem (5) with (6). Note that if the minimally mergeable criterion is chosen for segment clustering, we will instead find N_i of order $\hat{k}_{S_i}$ that fits S_i.

The correctness of mode characterization is a direct consequence of the correctness of the previous steps: Since our segmentation procedure ensures that every trace segment in $Seg_{\mathcal{D}}$ is fittable by the template $\mathcal{N}$ whilst the clustering process guarantees that all the segments in S_i are (minimally) mergeable, we are able to find N_i (of order $\hat{k}_{S_i}$) that fits S_i (cf. Theorem 7).

4.5 Guard Learning

Guard learning aims to learn a guard condition $g \subseteq \mathbb{R}^{n+m}$ for each mode switching. This task can be viewed as a (binary) classification problem to predict whether a transition e between two modes will occur or not. Specifically, given two modes $q, q' \in Q_{\mathcal{D}}$, guard learning identifies the *decision boundary* that separates the region in the continuous state space $\mathbb{R}^{n+m}$ where the system will transition to q' from the region where it either remains in q or transitions to a mode different from q'. Recall that we assume a deterministic semantics of transitions, i.e., the guard conditions for the same (source) mode do *not* overlap in $\mathbb{R}^{n+m}$.

Considering the potential nonlinearity of guards $g \in G$ in the original hybrid automaton $\mathcal{H}$, we employ SVMs [12, 61] to effectively identify the decision boundaries in a similar fashion as [23, 27]. Let $Z = Z^+ \uplus Z^-$ be a training dataset consisting of positive samples Z^+ and negative samples Z^-. For linear decision boundaries, SVM gives the optimal hyperplane by maximizing the margin between points in Z^+ and Z^-, thus exhibiting strong generalization capabilities; For nonlinear decision boundaries (where Z^+ and Z^- may not be linearly separable), SVM can still yield effective classification using space transformations and kernel tricks (e.g., polynomial or RBF kernels) [12].

Below, we show how our guard learning problem can be reduced to SVMs by constructing the positive and negative datasets of samples. Let $\xi_j(\tau)$ be the τ-th timestamp-free data point $\vec{v}_\tau$ on trace ξ_j; let $\mathcal{M}_j(\tau)$ be the mode $q_i \in Q_{\mathcal{D}}$ such that the trace segment containing $\xi_j(\tau)$ is clustered into S_i.[6] Then, for any pair of modes $q, q' \in Q_{\mathcal{D}}$, the corresponding positive dataset $(q,q')^+$ and negative dataset $(q,q')^-$ are constructed as

$$(q,q')^+ = \bigcup_{j=1}^{M} \left\{ \xi_j(\tau) \mid \mathcal{M}_j(\tau) = q \text{ and } \mathcal{M}_j(\tau+1) = q' \right\},$$
$$(q,q')^- = \bigcup_{j=1}^{M} \left\{ \xi_j(\tau) \mid \mathcal{M}_j(\tau) = q \text{ and } \mathcal{M}_j(\tau+1) \neq q' \right\}.$$

Note that in $(q,q')^+$, $\tau+1$ is in fact a changepoint identified on trace ξ_j; moreover, we have $(q,q')^+ = \emptyset$ if our learning data $\mathcal{D}$ does not reflect a transition from q to q'. Fig. 4 illustrates the intuition of constructing $(q,q')^+$ and $(q,q')^-$.

4.6 Reset Learning

The final step aims to fill the hole in every transition $e = (q, g, \cdot, q') \in \mathcal{T}$ by learning a *linear* reset function $r\colon \mathbb{R}^{k\times n+m} \to \mathbb{R}^{k\times n}$ such that the output state vectors $\vec{x}, \vec{x}^{(1)}, \ldots, \vec{x}^{(k-1)}$ are updated to $r(\vec{x}, \vec{x}^{(1)}, \ldots, \vec{x}^{(k-1)}, \vec{u})$ upon taking this transition (triggered by the activation of g, i.e., $(\vec{x}, \vec{u}) \in g$).

For the simple case of first-order dynamics with $k = 1$, learning the reset function can be done via *linear regression* to capture the relation between the data points before and after the transition, regardless of the type of dynamics (linear or nonlinear); see, e.g., LearnHA [23] for more details.

However, for the case of high-order dynamics with $k > 1$, the same method based on linear regression does not apply because the values of higher-order derivatives $(\vec{x}^{(1)}, \ldots, \vec{x}^{(k-1)})$ are not present in our learning data $\mathcal{D}$. Therefore, we employ a different form of reset function as

$$r\colon \mathbb{R}^{k\times n+m} \to \langle \mathcal{N} \rangle^k, \quad (\vec{x}[\tau], \vec{x}[\tau-1], \ldots, \vec{x}[\tau-k+1], \vec{u}[\tau]) \mapsto \{N_1, N_2, \ldots, N_k\}, \tag{7}$$

such that each $N_i\colon (\vec{x}[\tau-1+i], \ldots, \vec{x}[\tau-k+i], \vec{u}[\tau+i]) \mapsto \vec{x}[\tau+i]$ can be used to generate $\vec{x}[\tau+i]$ for the new mode. Next, we show how to construct the k NARX models in (7) as the reset of transition $e = (q, g, \cdot, q')$ via NARX model fitting. For any $1 \le i \le k$,

$$N_i \models \bigcup_{j=1}^{M} \left\{ \xi_{l,l+k+1} \subseteq \xi_j \mid \xi_j(l+k-i) \in (q,q')^+ \right\},$$

where $\xi \subseteq \xi'$ means that ξ is a segment of ξ'; that is, we fit $N_i \in \langle \mathcal{N} \rangle$ using the data points on the i-th segment (of length $k+1$) enclosing the mode-switching point.

For hybrid systems with linear dynamics, we *conjecture* that – if the sampling rate of our learning data $\mathcal{D}$ is beyond the Nyquist frequency [16] – the above reset learning technique based on NARX model fitting is guaranteed to identify the correct resets (subject to the learning data $\mathcal{D}$). The rationale behind this conjecture is the Nyquist-Shannon sampling theorem [16] in signal processing and the Picard–Lindelöf theorem (aka, the Cauchy–Lipschitz theorem) [5] for initial value problems – these two results *may* be used to establish that one can perfectly reconstruct the original continuous-time signal adhering to linear ODEs from (sufficiently dense) data samples.

For nonlinear dynamics, due to the complex nature of the induced signal, we do not foresee a similar conjecture as for the linear case. However, we still observe effective approximations of the reset functions in practice, as demonstrated in Sections 2 and 5.2.2.

[6] $\mathcal{M}_j(\tau)$ does not exist if the trace segment containing $\xi_j(\tau)$ is screened out during segmentation.

Time Complexity. The total *worst-case* time complexity of DAINARX is $O(|\mathcal{D}|^3 \cdot n + |\mathcal{D}|^2 \cdot d^2 \cdot n)$ (see details in Appendix C), i.e., *polynomial* in the size of the learning data $\mathcal{D}$ and the size of the target system $d = (\alpha + kn + m + 1)$. In practice, however, the complexity is much lower (bounded by $O(|\mathcal{D}| \cdot d^2 \cdot n \cdot (w + |Seg_{\mathcal{D}}|))$.

5 Implementation and Experimental Results

We have implemented DAINARX as a prototype tool (of the same name) in Python 3.9.[7] By interfacing with NumPy [24] (for LLSQ and matrix operations) as well as scikit-learn [45] (for SVM classifications), DAINARX infers a possibly nonlinear hybrid automaton with high-order NARX-modeled dynamics (subject to template $\mathcal{N}$) from input-output discrete-time traces $\mathcal{D}$. This section reports the evaluation results of DAINARX in comparison with state-of-the-art tools on a collection of benchmarks. All experiments are conducted on an 8-core Apple M2 processor with 16GB RAM. The experiments are designed to evaluate DAINARX in terms of *effectiveness*, and *efficiency*.

5.1 Experimental Setup

DAINARX *Configurations.* DAINARX supports user-defined configurations of template NARX models, SVM parameters, and other experimental settings through a dedicated JSON file for each benchmark.[8] We note that, in contrast to threshold-based approaches, DAINARX does *not* require user-supplied, fine-tuned parameter valuations for both segmentation and clustering. In our experiments, the floating-point tolerance is set to 1×10^{-7}. We use the RKF45 method (the Runge–Kutta–Fehlberg method of order 4 with an error estimator of order 5 [5]) to obtain discrete-time traces of ODEs with fixed sampling periods; For each benchmark, we sample 15 traces – 9 used for training (learning) the automaton and 6 used for testing the automaton.

Baselines. We compare DAINARX agaist two state-of-the-art implementations (with fine-tuned parameters) for hybrid system identification: FaMoS [47] and LearnHA [23]. The former is dedicated to learning hybrid automata with high-order dynamics captured by *linear* ARX models, whilst the latter aims to infer automata with *first-order* polynomial ODEs. Both frameworks align with the methodology of decomposing the inference problem into a similar sequence of subproblems. While HAutLearn [66] is another notable method in this domain, empirical findings in [47] demonstrate FaMoS' distinct advantages over HAutLearn in terms of performance and generalizability.

Benchmarks. We collect 26 hybrid automata from the literature. Of there, 18 are linear systems: 7 from FaMoS [47], 4 from LearnHA [23], 7 additional curated instances: a synthetic minimal system, a two-tank system, (2- and 4-mode variants of) underdamped systems, a closed-loop control system, a PID-controlled DC motor position system, and a jumping robotic dog [11]. The nonlinear subset consists of 8 hybrid systems adapted from the literature with polynomial, rational, or trigonometric dynamics as well as guard conditions involving polynomial or trigonometric constraints. The benchmarks exhibit 2–4 modes in most cases, extending to a maximum of 8 modes (`complex_tank`); The number of output variables typically ranges from 2 to 4, with one outlier featuring 9 output variables (`sys_bio`). While most examples have 1st- or 2nd-order dynamics, we include a 4th-order system to evaluate DAINARX's scalability to higher-order regimes (`dc_motor`).

5.2 Effectiveness of DAINARX

Below, we first evaluate DAINARX against LearnHA and FaMoS – on both linear (Section 5.2.1) and nonlinear (Section 5.2.2) hybrid systems – in terms of the *accuracy of mode-switching detection*,

[7] Available at https://github.com/FICTION-ZJU/Dainarx.
[8] The default parameter values and the configurations used in our experiments are available in the artifact.

Table 2. Experimental results on inferring hybrid automata with linear dynamics and guards. $|Q|$, $|X|$, and k represent the number of modes, the number of output variables, and the highest order of ODEs, respectively; HDT_c is the Hausdorff distance (in time) of the sequence of identified changepoints with the sequence of genuine mode-switching points; $Diff_{max}$ ($Diff_{avg}$, resp.) represents the maximal (average, resp.) point-wise difference between the traces of the inferred system and the original traces (ground truth). `tanks` contains one input variable; `ball` involves linear resets; `underdamped-c` contains both linear resets and an input variable. "–" indicates that a tool does not suffice for the inference task of the corresponding benchmark.

Benchmark Details				HDT_c (seconds)			$Diff_{max}$			$Diff_{avg}$		
Name	$\|Q\|$	$\|X\|$	k	LearnHA	FaMoS	DAINARX	LearnHA	FaMoS	DAINARX	LearnHA	FaMoS	DAINARX
`buck_converter` [47]	3	2	1	0.0001	0.0004	**0.0000**	0.1674	0.2065	**0.0000**	0.0046	0.0142	**0.0000**
`complex_tank` [47]	8	3	1	3.6350	1.6350	**0.0950**	0.3483	0.2411	**0.0451**	0.0655	0.0151	**0.0028**
`multi_room_heating` [47]	4	3	1	2.9750	0.0600	**0.0350**	0.2531	0.0094	**0.0078**	0.0590	0.0013	**0.0008**
`simple_heating_syst` [47]	2	1	1	0.0400	0.4400	**0.0200**	0.0202	0.2126	**0.0102**	0.0037	0.0423	**0.0007**
`three_state_HA` [47]	3	1	2	–	0.5700	**0.0000**	–	0.5388	**0.0000**	–	0.0178	**0.0000**
`two_state_HA` [47]	2	1	2	–	0.3400	**0.0000**	–	0.1284	**0.0000**	–	0.0132	**0.0000**
`variable_heating_syst` [47]	3	2	1	0.1100	0.0700	**0.0300**	0.0581	0.0272	**0.0200**	0.0082	0.0012	**0.0005**
`cell` [23]	4	1	1	**0.0100**	29.2800	**0.0100**	**0.0176**	1.6144	**0.0176**	**0.0001**	0.1494	0.0002
`oci` [23]	2	2	1	0.0500	–	**0.0000**	0.2002	–	**0.0000**	0.0259	–	**0.0000**
`tanks` [23]	4	2	1	13.9100	–	**0.0100**	1.1077	–	**0.0177**	0.2589	–	**0.0007**
`ball` [23]	1	2	1	**0.0000**	–	**0.0000**	**0.0000**	–	**0.0000**	**0.0000**	–	**0.0000**
`dc_motor`	2	2	4	–	–	**0.0000**	–	–	**0.0000**	–	–	**0.0000**
`simple_linear`	2	2	1	7.9600	0.0600	**0.0000**	0.9999	0.1107	**0.0000**	0.1448	0.0071	**0.0000**
`jumper` [11]	2	4	1	0.4500	–	**0.0000**	1.8182	–	**0.0000**	0.0899	–	**0.0000**
`loop_syst`	4	2	2	–	4.9100	**0.0000**	–	1.8929	**0.0000**	–	0.2470	**0.0000**
`two_tank`	2	2	1	**0.0000**	8.6400	**0.0000**	**0.0000**	1.5273	**0.0000**	**0.0000**	0.2324	**0.0000**
`underdamped`	2	2	2	–	–	**0.0000**	–	–	**0.0000**	–	–	**0.0000**
`underdamped-c`	4	2	2	–	–	**0.0100**	–	–	**0.0080**	–	–	**0.0004**

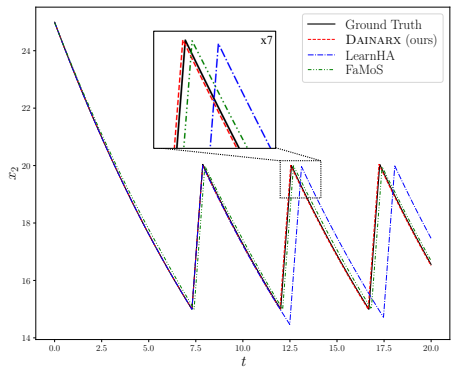


Fig. 5. Trace fidelity for `complex_tank`.

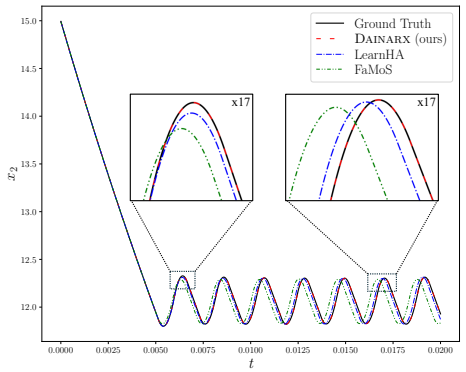
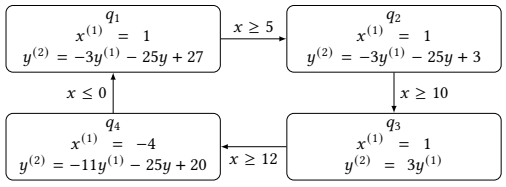


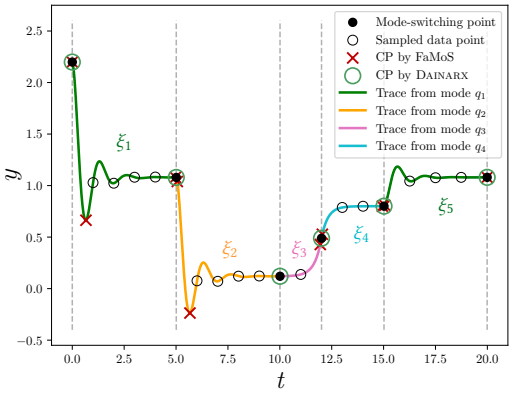
Fig. 6. Trace fidelity for `buck_converter`.

trace fidelity, and *applicability*. Empirical results demonstrate DAINARX's overall superiority in these metrics. Then, we conduct a case study to showcase the rationale behind the advantage of DAINARX in comparison to derivative-based segmentation and trace-similarity-based clustering.

5.2.1 Inference of Linear Hybrid Systems. Table 2 reports the experimental results on inferring hybrid automata with linear dynamics and guards. In this category, DAINARX exhibits a distinct advantage in effectiveness: It achieves the *(tied-)highest accuracy* in detecting mode switching (signified by HDT_c) in all the 18 benchmarks; It also attains the *highest trace fidelity* (i.e., minimal discrepancies between inferred traces and ground-truth traces, signified by $Diff_{max}$ and $Diff_{avg}$) across all the benchmarks except for a comparable result with LearnHA for the `cell` example. Figs. 5 and 6 depict a more intuitive comparison of the performance on fitting testing traces.



(a) The hybrid automaton of `loop_syst`.



(b) Segmentation and clustering of `loop_syst`.

Fig. 7. Case study on `loop_syst`.

In terms of applicability, DAINARX suffices to infer the underlying HA for *all* the linear benchmarks with considerably accurate trace approximations. In contrast, LearnHA does not apply to high-order dynamics (with $k > 1$), which is a crucial feature in modeling real-world dynamical systems. FaMoS fails in some benchmarks because of the presence of resets or inputs (`tanks`, `ball`, `underdamped-c`) or implementation instability in clustering (the rest benchmarks marked by "–").

5.2.2 Inference of Nonlinear Hybrid Systems. Next, we report the experimental results on learning hybrid automata with nonlinear dynamics and/or guards. In this category, we exclude FaMoS and LearnHA due to the following reasons: (i) FaMoS is dedicated to learning *linear* hybrid systems; and (ii) In principle, LearnHA supports polynomial ODEs and guard conditions. However, most of the LearnHA benchmarks (4 out of 5) in [23] involve linear systems whilst the only nonlinear case lacks implementation details in its documentation. When applied to our nonlinear benchmarks, LearnHA generates hybrid automata with *spurious variables* absent from the ground-truth specifications, making it challenging to evaluate the trace fidelity. Furthermore, LearnHA does not cope with systems with transcendental functions (e.g., trigonometric dynamics) or high-order nonlinearities. We were thus not able to obtain valid inference results for nonlinear cases using LearnHA.

Table 3. Results on inferring hybrid automata with nonlinear dynamics and/or guards. `lotkaVolterra`, `spacecraft`, and `duffing` feature nonlinear guards; `oscillator` contains trigonometric dynamics and guards; `duffing` has linear resets and inputs.

Benchmark Details				DAINARX		
Name	$\|Q\|$	$\|X\|$	k	HDT_c	$Diff_{max}$	$Diff_{avg}$
`lander` [68]	2	4	1	0.010	0.0024	0.0000
`lotkaVolterra` [20]	2	2	1	0.020	0.0047	0.0006
`simple_non_linear`	2	1	1	0.006	0.0367	0.0020
`simple_non_poly`	2	1	1	0.000	0.0000	0.0000
`oscillator` [49]	2	2	1	0.000	0.0000	0.0000
`spacecraft` [13]	2	4	1	0.000	0.0000	0.0000
`sys_bio` [64]	2	9	1	0.012	0.0960	0.0081
`duffing`	2	1	2	0.001	0.0003	0.0000

Table 3 summarizes DAINARX's performance on the inference of nonlinear hybrid systems. Akin to the linear case, DAINARX achieves fairly high accuracy both in detecting mode switching and in approximating the ground-truth traces. These results demonstrate the general applicability of DAINARX to high-order nonlinear systems beyond polynomials, e.g., rational and transcendental dynamics and/or guards. Notably, DAINARX retains high fidelity even in regimes where the other two baselines fail, underscoring its robustness in handling diverse and complex nonlinear systems.

5.2.3 Rationale behind Segmentation and Clustering. We conduct a case study using a closed-loop control system (`loop_syst`; see Fig. 7a) to intuitively explain why the segmentation and clustering

Table 4. Experimental results on the execution time (in seconds) of the inference tools.

Benchmark Details				Segmentation Time			Clustering Time			Guard-Learning Time			Total Time		
Name	$\|Q\|$	$\|X\|$	k	LearnHA	FaMoS	DAINARX	LearnHA	FaMoS	DAINARX	LearnHA	FaMoS	DAINARX	LearnHA	FaMoS	DAINARX
buck_converter [47]	3	2	1	2.10	**0.12**	1.06	25.60	3.28	**0.93**	0.95	0.03	**0.02**	28.70	4.43	**2.12**
complex_tank [47]	8	3	1	4.65	**0.53**	1.70	34.00	18.40	**3.46**	1.84	**0.03**	0.34	40.50	19.00	**5.73**
multi_room_heating [47]	4	3	1	2.90	**0.35**	1.12	19.12	7.71	**1.57**	1.22	**0.03**	0.12	23.30	8.12	**2.96**
simple_heating_syst [47]	2	1	1	2.09	**0.26**	0.52	10.70	8.10	**0.19**	0.58	**0.01**	**0.01**	13.40	8.39	**0.78**
three_state_HA [47]	3	1	2	–	**0.14**	0.55	–	1.30	**0.31**	–	0.01	**0.00**	–	1.45	**0.92**
two_state_HA [47]	2	1	2	–	**0.20**	0.54	–	1.50	**0.22**	–	**0.01**	**0.01**	–	1.72	**0.80**
variable_heating_syst [47]	3	2	1	2.10	**0.29**	0.95	8.55	3.93	**0.65**	0.65	**0.01**	0.11	11.30	4.23	**1.79**
cell [23]	4	1	1	9.28	**0.44**	3.38	55.10	39.94	**3.80**	0.80	**0.02**	**0.02**	65.20	40.42	**7.47**
oci [23]	2	2	1	2.14	–	**0.93**	5.83	–	**0.36**	0.48	–	**0.03**	8.46	–	**1.41**
tanks [23]	4	2	1	2.21	–	**1.17**	16.98	–	**1.50**	1.56	–	**1.04**	20.79	–	**3.82**
ball [23]	1	2	1	0.59	–	**0.26**	2.17	–	**0.13**	0.62	–	**0.31**	3.38	–	**0.72**
dc_motor	2	2	4	–	–	**1.50**	–	–	**2.09**	–	–	**0.05**	–	–	**4.08**
simple_linear	2	2	1	2.23	**0.19**	0.60	9.99	3.58	**0.86**	0.56	**0.02**	0.42	12.79	3.80	**1.97**
jumper [11]	2	4	1	0.62	–	**0.11**	2.24	–	**0.11**	**0.49**	–	1.78	3.35	–	**2.38**
loop_syst	4	2	2	–	**0.20**	1.36	–	3.09	**1.57**	–	**0.01**	**0.01**	–	3.30	**3.10**
two_tank	2	2	1	2.04	0.24	**0.18**	4.97	2.69	**0.18**	0.41	**0.01**	3.20	7.42	**2.94**	4.61
underdamped	2	4	1	–	–	**1.42**	–	–	**0.67**	–	–	**0.03**	–	–	**2.25**
underdamped-c	2	2	2	–	–	**1.52**	–	–	**2.65**	–	–	**0.06**	–	–	**4.52**
lander [68]	2	4	1			**1.38**			**1.03**			**0.00**			**2.58**
lotkaVolterra [20]	2	2	1			**0.44**			**0.24**			**0.02**			**0.76**
simple_non_linear	2	1	1			**3.04**			**2.28**			**0.02**			**5.70**
simple_non_poly	2	1	1	–	–	**2.46**	–	–	**1.20**	–	–	**0.03**	–	–	**3.96**
oscillator [49]	2	2	1			**0.65**			**0.43**			**0.01**			**1.13**
spacecraft [13]	2	4	1			**1.20**			**0.72**			**0.01**			**2.01**
sys_bio [64]	2	9	1			**13.17**			**12.60**			**0.20**			**27.40**
duffing	2	1	2			**3.92**			**2.98**			**0.06**			**7.34**

in DAINARX – both based on NARX model fitting – yield better results than derivative-based segmentation and trace-similarity-based clustering. For *segmentation*, as shown in Fig. 7b, the derivative-based approaches, like FaMoS, identify spurious mode-switching points exhibiting drastic variations in shape yet fail to detect genuine mode-switching points inducing smooth or subtle transitions.[9] In contrast, DAINARX succeeds in detecting *all* mode-switching points by analyzing intrinsic dynamics of the trace *without* manually specified thresholds. For a fair comparison of *clustering*, we feed the true mode-switching points to FaMoS and DAINARX and evaluate the clustering performance under this condition. We observe that DAINARX's NARX model-fitting-based strategy achieves the correct clustering:

$$\{\xi_1, \xi_5\} \uplus \{\xi_2\} \uplus \{\xi_3\} \uplus \{\xi_4\} .$$

However, the clustering produced by FaMoS is $\{\xi_1, \xi_2\} \uplus \{\xi_3\} \uplus \{\xi_4\} \uplus \{\xi_5\}$. This is because the trace-similarity-based strategy *erroneously* concludes that segments of a similar shape (ξ_1 and ξ_2) are generated by the same dynamics and segments of different shapes (ξ_1 and ξ_5) are not.

5.3 Efficiency of DAINARX

Next, we report the efficiency of DAINARX in comparison to the other baselines. Table 4 summarizes the execution time of different inference tools across all benchmarks. We omit the explicit timings for the steps of mode characterization and reset learning as they are negligible compared to the other three steps. We note that the comparison may not be fair since the tools run on different platforms (FaMoS in MATLAB; DAINARX and LearnHA in Python).

Overall, we observe that both FaMoS and LearnHA suffer from a significant computational overhead for segment clustering, which dominates their total execution time. LearnHA, in particular, exhibits prolonged clustering phases – typically 2–3 times slower than the other steps – resulting

[9]The implementation of FaMoS provides a mechanism to rule out changepoints based on the result of clustering. However, we did not apply this mechanism in this experiment as it has the side-effect of eliminating true mode-switching points.

in suboptimal scalability to large-sized systems. In contrast, DAINARX exhibits a clear advantage in efficiency and scalability: It solves most of the benchmarks (21/26) within 5 seconds. A notable exception is the `sys_bio` benchmark – a nonlinear hybrid system with 9 output variables, which takes 27.4 seconds due to the combinatorial complexity of resolving high-dimensional mode interactions. Except for `two_tank`, DAINARX outperforms both baselines by a wide margin: 1.5–4 times faster than FaMoS and 2–6 times faster than LearnHA. This further demonstrates the potential of applying DAINARX to the inference of complex real-world hybrid system models.

6 Limitations and Future Work

We pinpoint several limitations of DAINARX and provide potential solutions thereof.

First, DAINARX requires – in the nonlinear case – user-provided nonlinear terms $\{f_i\}$ to form the *template* NARX model $\mathcal{N}$. Although we do not observe significant sensitivity of DAINARX's performance to the quality of templates, it still relies on a priori knowledge about the target system to obtain tight approximations. We foresee that machine-learning models and techniques may be developed to *learn a good NARX template* $\mathcal{N}$ based on characteristics of the learning data.

Second, DAINARX does not take into account *noise* in the learning data, thereby exhibiting confined robustness to noisy inputs. Such noise may be introduced during, e.g., external measurements of the system behaviors under perturbations. It is therefore worth investigating robust identification of hybrid systems, e.g., [43], and explore a possible extension of our techniques based on NARX model fitting to cater for robust identification. A viable way is to employ NARX-modeled dynamics with an *error term* $\epsilon[\tau]$ in the same vein as [33, 34, 37] and adapt DAINARX to the extended form.

Third, DAINARX admits a quantitative means to measure the discrepancy between the inferred model and the *sampled learning data instead of the real system trajectories*. This is a common challenge for hybrid system identification as authentic system trajectories are often intractable to obtain, especially for nonlinear dynamics [19]. However, it would be interesting to explore whether techniques like *probably approximately correct* (PAC) learning [59] can be leveraged to establish quantitative guarantees for the inferred hybrid automata, as is for the learning of timed automata [52], symbolic automata [39], and dynamical systems [1, 48].

7 Related Work

Model Learning. The inference of system models from observational data has been a longstanding research focus in computer science. This field traces its origin to Moore's seminal work on Gedanken-experiments for sequential machines in 1956, which formalized the problem of inferring black-box system models through systematic experimentation [40]. Modern model learning [57] frameworks are broadly categorized into passive learning, which constructs models from static datasets, and active learning, which iteratively refines hypotheses via structured queries to oracles. The latter paradigm was pioneered by Angluin's L^* algorithm [4], a cornerstone of automata theory. Model learning methodologies have since been widely applied to infer diverse formal representations, including finite-state machines [57, 58], Markov decision processes [55], timed automata [3, 63], and dynamical systems such as differential equations [30] and switched systems [27]. These approaches span both passive and active paradigms. Model learning techniques have also been utilized for system analysis [8, 17, 51, 65], model checking [18, 46, 53], falsification [7, 62], monitoring [21, 28], etc. Our work is positioned within the domain of passive learning, specifically targeting hybrid systems – the class characterized by mixed discrete and continuous dynamics.

Hybrid System Identification. The problem of deriving dynamical models from observed data is formalized in control theory as system identification [35], proving essential for modeling complex cyber-physical systems [67]. Within this paradigm, hybrid automata inference has emerged as a

critical challenge, with methodologies spanning classical optimization [44], data-driven feature extraction [32, 42], and machine-learning techniques. For instance, [25] proposes a hierarchical deep learning framework using stacked restricted Boltzmann machines (RBMs) to abstract latent features from continuous traces, while [60] introduces a two-step approach combining DyClee clustering with support vector or random forest regression. Despite progress, current approaches exhibit critical constraints. Many frameworks [23, 47, 66] are restricted to linear or polynomial dynamics. Several methods apply exclusively to switched systems [14, 27], precluding generalization to hybrid systems with resets or guard conditions. Learning resets is also a key point that some of the previous work [50] is not capable of handling. [23] introduces type annotation to improve the inference of resets by utilizing domain knowledge. Our method is able to learn nonlinear hybrid automata, including polynomial ones, and can also handle linear resets.

NARX Model Inference. The ARX models have been widely utilized in system identification [10, 44]. In [33], Lauer et al. consider both switched and piecewise ARX models and their nonlinear versions by applying a combination of linear programming and SVM to compute both the discrete state and the sub-models in a single step. In the succeeding work [34], they extended their methods to nonparametric identification using the kernel method in SVM. In [37], a hierarchical Bayesian framework has been proposed to infer switched nonlinear ARX. In [15], an optimization-based approach is proposed for inferring switched ARX models with an active learning loop for result refinement. Recently, FaMos [47] is proposed to learn hybrid automata with linear ARX models.

Our method DAINARX provides an approach to inferring hybrid automata with nonlinear ARX models, advancing the state-of-the-art by addressing several gaps in learning hybrid systems. Specifically, it supports nonlinear hybrid automata with polynomial, rational, or trigonometric dynamics and/or guards while also enabling linear reset inference.

8 Conclusion

We have presented DAINARX – a derivative-agnostic framework for the inference of nonlinear hybrid automata. This framework enables *threshold-free* trace segmentation and clustering leveraging the technique of NARX model fitting throughout its workflow. DAINARX is – to the best of our knowledge – the first approach that admits the inference of *high-order non-polynomial dynamics* with *exogenous inputs*, *non-polynomial guard conditions*, and *linear resets*. Future directions include extending DAINARX to cope with noise and equip it with the PAC learning paradigm.

Acknowledgments

This work has been partially funded by the ZJNSF Major Program (No. LD24F020013), by the Fundamental Research Funds for the Central Universities of China (No. 226-2024-00140), and by the ZJU Education Foundation's Qizhen Talent program. The authors would like to thank XXX for the fruitful discussions and the anonymous reviewers for their constructive feedback.

References

[1] Abhijin Adiga, Chris J. Kuhlman, Madhav V. Marathe, S. S. Ravi, and Anil Vullikanti. 2019. PAC Learnability of Node Functions in Networked Dynamical Systems. In *ICML (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 82–91.

[2] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. 1992. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In *Hybrid Systems (Lecture Notes in Computer Science, Vol. 736)*. Springer, 209–229.

[3] Jie An, Mingshuai Chen, Bohua Zhan, Naijun Zhan, and Miaomiao Zhang. 2020. Learning One-Clock Timed Automata. In *TACAS (1) (Lecture Notes in Computer Science, Vol. 12078)*. Springer, 444–462.

[4] Dana Angluin. 1987. Learning Regular Sets from Queries and Counterexamples. *Inf. Comput.* 75, 2 (1987), 87–106.

[5] Vladimir I Arnold. 1992. *Ordinary differential equations*. Springer Science & Business Media.

[6] Eugene Asarin, Olivier Bournez, Thao Dang, Oded Maler, and Amir Pnueli. 2000. Effective synthesis of switching controllers for linear systems. *Proc. IEEE* 88, 7 (2000), 1011–1025.

[7] Stanley Bak, Sergiy Bogomolov, Abdelrahman Hekal, Niklas Kochdumper, Ethan Lew, Andrew Mata, and Amir Rahmati. 2024. Falsification using Reachability of Surrogate Koopman Models. In *HSCC*. ACM, 27:1–27:13.

[8] Ezio Bartocci, Jyotirmoy Deshmukh, Felix Gigler, Cristinel Mateis, Dejan Nickovic, and Xin Qin. 2020. Mining Shape Expressions From Positive Examples. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 39, 11 (2020), 3809–3820.

[9] Richard Bellman and Robert Kalaba. 1959. On adaptive control processes. *IRE Transactions on Automatic Control* 4, 2 (1959), 1–9.

[10] Stephen A Billings. 2013. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons.

[11] Javier Borquez, Shuang Peng, Yiyu Chen, Quan Nguyen, and Somil Bansal. 2024. Hamilton-Jacobi Reachability Analysis for Hybrid Systems with Controlled and Forced Transitions. In *Robotics: Science and Systems*.

[12] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. 1992. A Training Algorithm for Optimal Margin Classifiers. In *COLT'92*. 144–152. https://doi.org/10.1145/130385.130401

[13] Nicole Chan and Sayan Mitra. 2017. Verifying safety of an autonomous spacecraft rendezvous mission. In *ARCH@CPSWeek (EPiC Series in Computing, Vol. 48)*. EasyChair, 20–32.

[14] Hadi Dayekh, Nicolas Basset, and Thao Dang. 2024. Active Learning of Switched Nonlinear Dynamical Systems. In *CDC*. IEEE, 7834–7839.

[15] Hadi Dayekh, Nicolas Basset, and Thao Dang. 2024. Hybrid system identification through optimization and active learning. *IFAC-PapersOnLine* 58, 11 (2024), 87–92.

[16] Paulo SR Diniz, Eduardo AB Da Silva, and Sergio L Netto. 2010. *Digital signal processing: system analysis and design*. Cambridge University Press.

[17] Paul Fiterau-Brostean, Ramon Janssen, and Frits W. Vaandrager. 2016. Combining Model Learning and Model Checking to Analyze TCP Implementations. In *CAV (2) (Lecture Notes in Computer Science, Vol. 9780)*. Springer, 454–471.

[18] Paul Fiterau-Brostean, Toon Lenaerts, Erik Poll, Joeri de Ruiter, Frits W. Vaandrager, and Patrick Verleg. 2017. Model learning and model checking of SSH implementations. In *SPIN*. ACM, 142–151.

[19] Ting Gan, Mingshuai Chen, Yangjia Li, Bican Xia, and Naijun Zhan. 2018. Reachability analysis for solvable dynamical systems. *IEEE Trans. Automat. Contr.* 63, 7 (2018), 2003–2018.

[20] Luca Geretti, Julien Alexandre Dit Sandretto, Matthias Althoff, Luis Benet, Pieter Collins, Marcelo Forets, Elena Ivanova, Yangge Li, Sayan Mitra, Stefan Mitsch, Christian Schilling, Mark Wetzlinger, and Daniel Zhuang. 2023. ARCH-COMP23 Category Report: Continuous and Hybrid Systems with Nonlinear Dynamics. In *ARCH (EPiC Series in Computing, Vol. 96)*. EasyChair, 61–88.

[21] Bineet Ghosh and Étienne André. 2024. Offline and online energy-efficient monitoring of scattered uncertain logs using a bounding model. *Log. Methods Comput. Sci.* 20, 1 (2024).

[22] Gene H. Golub and Christian H. Reinsch. 2007. Singular value decomposition and least squares solutions. In *Milestones in Matrix Computation*. Oxford University Press, 160–180.

[23] Amit Gurung, Masaki Waga, and Kohei Suenaga. 2023. Learning Nonlinear Hybrid Automata from Input-Output Time-Series Data. In *ATVA (1) (Lecture Notes in Computer Science, Vol. 14215)*. Springer, 33–52.

[24] Charles R. Harris, K. Jarrod Millman, Stéfan van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nat.* 585 (2020), 357–362.

[25] Nemanja Hranisavljevic, Alexander Maier, and Oliver Niggemann. 2020. Discretization of hybrid CPPS data into timed automaton using restricted Boltzmann machines. *Eng. Appl. Artif. Intell.* 95 (2020), 103826.

[26] Jeff C. Jensen, Danica H. Chang, and Edward A. Lee. 2011. A model-based design methodology for cyber-physical systems. In *IWCMC*. IEEE, 1666–1671.

[27] Xiangyu Jin, Jie An, Bohua Zhan, Naijun Zhan, and Miaomiao Zhang. 2021. Inferring Switched Nonlinear Dynamical Systems. *Formal Aspects Comput.* 33, 3 (2021), 385–406.

[28] Sebastian Junges, Sanjit A. Seshia, and Hazem Torfah. 2024. Active Learning of Runtime Monitors Under Uncertainty. In *IFM (Lecture Notes in Computer Science, Vol. 15234)*. Springer, 297–306.

[29] Gabor Karsai, Janos Sztipanovits, Ákos Lédeczi, and Ted Bapty. 2003. Model-integrated development of embedded software. *Proc. IEEE* 91, 1 (2003), 145–164.

[30] Rachael T. Keller and Qiang Du. 2021. Discovery of Dynamics Using Linear Multistep Methods. *SIAM J. Numer. Anal.* 59, 1 (2021), 429–455.

[31] Nikolay A. Kudryashov. 2021. The generalized Duffing oscillator. *Commun. Nonlinear Sci. Numer. Simul.* 93 (2021), 105526.

[32] Imane Lamrani, Ayan Banerjee, and Sandeep K. S. Gupta. 2018. HyMn: Mining linear hybrid automata from input output traces of cyber-physical systems. In *ICPS*. IEEE, 264–269.

[33] Fabien Lauer and Gérard Bloch. 2008. Switched and PieceWise Nonlinear Hybrid System Identification. In *HSCC (Lecture Notes in Computer Science, Vol. 4981)*. Springer, 330–343.

[34] Fabien Lauer, Gérard Bloch, and René Vidal. 2010. Nonlinear hybrid system identification with kernel models. In *CDC*. IEEE, 696–701.

[35] Lennart Ljung. 2015. System Identification: An Overview. In *Encyclopedia of Systems and Control*. Springer.

[36] John Lygeros, Karl Henrik Johansson, Slobodan N. Simic, Jun Zhang, and S. Shankar Sastry. 2003. Dynamical properties of hybrid automata. *IEEE Trans. Autom. Control.* 48, 1 (2003), 2–17.

[37] Ahmad Madary, Hamid Reza Momeni, Alessandro Abate, and Kim G. Larsen. 2022. Hierarchical identification of nonlinear hybrid systems in a Bayesian framework. *Inf. Comput.* 289 (2022), 104947.

[38] Oded Maler, Zohar Manna, and Amir Pnueli. 1991. From Timed to Hybrid Systems. In *REX Workshop (Lecture Notes in Computer Science, Vol. 600)*. Springer, 447–484.

[39] Oded Maler and Irini-Eleftheria Mens. 2017. A Generic Algorithm for Learning Symbolic Automata from Membership Queries. In *Models, Algorithms, Logics and Tools (Lecture Notes in Computer Science, Vol. 10460)*. Springer, 146–169.

[40] Edward F Moore et al. 1956. Gedanken-experiments on sequential machines. *Automata studies* 34 (1956), 129–153.

[41] Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. 1992. An Approach to the Description and Analysis of Hybrid Systems. In *Hybrid Systems (Lecture Notes in Computer Science, Vol. 736)*. Springer, 149–178.

[42] Oliver Niggemann, Benno Stein, Asmir Vodencarevic, Alexander Maier, and Hans Kleine Büning. 2012. Learning Behavior Models for Hybrid Timed Systems. In *AAAI*. AAAI Press, 1083–1090.

[43] Necmiye Ozay, Constantino M. Lagoa, and Mario Sznaier. 2009. Robust identification of switched affine systems via moments-based convex optimization. In *CDC*. IEEE, 4686–4691.

[44] Simone Paoletti, Aleksandar Lj. Juloski, Giancarlo Ferrari-Trecate, and René Vidal. 2007. Identification of Hybrid Systems: A Tutorial. *Eur. J. Control* 13, 2-3 (2007), 242–260.

[45] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* 12 (2011), 2825–2830.

[46] Doron A. Peled, Moshe Y. Vardi, and Mihalis Yannakakis. 2002. Black Box Checking. *J. Autom. Lang. Comb.* 7, 2 (2002), 225–246.

[47] Swantje Plambeck, Aaron Bracht, Nemanja Hranisavljevic, and Görschwin Fey. 2024. FaMoS – Fast Model Learning for Hybrid Cyber-Physical Systems using Decision Trees. In *HSCC*. ACM, 7:1–7:10.

[48] Zirou Qiu, Abhijin Adiga, Madhav V. Marathe, S. S. Ravi, Daniel J. Rosenkrantz, Richard Edwin Stearns, and Anil Kumar S. Vullikanti. 2024. Efficient PAC Learnability of Dynamical Systems Over Multilayer Networks. In *ICML*. OpenReview.net.

[49] Nacim Ramdani and Nedialko S. Nedialkov. 2009. Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint propagation techniques. In *ADHS (IFAC Proceedings Volumes, Vol. 42)*. Elsevier, 156–161.

[50] Iman Saberi, Fathiyeh Faghih, and Farzad Sobhi Bavil. 2023. A Passive Online Technique for Learning Hybrid Automata from Input/Output Traces. *ACM Trans. Embed. Comput. Syst.* 22, 1 (2023), 9:1–9:24.

[51] Colin Shea-Blymyer and Houssam Abbas. 2021. Learning a Robot's Social Obligations from Comparisons of Observed Behavior. In *ARSO*. IEEE, 15–20.

[52] Wei Shen, Jie An, Bohua Zhan, Miaomiao Zhang, Bai Xue, and Naijun Zhan. 2020. PAC Learning of Deterministic One-Clock Timed Automata. In *ICFEM (Lecture Notes in Computer Science, Vol. 12531)*. Springer, 129–146.

[53] Junya Shijubo, Masaki Waga, and Kohei Suenaga. 2021. Efficient Black-Box Checking via Model Checking with Strengthened Specifications. In *RV (Lecture Notes in Computer Science, Vol. 12974)*. Springer, 100–120.

[54] Miriam García Soto, Thomas A. Henzinger, and Christian Schilling. 2021. Synthesis of hybrid automata with affine dynamics from time-series data. In *HSCC*. ACM, 2:1–2:11.

[55] Martin Tappler, Bernhard K. Aichernig, Giovanni Bacci, Maria Eichlseder, and Kim G. Larsen. 2019. L^*-Based Learning of Markov Decision Processes. In *FM (Lecture Notes in Computer Science, Vol. 11800)*. Springer, 651–669.

[56] Charles Truong, Laurent Oudre, and Nicolas Vayatis. 2020. Selective review of offline change point detection methods. *Signal Process.* 167 (2020).

[57] Frits W. Vaandrager. 2017. Model learning. *Commun. ACM* 60, 2 (2017), 86–95.

[58] Frits W. Vaandrager, Masoud Ebrahimi, and Roderick Bloem. 2023. Learning Mealy machines with one timer. *Inf. Comput.* 295, Part B (2023), 105013.

[59] Leslie G. Valiant. 1984. A Theory of the Learnable. In *STOC*. ACM, 436–445.

[60] Amaury Vignolles, Elodie Chanthery, and Pauline Ribot. 2022. Hybrid Model Learning for System Health Monitoring. *IFAC-PapersOnLine* 55, 6 (2022), 7–14.

[61] Vapnik Vladimir. 1963. Pattern recognition using generalized portrait method. *Automation and remote control* 24 (1963), 774–780.
[62] Masaki Waga. 2020. Falsification of cyber-physical systems with robustness-guided black-box checking. In *HSCC*. ACM, 11:1–11:13.
[63] Masaki Waga. 2023. Active Learning of Deterministic Timed Automata with Myhill-Nerode Style Characterization. In *CAV (1) (Lecture Notes in Computer Science, Vol. 13964)*. Springer, 3–26.
[64] Qiuye Wang, Mingshuai Chen, Bai Xue, Naijun Zhan, and Joost-Pieter Katoen. 2022. Encoding inductive invariants as barrier certificates: Synthesis via difference-of-convex programming. *Inf. Comput.* 289, Part (2022), 104965.
[65] Kandai Watanabe, Georgios Fainekos, Bardh Hoxha, Morteza Lahijanian, Danil V. Prokhorov, Sriram Sankaranarayanan, and Tomoya Yamaguchi. 2023. Timed Partial Order Inference Algorithm. In *ICAPS*. AAAI Press, 639–647.
[66] Xiaodong Yang, Omar Ali Beg, Matthew Kenigsberg, and Taylor T. Johnson. 2022. A Framework for Identification and Validation of Affine Hybrid Automata from Input-Output Traces. *ACM Trans. Cyber Phys. Syst.* 6, 2 (2022), 13:1–13:24.
[67] Ye Yuan, Xiuchuan Tang, Wei Pan, Xiuting Li, Wei Zhou, Hai-Tao Zhang, Han Ding, and Jorge M. Gonçalves. 2018. Data-driven Discovery of Cyber-Physical Systems. *CoRR* abs/1810.00697 (2018).
[68] Hengjun Zhao, Mengfei Yang, Naijun Zhan, Bin Gu, Liang Zou, and Yao Chen. 2014. Formal Verification of a Descent Guidance Control Program of a Lunar Lander. In *FM (Lecture Notes in Computer Science, Vol. 8442)*. Springer, 733–748.

A Proofs of Theorems

Theorem 7 (Correctness of Trace Fitting). *Given a template NARX model $\mathcal{N}$ and a set of finite discrete-time trace (segments) $S = \{\xi_j\}_{1\le j\le M}$. Let $E_S = \max\{E_1, E_2, \ldots, E_n\}$ where each E_i is the optimal value of (5) with (6) w.r.t. i; let $N = \mathcal{N}[\Lambda] \in \langle\mathcal{N}\rangle$ be the NARX model corresponding to the optimum Λ. Then, S is fittable by $\mathcal{N}$ if and only if $E_S = 0$. In this case, we have $N \models S$.*

Proof. We first show that $E_S = 0$ and $N \models S$ are equivalent. Since $E_i = \|\cdot\|_2 \ge 0$, $E_S = 0$ implies $E_i = 0$ for any i and $O_{i,:} - \Lambda_{i,:} \cdot D_i$ is the corresponding form for variable x_i in (4). Namely, $E_S = 0$ indicates that (4) is true for every variable, i.e. $N \models S$. Next, we show that $E_S = 0$ if and only if S is fittable by $\mathcal{N}$. Since the above proof shows that $E_S = 0$ yields an instance $N \models S$, we only need to prove that $E_S = 0$ when S is fittable by $\mathcal{N}$. For any instance $N' = \mathcal{N}[\Lambda'] \in \langle\mathcal{N}\rangle$ such that $N' \models S$, by substituting Λ' for Λ in (5), we get $E_S = 0$ according to (4). This completes the proof. □

Lemma 14 (Monotonicity of Fittability). *Given a template NARX model $\mathcal{N}$ and two traces ξ, ξ', where ξ' is a segment of ξ. If ξ is fittable by $\mathcal{N}$, then ξ' is fittable by $\mathcal{N}$.*

Proof. Since ξ is fittable by $\mathcal{N}$, according to Definition 5, there exists $N \in \langle\mathcal{N}\rangle$ that (4) holds for all τ. Since ξ' is a segment of ξ, (4) holds for all τ also for ξ', i.e., ξ' is fittable by $\mathcal{N}$. □

Lemma 15 (Uniqueness of CP). *Given a template NARX model $\mathcal{N}$ and a trace ξ. The set of changpoints defined by Property 9 is unique.*

Proof. We have $CP = \{p_0, p_i, \ldots p_s\}$ where p_i satisfies the well-ordered relation: $p_0 = 0 < p_1 < \ldots p_i < \ldots < p_s = \ell + 1$. We prove the claim by induction on i. For the base case, we have $p_0 = 0$ is unique; For the induction step, assume p_i is unique, we prove by contradiction: Suppose p_{i+1} is not unique, i.e., there exists $p_{i+1} < p'_{i+1}$ satisfying that both $\xi_{p_i,p_{i+1}}$ and $\xi_{p_i,p'_{i+1}}$ are fittable by $\mathcal{N}$, but $\xi_{p_i,(p_{i+1})+1}$ is not fittable by $\mathcal{N}$. Since $\xi_{p_i,(p_{i+1})+1}$ is a segment of $\xi_{p_i,p'_{i+1}}$, according to Lemma 14, $\xi_{p_i,(p_{i+1})+1}$ is fittable by $\mathcal{N}$ leading to a contradiction. This completes the proof. □

Theorem 10 (Correctness of Segmentation in Algorithm 1). *Given a discrete-time trace ξ and a template NARX model $\mathcal{N}$. The set Seg_ξ of trace segments returned by Algorithm 1 – in the absence of the above three corner cases – satisfies the changepoint property as stated in Property 9.*

Proof. In the absence of the three corner cases, we have that the changepoint interval is greater than w, and the segment between two changepoints are fittable by $\mathcal{N}$. We also know that $\{\xi, \xi'\}$is fittable by $\mathcal{N}$ when (i) the overlap of two segments ξ, ξ' is greater than $w - 1$, and (ii) ξ, ξ' are fittable by $\mathcal{N}$. Let the changepoints obtained by Algorithm 1 be $CP' = \{p'_0, p'_1, ..., p'_s\}$ and the correct changepoint be $CP = \{p_0, p_1, ...p_c\}$ (which is unique according to Lemma 15). We now prove that $CP' = CP$ via mathematical induction: For the base case, we have $p'_0 = p_0 = 0$; For the induction step, assume $p'_i = p_i$, we first prove that $\xi_{p_i.p'_{i+1}}$ is fittable by $\mathcal{N}$. Since the interval of changpoints is greater than w, ξ_{p_i,p_i+w} is fittable by $\mathcal{N}$. Now we need to prove that for every $h = 0, 1, \ldots p'_{i+1} - w$, ξ_{p_i,p_i+h+w} is fittable by $\mathcal{N}$. We again use induction on h. First, it is proved for $h = 0$; Second, when ξ_{p_i,p_i+h+w} is fittable by $\mathcal{N}$, according to Algorithm 1, we have $\xi_{p_i+h+1,p_i+h+w+1}$ is fittable by $\mathcal{N}$, thus $\{\xi_{p_i,p_i+h+w}, \xi_{p_i+h+1,p_i+h+w+1}\}$ is fittable by $\mathcal{N}$, which implies the fittability of $\xi_{p_i,p_i+h+w+1}$ (as $w > k + 1$) by definition. Hence, by induction, $\xi_{p_i.p'_{i+1}}$ is fittable by $\mathcal{N}$. Now, suppose $p'_{i+1} \neq p_{i+1}$, we do a case distinction: (i) For $p'_{i+1} > p_{i+1}$, we have that $\xi_{p_i,(p_{i+1})+1}$ is not fittable by $\mathcal{N}$ and $\xi_{p_i,(p_{i+1})+1}$ is a segment of $\xi_{p_i,p'_{i+1}}$. According to Lemma 14, $\xi_{p_i,(p_{i+1})+1}$ is fittable by $\mathcal{N}$, which leads to a contradiction; (ii) For $p'_{i+1} < p_{i+1}$, according to Algorithm 1, $\xi_{(p'_{i+1})-w+1,(p'_{i+1})+1}$ is not fittable by $\mathcal{N}$, but $\xi_{(p'_{i+1})-w+1,(p'_{i+1})+1}$ is a segment of $\xi_{p_i,p_{i+1}}$, a contradiction follows by applying Lemma 14. Hence, p'_{i+1} is equal to p_{i+1} and we have that $CP' = CP$. □

B Trace-Segmentation Algorithm via Binary Search

Algorithm 2: Trace Segmentation Based on NARX Model Fitting (via binary search)

Input: A discrete-time trace ξ of length $\ell + 1$ and a template NARX model $\mathcal{N}$ of order $k \in \mathbb{N}$.
Output: A set Seg_ξ of trace segments of ξ under $\mathcal{N}$.

1 $CP \leftarrow \{0\}$; $last\_cp \leftarrow 0$; ▷ initialization
2 **while** $last\_cp < \ell + 1$ **do** ▷ find changepoints via binary search
3 $lb \leftarrow last\_cp + k + 1$; $rb \leftarrow \ell + 1$; ▷ initialize lower- and upper-bounds
4 **while** $lb < rb$ **do**
5 $mid \leftarrow \lceil \frac{lb+rb}{2} \rceil$;
6 **if** $\xi_{last\_cp,mid}$ *is not fittable by* $\mathcal{N}$ **then** ▷ detected potential changepoint
7 $rb \leftarrow mid - 1$;
8 **else**
9 $lb \leftarrow mid$;
10 $CP \leftarrow CP \cup \{rb\}$; $last\_cp \leftarrow rb$; ▷ update the changepoints
11 $Seg_\xi \leftarrow \{\xi_{p_0,p_1}, \xi_{p_1,p_2}, \ldots, \xi_{p_{s-1},p_s}\}$ for $CP = \{p_0, p_1, \ldots, p_s\}$; ▷ segment ξ as per CP
12 **for** $seg \in Seg$ **do**
13 **if** *seg is not fittable by* $\mathcal{N}$ **then**
14 $Seg_\xi \leftarrow Seg_\xi \setminus \{seg\}$; ▷ drop non-fittable segments in Seg_ξ
15 **return** Seg_ξ;

This algorithm exhibits a (logarithmic) increase in time complexity and, as observed in practice, the binary search-based segmentation relies on high-quality template NARX models. Therefore, we opt for the sliding window-based implementation, which yields considerably accurate models even in the absence of high-quality templates, as demonstrated in Section 5.3.

Note that, the *posterior screening* in Lines 12 to 14 of Algorithm 2 ensures that every trace segment in the eventually obtained set Seg_ξ is fittable by the template NARX model $\mathcal{N}$. Such a screening process is necessary due to the following two *corner cases*: (i) the provided template NARX model $\mathcal{N}$ does not suffice to fit some of the trace segments; and (ii) the distance between some neighboring mode-switching points falls below the order k of $\mathcal{N}$. In the absence of these two corner cases, we can establish the correctness of Algorithm 2 as follows:

Theorem 16 (Correctness of Segmentation in Algorithm 2). *Given a discrete-time trace ξ and a template NARX model $\mathcal{N}$. The set Seg_ξ of trace segments returned by Algorithm 2 – in the absence of the above two corner cases – satisfies the changepoint property as stated in Property 9.*

Proof. The proof is similar to that of Theorem 10. We need to show that the set of changepoints CP' obtained by Algorithm 2 coincides with CP. We prove again by induction. For the base case, we have $p'_0 = p_0 = 0$; For the induction step, assume $p'_i = p_i$, we prove that p_{i+1} lies at the bisection boundary, which means that $p'_{i+1} = p_{i+1}$. For any $p_g > p_i + 1$, since $\xi_{p_i,(p_{i+1})+1}$ is a segment of ξ_{p_i,p_g}, if ξ_{p_i,p_g} is fittable by $\mathcal{N}$, by Lemma 14, $\xi_{p_i,(p_{i+1})+1}$ is fittable by $\mathcal{N}$, which contradicts Property 9. Thus, ξ_{p_i,p_g} is not fittable by $\mathcal{N}$. For any $p_i + k < p_l < p_{i+1}$, since $\xi_{p_i,p_{i+1}}$ is fittable by $\mathcal{N}$ and ξ_{p_i,p_l} is a segment of $\xi_{p_i,p_{i+1}}$, it follows from Lemma 14 that ξ_{p_i,p_l} is fittable by $\mathcal{N}$. Therefore, fittability satisfies the dichotomy property and p_{i+1} lies at the dichotomy boundary. Thus, p'_{i+1} obtained by the binary search-based algorithm is equal to p_{i+1}. This completes the proof. □

C Detailed Complexity Analysis of DAINARX

Below, we establish DAINARX's worst-case time complexity.

First, we explore the core procedure of NARX model fitting. Recall that the data matrices D_i in the LLSQ optimization problem (5) is of dimension $(\alpha + kn + m + 1) \times (\ell - k + 1)$, where α and k are respectively the number of nonlinear terms and the order of the template NARX model $\mathcal{N}$, n and m are respectively the size of output variables X and input variables U, and $(\ell + 1)$ is the length of trace to be fitted. Let $d = (\alpha + kn + m + 1)$ be the row cardinality of D_i (encoding the size of the target system). Then, the complexity of solving (5) (e.g., using singular value decomposition [22]) is $O((\ell - k) \cdot d^2)$. Therefore, the total complexity of NARX fitting – by solving n optimization problems – is $O(n \cdot (\ell - k) \cdot d^2)$, denoted as $O(T_N(\ell))$. This complexity reduces to $O(T_N(\ell)) = O(n \cdot \ell \cdot d^2)$ as $\ell \gg k$, which satisfies $O(T_N(\ell_1)) + O(T_N(\ell_2)) = O(T_N(\ell_1 + \ell_2))$.

Then, we analyze the algorithmic complexity of each step in DAINARX. Let $|\mathcal{D}| = (\ell + 1) \cdot M$ be the number of data points in the learning data. Then, (I) *Segmentation*: Each trace segment within the sliding window needs to be fitted, thus yielding the complexity $O(|\mathcal{D}| \cdot T_N(w))$, where w is the window size; (II) *Clustering*: As we have to find mergeable segments[10] by traversing each pair of segments, the complexity is $O(\sum_{\xi,\xi' \in Seg_{\mathcal{D}}} T_N(|\xi| + |\xi'|)) = O(T_N(|\mathcal{D}|) \cdot |Seg_{\mathcal{D}}|)$, and in the worst case, $O(T_N(|\mathcal{D}|) \cdot |\mathcal{D}|)$; (III) *Mode characterization*: To perform NARX fitting for each cluster of traces, the complexity is $O(T_N(|\mathcal{D}|))$; (IV) *Guard learning*: The complexity of SVM is $O((|(q, q')^+| + |(q, q')^-|)^2 \cdot n)$ for each pair (q, q') of modes, and the worst-case complexity of the whole SVM procedure is $O(|\mathcal{D}|^3 \cdot n)$; (V) *Reset learning*: In the worst case, NARX model fitting needs to be performed on each segment with length $k + 1$, and thus the complexity is $O(|\mathcal{D}| \cdot T_N(k))$.

As a consequence, the total *worst-case* complexity of DAINARX is $O(|\mathcal{D}|^3 \cdot n + T_N(|\mathcal{D}|) \cdot |\mathcal{D}|)$, i.e., *polynomial* in the size of the learning data $\mathcal{D}$ and the size of the target system d. In practice, however, the complexity is much lower (bounded by $O(T_N(|\mathcal{D}|) \cdot |Seg_{\mathcal{D}}| + |\mathcal{D}| \cdot T_N(w))$).

[10] For the minimally mergeable criterion, the complexity has to be multiplied by k to take into account the enumeration.