Derivative-Agnostic Inference of Nonlinear Hybrid Systems

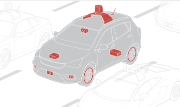
Mingshuai Chen

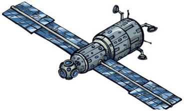
With Hengzhi Yu, Bohan Ma, Huangying Dong, Jie An, Bin Gu, Naijun Zhan, Jianwei Yin

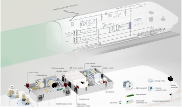
PIFI Day, Hangzhou · November 2025

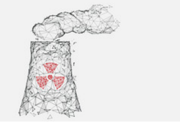
Cyber-Physical Systems (CPS)

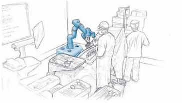
An open, interconnected form of embedded systems that integrates capabilities of *computing*, *communication*, and *control* :

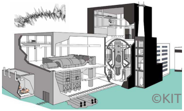

©TheOneBrief
automobiles


©clipartlogo
spacecrafts

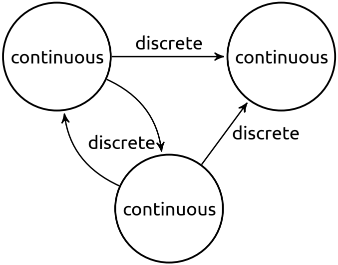

©Sécheron
high-speed rail


©VectorStock
nuclear reactors

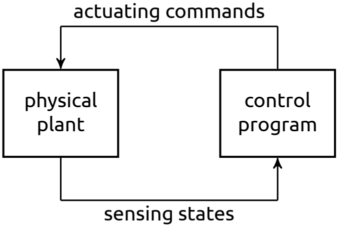

©Scoop.it
robot surgeon


©KIT
robust control

Hybrid Systems (HS) – A Formal Model of CPS

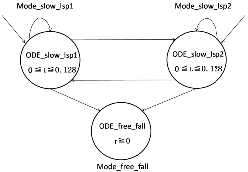


Macro : switching modes



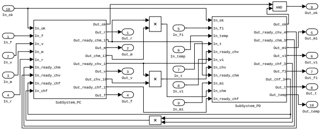
Micro : closed-loop feedback

Model-Based Design of CPS



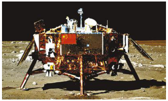
formal HS model

⇒



Simulink model

⇒



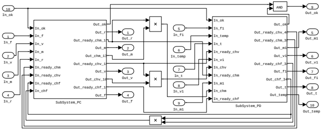
real system

Model-Based Design of CPS

?

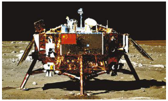
formal HS model

⇒



Simulink model

⇒



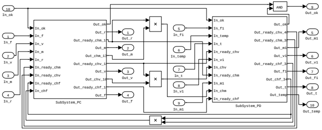
real system

Model-Based Design of CPS

?

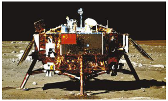
formal HS model

⇒



Simulink model

⇒

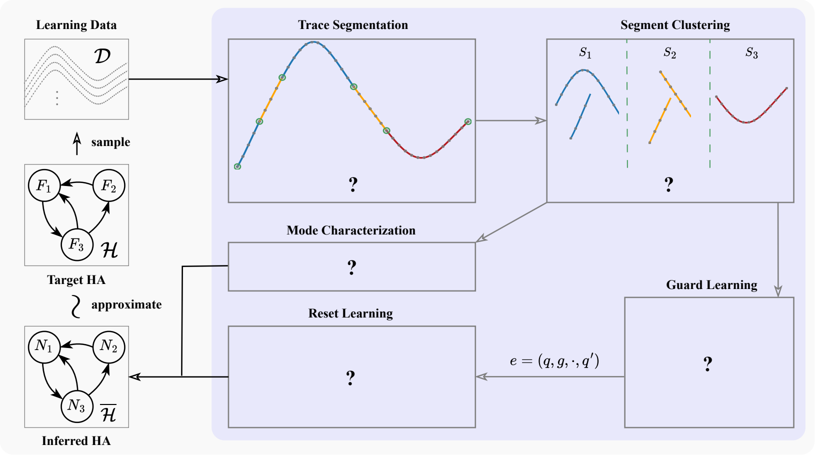


real system

HS identification through external observations

HS Identification – A Typical Pipeline

HS Identification – A Typical Pipeline



The Inference Landscape

Method	System Scope	Dynamics Type	High-Order	Resets	Inputs
Jin et al. [1]	Switched Systems	Polynomial ODEs	✗	✗	✗
Dayekh et al. [2]	Switched Systems	Polynomial ODEs	✗	✗	✗
POSEHAD [3]	Hybrid Automata	Polynomial ODEs	✗	✗	✓
LearnHA [4]	Hybrid Automata	Polynomial ODEs	✗	✓	✓
HySynth [5]	Hybrid Automata	Linear ODEs	✗	✗	✗
HAutLearn [6]	Hybrid Automata	Linear ODEs	✓	✗	✓
FaMoS [7]	Hybrid Automata	Linear ARX	✓	✗	✓
Madary et al. [8]	Switched Systems	Nonlinear ARX	✓	✗	✓

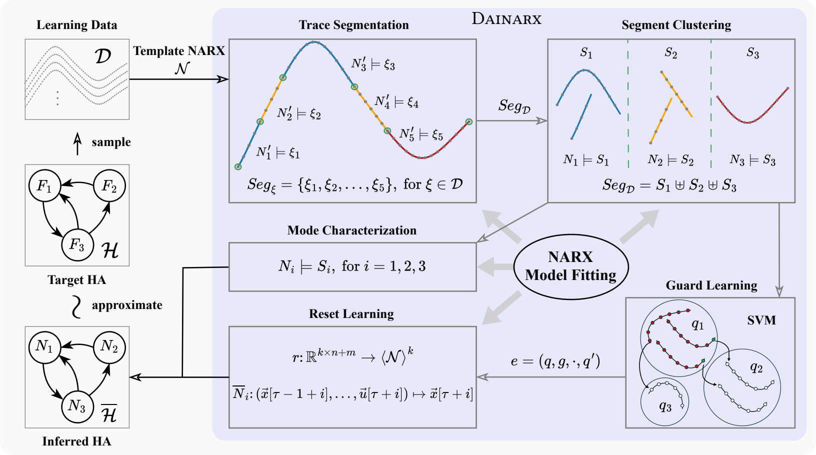
[1] X. Jin et al. *Inferring switched nonlinear dynamical systems*. Formal Aspects Comput. '21
[2] H. Dayekh, N. Basset, T. Dang. *Active learning of switched nonlinear dynamical systems*. CDC '24
[3] I. Saberi, F. Faghih, F. S. Bavil. *A passive online technique for learning hybrid automata from input/output traces*. ACM TECS '24
[4] A. Gurung, M. Waga, K. Suenaga. *Learning nonlinear hybrid automata from input-output time-series data*. ATVA '23
[5] M. G. Soto, T. A. Henzinger, C. Schilling. *Synthesis of hybrid automata with affine dynamics from time-series data*. HSCC '21
[6] X. Yang et al. *A framework for identification and validation of affine hybrid automata from input-output traces*. ACM TCPS '22
[7] S. Plambeck et al. *FaMoS – Fast model learning for hybrid cyber-physical systems using decision trees*. HSCC '24
[8] A. Madary et al. *Hierarchical identification of nonlinear hybrid systems in a Bayesian framework*. Inf. Comput. '22

The Inference Landscape

Method	System Scope	Dynamics Type	High-Order	Resets	Inputs
Jin et al. [1]	Switched Systems	Polynomial ODEs	✗	✗	✗
Dayekh et al. [2]	Switched Systems	Polynomial ODEs	✗	✗	✗
POSEHAD [3]	Hybrid Automata	Polynomial ODEs	✗	✗	✓
LearnHA [4]	Hybrid Automata	Polynomial ODEs	✗	✓	✓
HySynth [5]	Hybrid Automata	Linear ODEs	✗	✗	✗
HAutLearn [6]	Hybrid Automata	Linear ODEs	✓	✗	✓
FaMoS [7]	Hybrid Automata	Linear ARX	✓	✗	✓
Madary et al. [8]	Switched Systems	Nonlinear ARX	✓	✗	✓
Dainarx (ours)	Hybrid Automata	Nonlinear ARX	✓	✓	✓

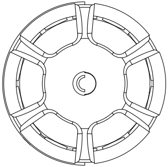
[1] X. Jin et al. *Inferring switched nonlinear dynamical systems*. Formal Aspects Comput. '21
[2] H. Dayekh, N. Basset, T. Dang. *Active learning of switched nonlinear dynamical systems*. CDC '24
[3] I. Saberi, F. Faghih, F. S. Bavil. *A passive online technique for learning hybrid automata from input/output traces*. ACM TECS '24
[4] A. Gurung, M. Waga, K. Suenaga. *Learning nonlinear hybrid automata from input-output time-series data*. ATVA '23
[5] M. G. Soto, T. A. Henzinger, C. Schilling. *Synthesis of hybrid automata with affine dynamics from time-series data*. HSCC '21
[6] X. Yang et al. *A framework for identification and validation of affine hybrid automata from input-output traces*. ACM TCPS '22
[7] S. Plambeck et al. *FaMoS – Fast model learning for hybrid cyber-physical systems using decision trees*. HSCC '24
[8] A. Madary et al. *Hierarchical identification of nonlinear hybrid systems in a Bayesian framework*. Inf. Comput. '22

The Dainarx Framework

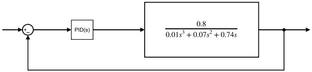


An Illustrating Example

A PID-controlled permanent-magnet synchronous motor

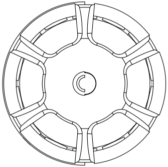


$$G(s) = \frac{\theta(s)}{V(s)} = \frac{K_t}{s[(Ls + R)(Js + B) + K_t K_e]}$$

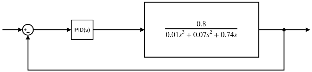


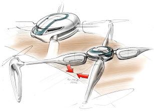
An Illustrating Example

A PID-controlled permanent-magnet synchronous motor



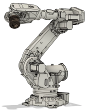
$$G(s) = \frac{\theta(s)}{V(s)} = \frac{K_t}{s[(Ls + R)(Js + B) + K_t K_e]}$$





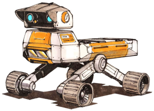
©Coroflot

UAV



©autodesk

mechanical arms

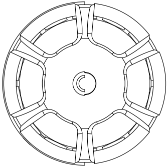


©ArtStation

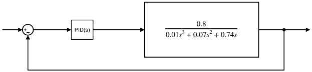
robotics

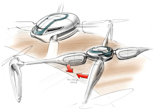
An Illustrating Example

A PID-controlled permanent-magnet synchronous motor



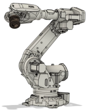
$$G(s) = \frac{\theta(s)}{V(s)} = \frac{K_t}{s[(Ls + R)(Js + B) + K_t K_e]}$$





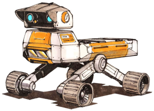
©Coroflot

UAV



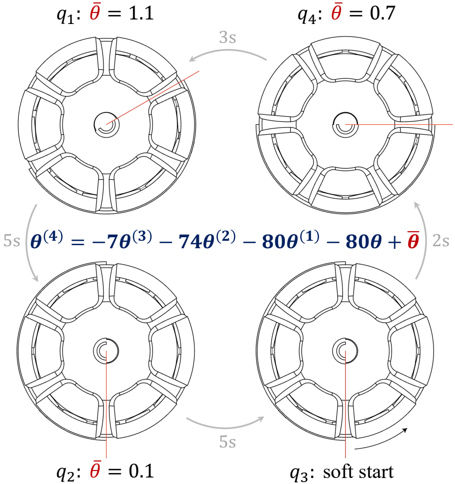
©autodesk

mechanical arms



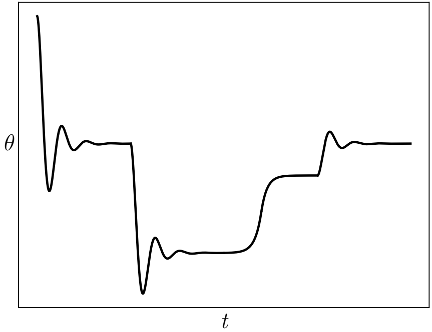
©ArtStation

robotics

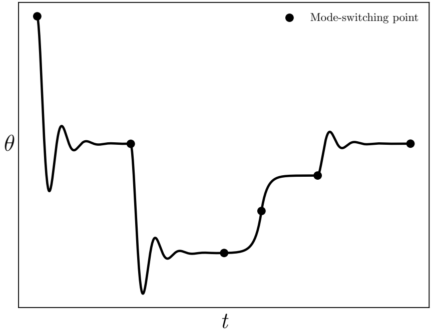


$$\boldsymbol{\theta}^{(4)} = -7\boldsymbol{\theta}^{(3)} - 74\boldsymbol{\theta}^{(2)} - 80\boldsymbol{\theta}^{(1)} - 80\boldsymbol{\theta} + \bar{\boldsymbol{\theta}}$$

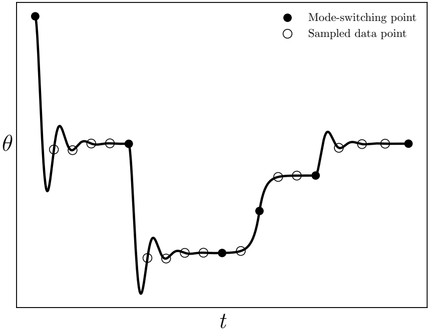
An Illustrating Example



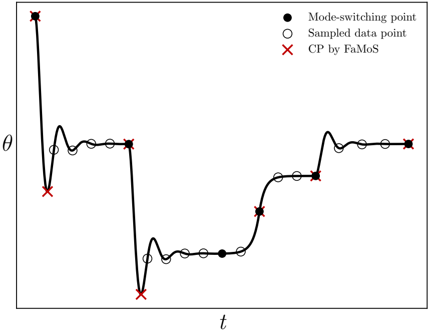
An Illustrating Example



An Illustrating Example



An Illustrating Example

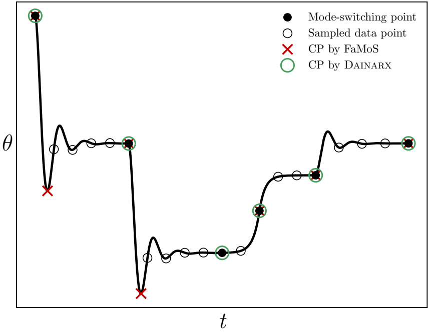


Derivative-based segmentation :

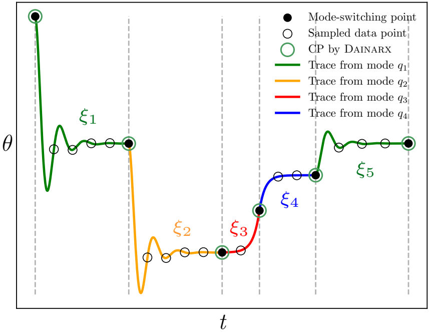
"derivative change" $> \gamma$

genuine CPs may be missed if γ is too large;
spurious CPs may be found if γ is too small

An Illustrating Example



An Illustrating Example

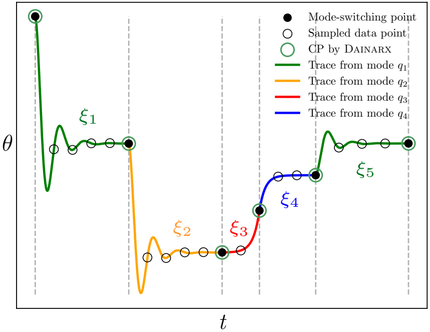


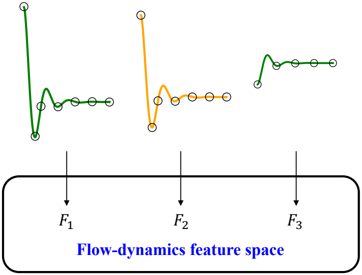
Trace similarity-based clustering :

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

similar traces $\not\Rightarrow$ same mode dynamics ;
neither vice versa

An Illustrating Example



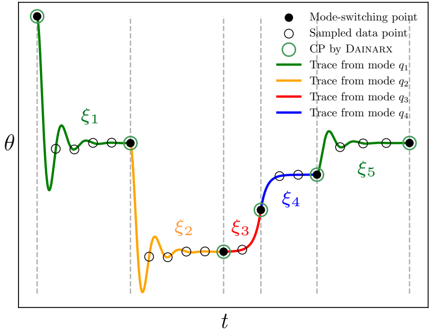


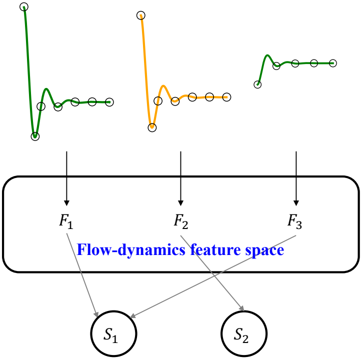
Trace similarity-based clustering :

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

similar traces $\not\Rightarrow$ same mode dynamics;
neither vice versa

An Illustrating Example





Trace similarity-based clustering :

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

similar traces $\not\Rightarrow$ same mode dynamics; neither vice versa

NARX Models

Definition (Nonlinear Autoregressive Exogenous (NARX) Model)

A *NARX model* is a k-th order *difference equation* :

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

NARX Models

Definition (Nonlinear Autoregressive Exogenous (NARX) Model)

A *NARX model* is a k-th order *difference equation*:

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

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

NARX Models

Definition (Nonlinear Autoregressive Exogenous (NARX) Model)

A *NARX model* is a k-th order *difference equation*:

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

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

$$\mathcal{A} \;=\; \left[\vec{a}_1 \quad \vec{a}_2 \quad \ldots \quad \vec{a}_\alpha \quad B_1 \quad B_2 \quad \ldots \quad B_k \quad B_{k+1} \quad \vec{c}\right]$$

NARX Models

Definition (Nonlinear Autoregressive Exogenous (NARX) Model)

A *NARX model* is a k-th order *difference equation* :

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

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

$$\mathcal{A} \;=\; \left[\vec{a}_1 \quad \vec{a}_2 \quad \ldots \quad \vec{a}_\alpha \quad B_1 \quad B_2 \quad \ldots \quad B_k \quad B_{k+1} \quad \vec{c}\,\right]$$

$\mathcal{N} = (\mathcal{A}, \{f_i\})$: a *template NARX model* with unknown parameters $\mathcal{A}$;
$\mathcal{N}[\Lambda]$: the *instance* of $\mathcal{N}$ under $\Lambda \in \mathcal{A}$; $\langle\mathcal{N}\rangle$: the set of all instances of $\mathcal{N}$.

NARX Model Fitting – via Linear Least Squares Method

Given : A NARX template $\mathcal{N} = (\mathcal{A}, \{f_i\})$ and a set $S = \{\xi_j\}$ of discrete-time traces.

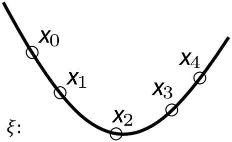
Goal : Find $\Lambda \in \mathcal{A}$ such that $\mathcal{N}[\Lambda]$ *fits* S, denoted by $\mathcal{N}[\Lambda] \models S$.

NARX Model Fitting – via Linear Least Squares Method

Given : A NARX template $\mathcal{N} = (\mathcal{A}, \{f_i\})$ and a set $S = \{\xi_j\}$ of discrete-time traces.

Goal : Find $\Lambda \in \mathcal{A}$ such that $\mathcal{N}[\Lambda]$ *fits* S, denoted by $\mathcal{N}[\Lambda] \models S$.

$$x[\tau] = a_1 \cdot x^2[\tau - 1] + a_2 \cdot x^3[\tau - 2] + B_1 \cdot x[\tau - 1] + B_2 \cdot x[\tau - 2] + c \cdot 1$$

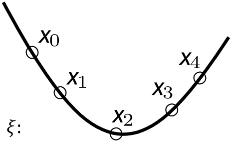


NARX Model Fitting – via Linear Least Squares Method

Given : A NARX template $\mathcal{N} = (\mathcal{A}, \{f_i\})$ and a set $S = \{\xi_j\}$ of discrete-time traces.

Goal : Find $\Lambda \in \mathcal{A}$ such that $\mathcal{N}[\Lambda]$ *fits* S, denoted by $\mathcal{N}[\Lambda] \models S$.

$$x[\tau] \;=\; a_1 \cdot x^2[\tau-1] + a_2 \cdot x^3[\tau-2] + B_1 \cdot x[\tau-1] + B_2 \cdot x[\tau-2] + c \cdot 1$$



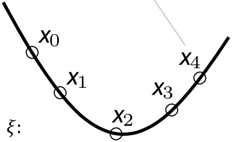
$$O = \begin{bmatrix} x_4 & x_3 & x_2 \end{bmatrix}; \qquad D = \begin{bmatrix} x_3^2 & x_2^3 & x_3 & x_2 & 1 \\ x_2^2 & x_1^3 & x_2 & x_1 & 1 \\ x_1^2 & x_0^3 & x_1 & x_0 & 1 \end{bmatrix}^{\mathsf{T}}$$

NARX Model Fitting – via Linear Least Squares Method

Given : A NARX template $\mathcal{N} = (\mathcal{A}, \{f_i\})$ and a set $S = \{\xi_j\}$ of discrete-time traces.

Goal : Find $\Lambda \in \mathcal{A}$ such that $\mathcal{N}[\Lambda]$ *fits* S, denoted by $\mathcal{N}[\Lambda] \models S$.

$$x[\tau] \;=\; a_1 \cdot x^2[\tau-1] + a_2 \cdot x^3[\tau-2] + B_1 \cdot x[\tau-1] + B_2 \cdot x[\tau-2] + c \cdot 1$$



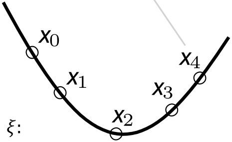
$$O \;=\; \begin{bmatrix} x_4 & x_3 & x_2 \end{bmatrix} \;; \qquad D \;=\; \begin{bmatrix} x_3^2 & x_2^3 & x_3 & x_2 & 1 \\ x_2^2 & x_1^3 & x_2 & x_1 & 1 \\ x_1^2 & x_0^3 & x_1 & x_0 & 1 \end{bmatrix}^{\mathsf{T}}$$

NARX Model Fitting – via Linear Least Squares Method

Given : A NARX template $\mathcal{N} = (\mathcal{A}, \{f_i\})$ and a set $S = \{\xi_j\}$ of discrete-time traces.

Goal : Find $\Lambda \in \mathcal{A}$ such that $\mathcal{N}[\Lambda]$ *fits* S, denoted by $\mathcal{N}[\Lambda] \models S$.

$$x[\tau] \;=\; a_1 \cdot x^2[\tau-1] + a_2 \cdot x^3[\tau-2] + B_1 \cdot x[\tau-1] + B_2 \cdot x[\tau-2] + c \cdot 1$$



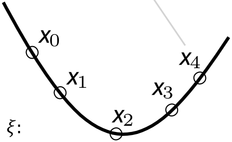
$$O \;=\; \begin{bmatrix} x_4 & x_3 & x_2 \end{bmatrix} \;; \qquad D \;=\; \begin{bmatrix} x_3^2 & x_2^3 & x_3 & x_2 & 1 \\ x_2^2 & x_1^3 & x_2 & x_1 & 1 \\ x_1^2 & x_0^3 & x_1 & x_0 & 1 \end{bmatrix}^{\mathsf{T}}$$

NARX Model Fitting – via Linear Least Squares Method

Given : A NARX template $\mathcal{N} = (\mathcal{A}, \{f_i\})$ and a set $S = \{\xi_j\}$ of discrete-time traces.

Goal : Find $\Lambda \in \mathcal{A}$ such that $\mathcal{N}[\Lambda]$ *fits* S, denoted by $\mathcal{N}[\Lambda] \models S$.

$$x[\tau] = a_1 \cdot x^2[\tau - 1] + a_2 \cdot x^3[\tau - 2] + B_1 \cdot x[\tau - 1] + B_2 \cdot x[\tau - 2] + c \cdot 1$$



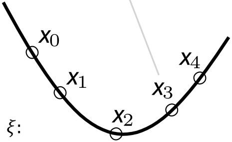
$$O = \begin{bmatrix} x_4 & x_3 & x_2 \end{bmatrix}; \qquad D = \begin{bmatrix} x_3^2 & x_2^3 & x_3 & x_2 & 1 \\ x_2^2 & x_1^3 & x_2 & x_1 & 1 \\ x_1^2 & x_0^3 & x_1 & x_0 & 1 \end{bmatrix}^{\mathsf{T}}$$

NARX Model Fitting – via Linear Least Squares Method

Given : A NARX template $\mathcal{N} = (\mathcal{A}, \{f_i\})$ and a set $S = \{\xi_j\}$ of discrete-time traces.

Goal : Find $\Lambda \in \mathcal{A}$ such that $\mathcal{N}[\Lambda]$ *fits* S, denoted by $\mathcal{N}[\Lambda] \models S$.

$$x[\tau] \; = \; a_1 \cdot x^2[\tau-1] + a_2 \cdot x^3[\tau-2] + B_1 \cdot x[\tau-1] + B_2 \cdot x[\tau-2] + c \cdot 1$$



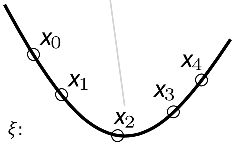
$$O \; = \; \begin{bmatrix} x_4 & x_3 & x_2 \end{bmatrix} \; ; \qquad D \; = \; \begin{bmatrix} x_3^2 & x_2^3 & x_3 & x_2 & 1 \\ x_2^2 & x_1^3 & x_2 & x_1 & 1 \\ x_1^2 & x_0^3 & x_1 & x_0 & 1 \end{bmatrix}^{\mathsf{T}}$$

NARX Model Fitting – via Linear Least Squares Method

Given : A NARX template $\mathcal{N} = (\mathcal{A}, \{f_i\})$ and a set $S = \{\xi_j\}$ of discrete-time traces.

Goal : Find $\Lambda \in \mathcal{A}$ such that $\mathcal{N}[\Lambda]$ *fits* S, denoted by $\mathcal{N}[\Lambda] \models S$.

$$x[\tau] = a_1 \cdot x^2[\tau-1] + a_2 \cdot x^3[\tau-2] + B_1 \cdot x[\tau-1] + B_2 \cdot x[\tau-2] + c \cdot 1$$



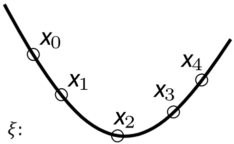
$$O = \begin{bmatrix} x_4 & x_3 & x_2 \end{bmatrix}; \qquad D = \begin{bmatrix} x_3^2 & x_2^3 & x_3 & x_2 & 1 \\ x_2^2 & x_1^3 & x_2 & x_1 & 1 \\ x_1^2 & x_0^3 & x_1 & x_0 & 1 \end{bmatrix}^{\mathsf{T}}$$

NARX Model Fitting – via Linear Least Squares Method

Given : A NARX template $\mathcal{N} = (\mathcal{A}, \{f_i\})$ and a set $S = \{\xi_j\}$ of discrete-time traces.

Goal : Find $\Lambda \in \mathcal{A}$ such that $\mathcal{N}[\Lambda]$ *fits* S, denoted by $\mathcal{N}[\Lambda] \models S$.

$$x[\tau] \; = \; a_1 \cdot x^2[\tau-1] + a_2 \cdot x^3[\tau-2] + B_1 \cdot x[\tau-1] + B_2 \cdot x[\tau-2] + c \cdot 1$$



$$O \; = \; \begin{bmatrix} x_4 & x_3 & x_2 \end{bmatrix} \; ; \qquad D \; = \; \begin{bmatrix} x_3^2 & x_2^3 & x_3 & x_2 & 1 \\ x_2^2 & x_1^3 & x_2 & x_1 & 1 \\ x_1^2 & x_0^3 & x_1 & x_0 & 1 \end{bmatrix}^{\mathsf{T}}$$

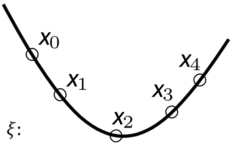
$$\min_{\Lambda} \; \| O - \Lambda \cdot D \|_2$$

NARX Model Fitting – via Linear Least Squares Method

Given : A NARX template $\mathcal{N} = (\mathcal{A}, \{f_i\})$ and a set $S = \{\xi_j\}$ of discrete-time traces.

Goal : Find $\Lambda \in \mathcal{A}$ such that $\mathcal{N}[\Lambda]$ *fits* S, denoted by $\mathcal{N}[\Lambda] \models S$.

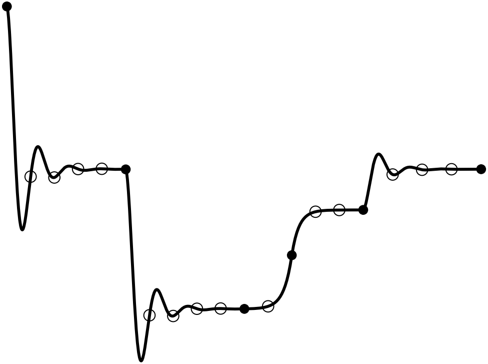
$$x[\tau] \;=\; a_1 \cdot x^2[\tau-1] + a_2 \cdot x^3[\tau-2] + B_1 \cdot x[\tau-1] + B_2 \cdot x[\tau-2] + c \cdot 1$$



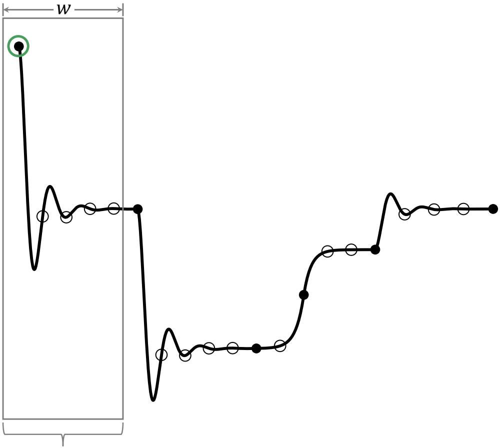
$$O \;=\; \begin{bmatrix} x_4 & x_3 & x_2 \end{bmatrix}\;; \qquad D \;=\; \begin{bmatrix} x_3^2 & x_2^3 & x_3 & x_2 & 1 \\ x_2^2 & x_1^3 & x_2 & x_1 & 1 \\ x_1^2 & x_0^3 & x_1 & x_0 & 1 \end{bmatrix}^{\mathsf{T}}$$

$$\min_{\Lambda} \; \| O - \Lambda \cdot D \|_2 \;=\; 0 \quad \Longleftrightarrow \quad \mathcal{N}[\Lambda] \;\models\; \xi$$

Trace Segmentation

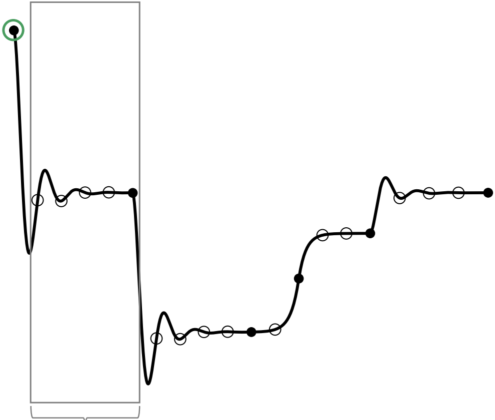


Trace Segmentation



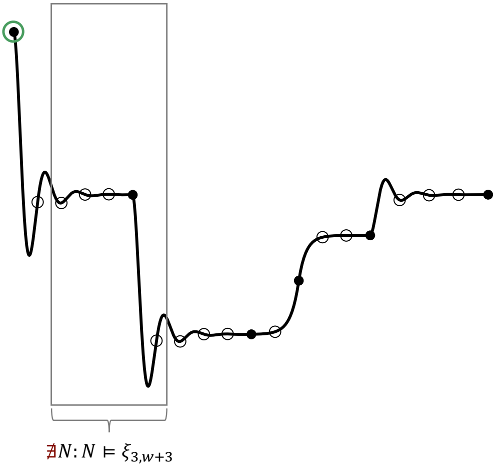
$\exists N : N \models \xi_{1,w+1}$

Trace Segmentation

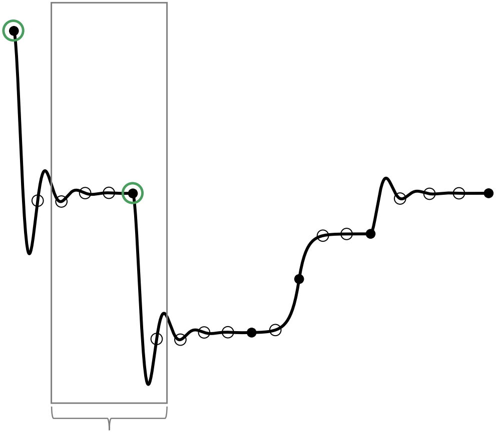


$\exists N : N \vDash \xi_{2,w+2}$

Trace Segmentation

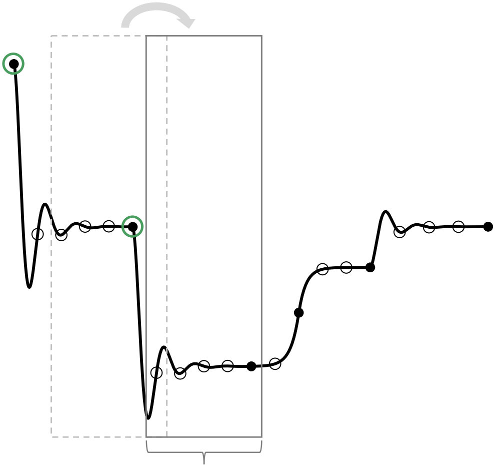


Trace Segmentation



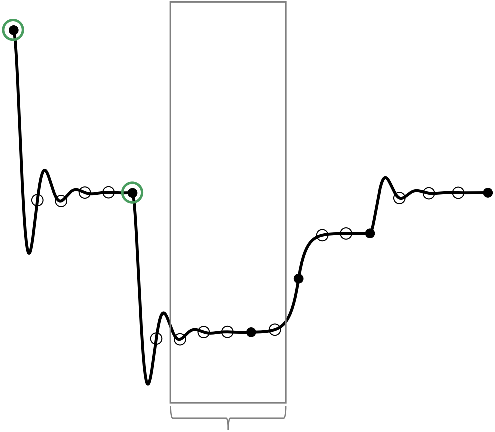
$\nexists N \colon N \vDash \xi_{3,w+3}$

Trace Segmentation



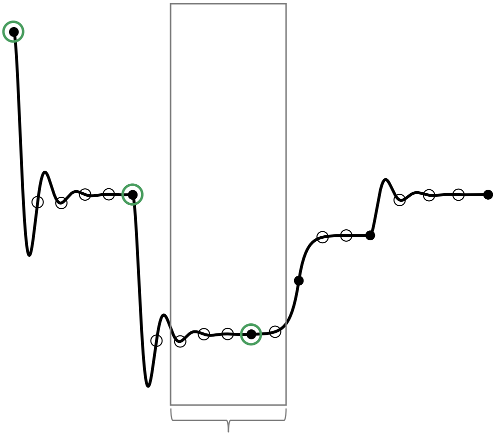
$\exists N : N \models \xi_{7,w+7}$

Trace Segmentation



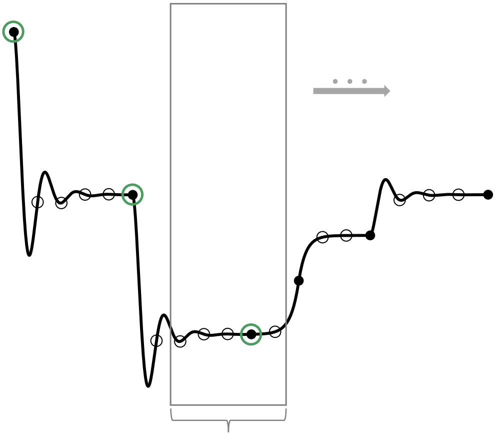
$\nexists N : N \vDash \xi_{8,w+8}$

Trace Segmentation



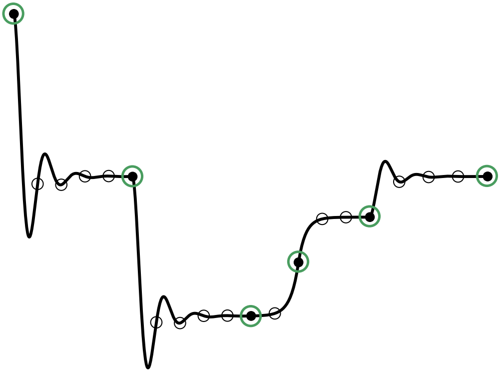
$\nexists N : N \models \xi_{8,w+8}$

Trace Segmentation

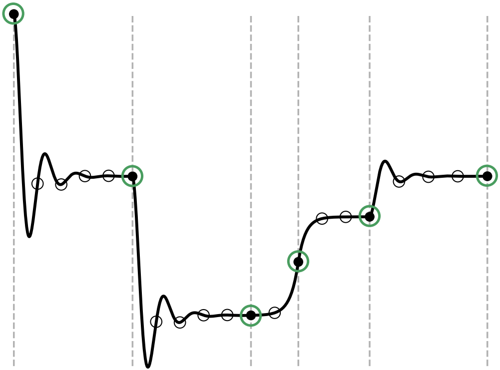


$\nexists N : N \models \xi_{8,w+8}$

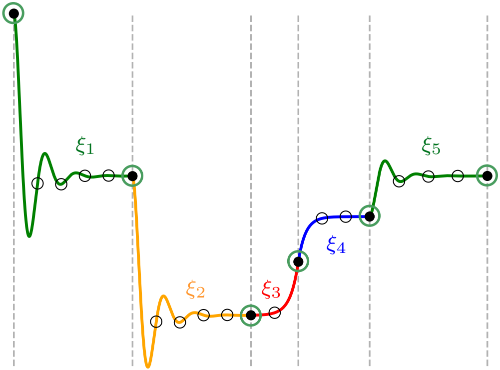
Trace Segmentation



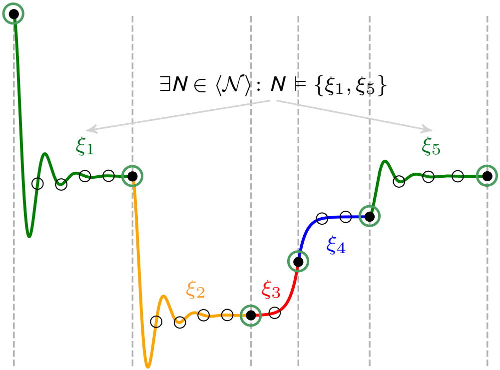
Trace Segmentation



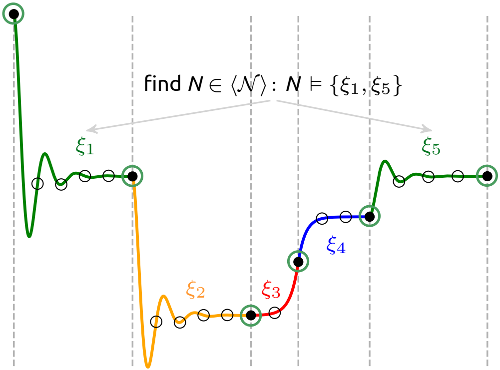
Segment Clustering



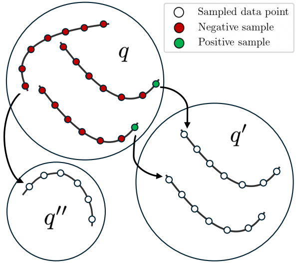
Segment Clustering



Mode Characterization



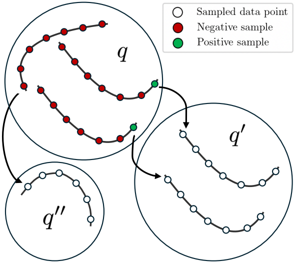
Guard Learning

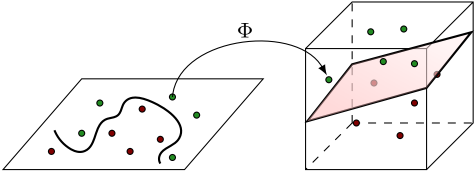


$$(q, q')^+ = \bigcup_{j=1}^{M} \{\xi_j(\tau) \mid \mathcal{M}_j(\tau) = q \text{ and } \mathcal{M}_j(\tau+1) = q'\} ,$$

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

Guard Learning



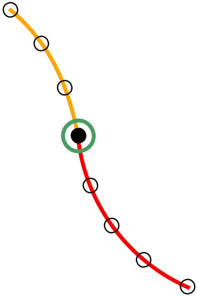


SVM space transformation and kernel tricks

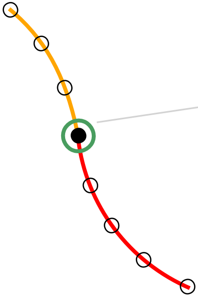
$$(q, q')^+ = \bigcup_{j=1}^{M} \{\xi_j(\tau) \mid \mathcal{M}_j(\tau) = q \text{ and } \mathcal{M}_j(\tau+1) = q'\} ,$$

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

Reset Learning



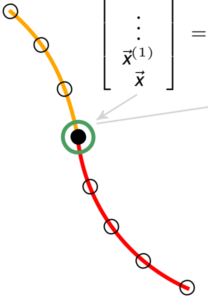
Reset Learning



$$\begin{bmatrix} \vec{x}^{(k-1)} \\ \vdots \\ \vec{x}^{(1)} \\ \vec{x} \end{bmatrix}$$

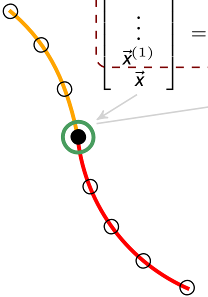
Reset Learning

$$\begin{bmatrix} \vec{x}^{(k-1)} \\ \vdots \\ \vec{x}^{(1)} \\ \vec{x} \end{bmatrix} = \mathcal{T} \cdot \begin{bmatrix} \vec{x}^{(k-1)} \\ \vdots \\ \vec{x}^{(1)} \\ \vec{x} \end{bmatrix}$$

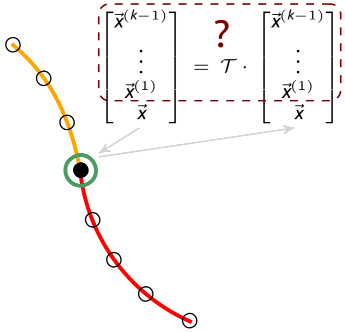


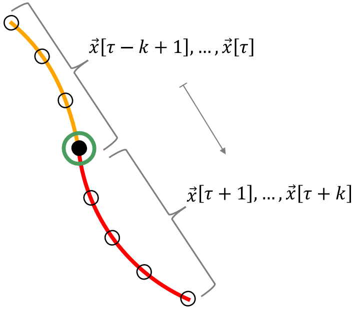
Reset Learning

$$\begin{bmatrix} \vec{x}^{(k-1)} \\ \vdots \\ \vec{x}^{(1)} \\ \vec{x} \end{bmatrix} \overset{?}{=} \mathcal{T} \cdot \begin{bmatrix} \vec{x}^{(k-1)} \\ \vdots \\ \vec{x}^{(1)} \\ \vec{x} \end{bmatrix}$$

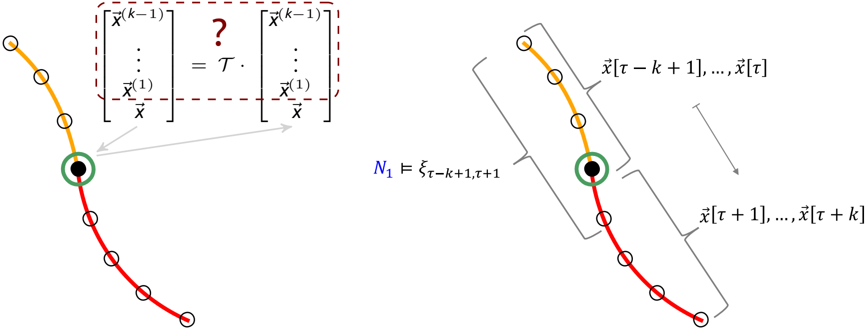


Reset Learning



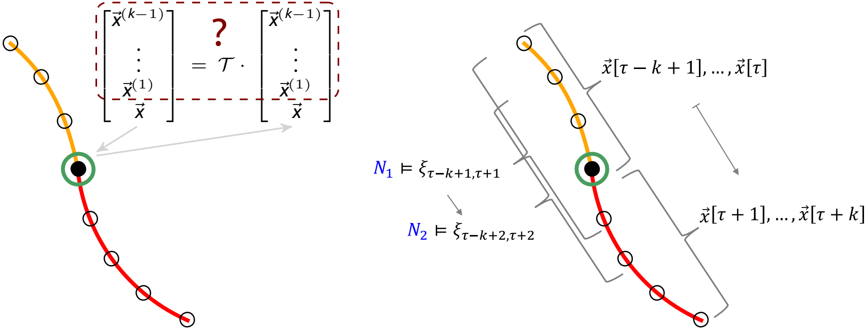


Reset Learning



$$N_1 : \quad \left(\vec{x}[\tau - k + 1], \ldots, \vec{x}[\tau], \vec{u}[\tau + 1]\right) \mapsto \vec{x}[\tau + 1]$$

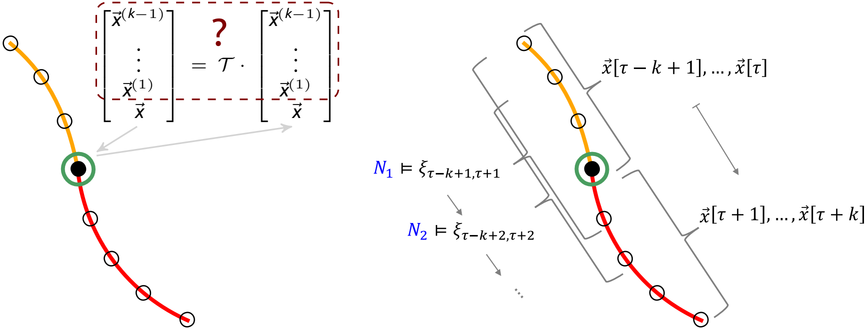
Reset Learning



$$N_1: \quad \left(\vec{x}[\tau-k+1], \ldots, \vec{x}[\tau], \vec{u}[\tau+1]\right) \mapsto \vec{x}[\tau+1]$$

$$N_2: \quad \left(\vec{x}[\tau-k+2], \ldots, \vec{x}[\tau+1], \vec{u}[\tau+2]\right) \mapsto \vec{x}[\tau+2]$$

Reset Learning



$$N_1: \quad (\vec{x}[\tau - k + 1], \ldots, \vec{x}[\tau], \vec{u}[\tau + 1]) \mapsto \vec{x}[\tau + 1]$$

$$N_2: \quad (\vec{x}[\tau - k + 2], \ldots, \vec{x}[\tau + 1], \vec{u}[\tau + 2]) \mapsto \vec{x}[\tau + 2]$$

$$\vdots$$

$$N_k: \quad (\vec{x}[\tau], \ldots, \vec{x}[\tau + k - 1], \vec{u}[\tau + k]) \mapsto \vec{x}[\tau + k]$$

Tool Support

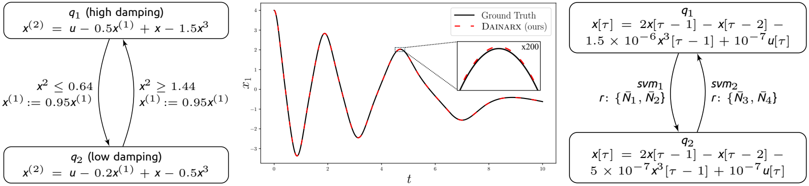
Dainarx : Derivative-Agnostic Inference via NARX model fitting

https://github.com/FICTION-ZJU/Dainarx

Tool Support

Dainarx : Derivative-Agnostic Inference via NARX model fitting

https://github.com/FICTION-ZJU/Dainarx



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

Figure – Inference of the Duffing oscillator via Dainarx (w. high-order nonlinear $\mathcal{F}$, non-polynomial U, and R).

Experiments – Inferring Linear HS

Benchmark Details				HDT_c (seconds)			$Diff_{max}$			$Diff_{avg}$		
Name	$\lvert Q\rvert$	$\lvert X\rvert$	k	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx
buck_converter	3	2	1	0.0001	0.0004	**0.0000**	0.1674	0.2065	**0.0000**	0.0046	0.0142	**0.0000**
complex_tank	8	3	1	3.6350	1.6350	**0.0950**	0.3483	0.2411	**0.0451**	0.0655	0.0151	**0.0028**
multi_room_heating	4	3	1	2.9750	0.0600	**0.0350**	0.2531	0.0094	**0.0078**	0.0590	0.0013	**0.0008**
simple_heating_syst	2	1	1	0.0400	0.4400	**0.0200**	0.0202	0.2126	**0.0102**	0.0037	0.0423	**0.0007**
three_state_HA	3	1	2	–	0.5700	**0.0000**	–	0.5388	**0.0000**	–	0.0178	**0.0000**
two_state_HA	2	1	2	–	0.3400	**0.0000**	–	0.1284	**0.0000**	–	0.0132	**0.0000**
variable_heating_syst	3	2	1	0.1100	0.0700	**0.0300**	0.0581	0.0272	**0.0200**	0.0082	0.0012	**0.0005**
cell	4	1	1	**0.0100**	29.2800	**0.0100**	**0.0176**	1.6144	**0.0176**	**0.0001**	0.1494	0.0002
oci	2	2	1	0.0500	–	**0.0000**	0.2002	–	**0.0000**	0.0259	–	**0.0000**
tanks (w. U)	4	2	1	13.9100	–	**0.0100**	1.1077	–	**0.0177**	0.2589	–	**0.0007**
ball (w. R)	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	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 (w. U, R)	4	2	2	–	–	**0.0100**	–	–	**0.0080**	–	–	**0.0004**

Experiments – Inferring Linear HS

Benchmark Details				HDT_c (seconds)			$Diff_{max}$			$Diff_{avg}$		
Name	$\|Q\|$	$\|X\|$	k	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx
buck_converter	3	2	1	0.0001	0.0004	**0.0000**	0.1674	0.2065	**0.0000**	0.0046	0.0142	**0.0000**
complex_tank	8	3	1	3.6350	1.6350	**0.0950**	0.3483	0.2411	**0.0451**	0.0655	0.0151	**0.0028**
multi_room_heating	4	3	1	2.9750	0.0600	**0.0350**	0.2531	0.0094	**0.0078**	0.0590	0.0013	**0.0008**
simple_heating_syst	2	1	1	0.0400	0.4400	**0.0200**	0.0202	0.2126	**0.0102**	0.0037	0.0423	**0.0007**
three_state_HA	3	1	2	–	0.5700	**0.0000**	–	0.5388	**0.0000**	–	0.0178	**0.0000**
two_state_HA	2	1	2	–	0.3400	**0.0000**	–	0.1284	**0.0000**	–	0.0132	**0.0000**
variable_heating_syst	3	2	1	0.1100	0.0700	**0.0300**	0.0581	0.0272	**0.0200**	0.0082	0.0012	**0.0005**
cell	4	1	1	**0.0100**	29.2800	**0.0100**	**0.0176**	1.6144	**0.0176**	**0.0001**	0.1494	0.0002
oci	2	2	1	0.0500	–	**0.0000**	0.2002	–	**0.0000**	0.0259	–	**0.0000**
tanks (w. U)	4	2	1	13.9100	–	**0.0100**	1.1077	–	**0.0177**	0.2589	–	**0.0007**
ball (w. R)	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	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 (w. U, R)	4	2	2	–	–	**0.0100**	–	–	**0.0080**	–	–	**0.0004**

⇒ **Applicability :** Dainarx suffices to infer the HA for all the 18 benchmarks;

Experiments – Inferring Linear HS

Benchmark Details				HDT_c (seconds)			$Diff_{max}$			$Diff_{avg}$		
Name	$\lvert Q \rvert$	$\lvert X \rvert$	k	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx
buck_converter	3	2	1	0.0001	0.0004	**0.0000**	0.1674	0.2065	**0.0000**	0.0046	0.0142	**0.0000**
complex_tank	8	3	1	3.6350	1.6350	**0.0950**	0.3483	0.2411	**0.0451**	0.0655	0.0151	**0.0028**
multi_room_heating	4	3	1	2.9750	0.0600	**0.0350**	0.2531	0.0094	**0.0078**	0.0590	0.0013	**0.0008**
simple_heating_syst	2	1	1	0.0400	0.4400	**0.0200**	0.0202	0.2126	**0.0102**	0.0037	0.0423	**0.0007**
three_state_HA	3	1	2	–	0.5700	**0.0000**	–	0.5388	**0.0000**	–	0.0178	**0.0000**
two_state_HA	2	1	2	–	0.3400	**0.0000**	–	0.1284	**0.0000**	–	0.0132	**0.0000**
variable_heating_syst	3	2	1	0.1100	0.0700	**0.0300**	0.0581	0.0272	**0.0200**	0.0082	0.0012	**0.0005**
cell	4	1	1	**0.0100**	29.2800	**0.0100**	**0.0176**	1.6144	**0.0176**	**0.0001**	0.1494	0.0002
oci	2	2	1	0.0500	–	**0.0000**	0.2002	–	**0.0000**	0.0259	–	**0.0000**
tanks (w. U)	4	2	1	13.9100	–	**0.0100**	1.1077	–	**0.0177**	0.2589	–	**0.0007**
ball (w. R)	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	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 (w. U, R)	4	2	2	–	–	**0.0100**	–	–	**0.0080**	–	–	**0.0004**

⇒ **Applicability :** Dainarx suffices to infer the HA for all the 18 benchmarks;

⇒ **Mode-switching accuracy :** Dainarx achieves (tied-)highest accuracy in detecting mode switching in all the 18 benchmarks;

Experiments – Inferring Linear HS

Benchmark Details				HDT_c (seconds)			$Diff_{max}$			$Diff_{avg}$		
Name	$\lvert Q\rvert$	$\lvert X\rvert$	k	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx
buck_converter	3	2	1	0.0001	0.0004	**0.0000**	0.1674	0.2065	**0.0000**	0.0046	0.0142	**0.0000**
complex_tank	8	3	1	3.6350	1.6350	**0.0950**	0.3483	0.2411	**0.0451**	0.0655	0.0151	**0.0028**
multi_room_heating	4	3	1	2.9750	0.0600	**0.0350**	0.2531	0.0094	**0.0078**	0.0590	0.0013	**0.0008**
simple_heating_syst	2	1	1	0.0400	0.4400	**0.0200**	0.0202	0.2126	**0.0102**	0.0037	0.0423	**0.0007**
three_state_HA	3	1	2	–	0.5700	**0.0000**	–	0.5388	**0.0000**	–	0.0178	**0.0000**
two_state_HA	2	1	2	–	0.3400	**0.0000**	–	0.1284	**0.0000**	–	0.0132	**0.0000**
variable_heating_syst	3	2	1	0.1100	0.0700	**0.0300**	0.0581	0.0272	**0.0200**	0.0082	0.0012	**0.0005**
cell	4	1	1	**0.0100**	29.2800	**0.0100**	**0.0176**	1.6144	**0.0176**	**0.0001**	0.1494	0.0002
oci	2	2	1	0.0500	–	**0.0000**	0.2002	–	**0.0000**	0.0259	–	**0.0000**
tanks (w. U)	4	2	1	13.9100	–	**0.0100**	1.1077	–	**0.0177**	0.2589	–	**0.0007**
ball (w. R)	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	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 (w. U, R)	4	2	2	–	–	**0.0100**	–	–	**0.0080**	–	–	**0.0004**

- ⇒ **Applicability :** Dainarx suffices to infer the HA for all the 18 benchmarks;
- ⇒ **Mode-switching accuracy :** Dainarx achieves (tied-)highest accuracy in detecting mode switching in all the 18 benchmarks;
- ⇒ **Trace fidelity :** Dainarx attains highest trace fidelity across 17/18 benchmarks.

Experiments – Inferring Linear HS

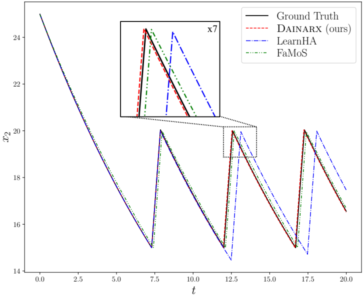


Figure – Trace fidelity for `complex_tank`.

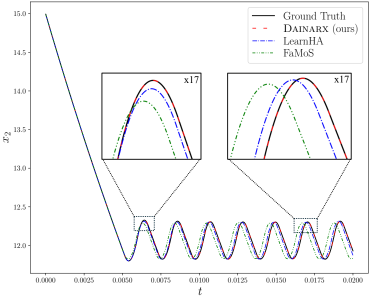


Figure – Trace fidelity for `buck_converter`.

Experiments – Inferring Nonlinear HS

Benchmark Details				Dainarx		
Name	$\|Q\|$	$\|X\|$	k	HDT_c	$Diff_{max}$	$Diff_{avg}$
lander	2	4	1	0.010	0.0024	0.0000
lotkaVolterra (w. nonlinear G)	2	2	1	0.020	0.0047	0.0006
simple_non_linear	2	1	1	0.006	0.0367	0.0020
simple_non_poly (w. R)	2	1	1	0.000	0.0000	0.0000
oscillator (w. trigonometric $\mathcal{F}, G$)	2	2	1	0.000	0.0000	0.0000
spacecraft (w. nonlinear G)	2	4	1	0.000	0.0000	0.0000
sys_bio	2	9	1	0.012	0.0960	0.0081
duffing (w. U, R, nonlinear G)	2	1	2	0.001	0.0003	0.0000

Experiments – Inferring Nonlinear HS

Benchmark Details				Dainarx		
Name	$\lvert Q\rvert$	$\lvert X\rvert$	k	HDT_c	$Diff_{max}$	$Diff_{avg}$
`lander`	2	4	1	0.010	0.0024	0.0000
`lotkaVolterra (w. nonlinear` G`)`	2	2	1	0.020	0.0047	0.0006
`simple_non_linear`	2	1	1	0.006	0.0367	0.0020
`simple_non_poly (w.` R`)`	2	1	1	0.000	0.0000	0.0000
`oscillator (w. trigonometric` $\mathcal{F}, G$`)`	2	2	1	0.000	0.0000	0.0000
`spacecraft (w. nonlinear` G`)`	2	4	1	0.000	0.0000	0.0000
`sys_bio`	2	9	1	0.012	0.0960	0.0081
`duffing (w.` U, R`, nonlinear` G`)`	2	1	2	0.001	0.0003	0.0000

⇒ Effectiveness in learning high-order nonlinear systems beyond polynomials.

Inference Time

Benchmark Details				Segmentation Time			Clustering Time			Guard-Learning Time			Total Time		
Name	$\lvert Q\rvert$	$\lvert X\rvert$	k	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx
buck_converter	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	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	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	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	3	1	2	–	**0.14**	0.55	–	1.30	**0.31**	–	0.01	**0.00**	–	1.45	**0.92**
two_state_HA	2	1	2	–	**0.20**	0.54	–	1.50	**0.22**	–	**0.01**	**0.01**	–	1.72	**0.80**
variable_heating_syst	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	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	2	2	1	2.14	–	**0.93**	5.83	–	**0.36**	0.48	–	**0.03**	8.46	–	**1.41**
tanks	4	2	1	2.21	–	**1.17**	16.98	–	**1.50**	1.56	–	**1.04**	20.79	–	**3.82**
ball	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	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	2	4	1			**1.38**			**1.03**			**0.00**			**2.58**
lotkaVolterra	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	2	2	1			**0.65**			**0.43**			**0.01**			**1.13**
spacecraft	2	4	1			**1.20**			**0.72**			**0.01**			**2.01**
sys_bio	2	9	1			**13.17**			**12.60**			**0.20**			**27.40**
duffing	2	1	2			**3.92**			**2.98**			**0.06**			**7.34**

Inference Time

Benchmark Details				Segmentation Time			Clustering Time			Guard-Learning Time			Total Time		
Name	$\lvert Q\rvert$	$\lvert X\rvert$	k	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx
buck_converter	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	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	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	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	3	1	2	–	**0.14**	0.55	–	1.30	**0.31**	–	0.01	**0.00**	–	1.45	**0.92**
two_state_HA	2	1	2	–	**0.20**	0.54	–	1.50	**0.22**	–	**0.01**	**0.01**	–	1.72	**0.80**
variable_heating_syst	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	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	2	2	1	2.14	–	**0.93**	5.83	–	**0.36**	0.48	–	**0.03**	8.46	–	**1.41**
tanks	4	2	1	2.21	–	**1.17**	16.98	–	**1.50**	1.56	–	**1.04**	20.79	–	**3.82**
ball	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	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	2	4	1	–	–	**1.38**	–	–	**1.03**	–	–	**0.00**	–	–	**2.58**
lotkaVolterra	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	2	2	1	–	–	**0.65**	–	–	**0.43**	–	–	**0.01**	–	–	**1.13**
spacecraft	2	4	1	–	–	**1.20**	–	–	**0.72**	–	–	**0.01**	–	–	**2.01**
sys_bio	2	9	1	–	–	**13.17**	–	–	**12.60**	–	–	**0.20**	–	–	**27.40**
duffing	2	1	2	–	–	**3.92**	–	–	**2.98**	–	–	**0.06**	–	–	**7.34**

⇒ Segment clustering dominates the time of trace similarity-based methods;

Inference Time

Benchmark Details				Segmentation Time			Clustering Time			Guard-Learning Time			Total Time		
Name	$\lvert Q\rvert$	$\lvert X\rvert$	k	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx
buck_converter	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	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	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	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	3	1	2	–	**0.14**	0.55	–	1.30	**0.31**	–	0.01	**0.00**	–	1.45	**0.92**
two_state_HA	2	1	2	–	**0.20**	0.54	–	1.50	**0.22**	–	**0.01**	**0.01**	–	1.72	**0.80**
variable_heating_syst	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	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	2	2	1	2.14	–	**0.93**	5.83	–	**0.36**	0.48	–	**0.03**	8.46	–	**1.41**
tanks	4	2	1	2.21	–	**1.17**	16.98	–	**1.50**	1.56	–	**1.04**	20.79	–	**3.82**
ball	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	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	2	4	1			**1.38**			**1.03**			**0.00**			**2.58**
lotkaVolterra	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	2	2	1			**0.65**			**0.43**			**0.01**			**1.13**
spacecraft	2	4	1			**1.20**			**0.72**			**0.01**			**2.01**
sys_bio	2	9	1			**13.17**			**12.60**			**0.20**			**27.40**
duffing	2	1	2			**3.92**			**2.98**			**0.06**			**7.34**

⇒ Segment clustering dominates the time of trace similarity-based methods;

⇒ Dainarx is 1.5–4 times faster than FaMoS; 2–6 times faster than LearnHA;

Inference Time

Benchmark Details				Segmentation Time			Clustering Time			Guard-Learning Time			Total Time		
Name	$\lvert Q \rvert$	$\lvert X \rvert$	k	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx	LearnHA	FaMoS	Dainarx
buck_converter	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	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	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	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	3	1	2	–	**0.14**	0.55	–	1.30	**0.31**	–	0.01	**0.00**	–	1.45	**0.92**
two_state_HA	2	1	2	–	**0.20**	0.54	–	1.50	**0.22**	–	**0.01**	**0.01**	–	1.72	**0.80**
variable_heating_syst	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	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	2	2	1	2.14	–	**0.93**	5.83	–	**0.36**	0.48	–	**0.03**	8.46	–	**1.41**
tanks	4	2	1	2.21	–	**1.17**	16.98	–	**1.50**	1.56	–	**1.04**	20.79	–	**3.82**
ball	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	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	2	4	1	–	–	**1.38**	–	–	**1.03**	–	–	**0.00**	–	–	**2.58**
lotkaVolterra	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	2	2	1	–	–	**0.65**	–	–	**0.43**	–	–	**0.01**	–	–	**1.13**
spacecraft	2	4	1	–	–	**1.20**	–	–	**0.72**	–	–	**0.01**	–	–	**2.01**
sys_bio	2	9	1	–	–	**13.17**	–	–	**12.60**	–	–	**0.20**	–	–	**27.40**
duffing	2	1	2	–	–	**3.92**	–	–	**2.98**	–	–	**0.06**	–	–	**7.34**

- ⇒ Segment clustering dominates the time of trace similarity-based methods;
- ⇒ Dainarx is 1.5–4 times faster than FaMoS; 2–6 times faster than LearnHA;
- ⇒ Time complexity : $\mathcal{O}(|\mathcal{D}|^3 \cdot n + |\mathcal{D}|^2 \cdot d^2 \cdot n)$, i.e., *polynomial* in the size of the learning data $\mathcal{D}$ and the size of the target system d (much lower in practice).

Summary

"Inferring a nonlinear hybrid automaton from a set of input-output traces of an HS."

Main results :

- Dainarx : threshold-free trace segmentation and clustering via NARX model fitting ;
- The first approach that admits the inference of high-order non-polynomial dynamics with exogenous inputs, non-polynomial guard conditions, and linear resets ;
- Dainarx exhibits promising performance for inferring diverse, complex nonlinear systems.

Summary

"Inferring a nonlinear hybrid automaton from a set of input-output traces of an HS."

Main results :

- Dainarx : threshold-free trace segmentation and clustering via NARX model fitting ;
- The first approach that admits the inference of high-order non-polynomial dynamics with exogenous inputs, non-polynomial guard conditions, and linear resets ;
- Dainarx exhibits promising performance for inferring diverse, complex nonlinear systems.

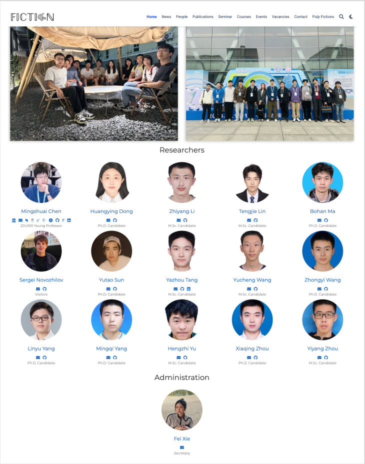
Future directions :

- Integrate Dainarx with ML for learning "good" NARX templates ;
- Extend Dainarx to robust HS identification that admits noise (almost done !) ;
- Establish quantitative guarantees via, e.g., probably approximately correct (PAC) learning ;
- Unleash Dainarx for online learning ?

⇒ H. Yu, B. Ma, H. Dong, M. Chen, J. An, B. Gu, N. Zhan, J. Yin : *Derivative-Agnostic Inference of Nonlinear Hyb. Syst.*

Formal Verification Group @ Zhejiang University

FICTION



Postdoc · Ph.D. · Master · Intern

✉ m.chen@zju.edu.cn

fiction-zju.github.io

Consider Submitting A Paper!

Max Tschaikowski

QEST+FORMATS @ CONFEST 2026

Liverpool, September 1-5, 2026

`confest-2026.github.io/qestformats/`

Mingshuai Chen