POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Data-driven discovery of dynamical systems through Symbolic-SINDy

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Author: **Mattia Gastoldi**

Student ID: 252166
Advisor: Prof. Andrea Manzoni
Co-advisors: Dr. Paolo Conti
Academic Year: 2024-25

Abstract

The process of distilling parsimonious closed-form models directly from measurements of a system is known as (Data-driven) Model Discovery. While Symbolic Regression and Sparse Identification of Nonlinear Dynamics (SINDy) have emerged as leading methodologies, they often suffer, respectively, from high computational costs and the requirement of prior knowledge, limiting their applicability when faced with more challenging tasks. Symbolic-SINDy is a promising framework that bridges these two algorithms, taking advantage of their complementary strengths and overcoming their respective practical limitations. Indeed, through their synergy, it is possible to increase user independence while preserving computational and time efficiency. This work consolidates and extends the Symbolic-SINDy framework in different scenarios. First, the method proves its effectiveness with respect to complex dynamical systems, where both Symbolic Regression and SINDy struggle with the model discovery scope. Subsequently, Symbolic-SINDy is combined with a change-point detection algorithm to address the online learning problems. The proposed implementation demonstrates its flexibility in adapting the discovered model to abrupt changes in the underlying dynamics over time. Finally, the framework is integrated with a convolutional autoencoder to perform model discovery within a latent space. The achieved results demonstrate that Symbolic-SINDy effectively identifies latent dynamics consistent with high-dimensional behavior, proving to be a powerful tool for Reduced Order Modeling (ROM).

Keywords: Model Discovery, Symbolic Regression, Sparse Identification of Nonlinear Dynamics (SINDy), Genetic Programming, Convolutional Autoencoders, Reduced-Order Modeling, Scientific Machine Learning.

Abstract in lingua italiana

Il processo di estrazione di modelli parsimoniosi in forma chiusa direttamente dalle misurazioni di un sistema è noto come Scoperta di Modelli (guidata dai dati). Sebbene la Regressione Simbolica e l'Identificazione Sparsa di Dinamiche Non Lineari (SINDy) siano emerse come metodologie di riferimento, spesso soffrono, rispettivamente, di costi computazionali elevati e della necessità di conoscenze preliminari, limitando la loro applicabilità quando si trovano ad affrontare compiti più impegnativi. Symbolic-SINDy è un framework promettente che collega questi due algoritmi, sfruttando i loro punti di forza complementari e superando i rispettivi limiti pratici. Infatti, grazie alla loro sinergia, è possibile aumentare l'indipendenza dell'utente preservando l'efficienza computazionale e temporale. Questo lavoro consolida ed estende il framework di Symbolic-SINDy in diversi scenari. In primo luogo, il metodo dimostra la sua efficacia rispetto a sistemi dinamici complessi, dove sia la Regressione Simbolica che SINDy hanno presentato difficoltà nella scoperta del modello. Successivamente, Symbolic-SINDy viene combinato con un algoritmo di rilevamento dei punti di cambiamento per affrontare i problemi di apprendimento online. L'implementazione proposta dimostra la sua flessibilità nell'adattare il modello scoperto ai cambiamenti improvvisi nelle dinamiche sottostanti nel tempo. Infine, il framework è integrato con un autoencoder convoluzionale per eseguire la scoperta di modelli all'interno di uno spazio latente. I risultati ottenuti dimostrano che Symbolic-SINDy identifica efficacemente le dinamiche latenti coerenti con il comportamento ad alta dimensionalità, dimostrandosi uno strumento potente per la Modellazione a Ordine Ridotto (ROM).

Parole chiave: Scoperta di Modelli, Regressione Simbolica, Identificazione Sparsa di Dinamiche Non Lineari, Programmazione Genetica, Autoencoder Convoluzionali, Modelli di Ordine Ridotto, Machine Learning Scientifico.

Contents

Introduction

The goal of Science is to understand and interpret phenomena observed in nature. Mathematical equations offer a quantitative representation of reality, as they allow us to formally describe the links and interactions among the phenomena involved. Nowadays, the process of identifying the underlying laws directly from measurements of a system goes under the name of *(data-driven) model discovery*. Today, model discovery lies at the intersection of interpretability, automation, and physical consistency, and plays a fundamental role in those fields of science for which the governing equations are unknown, such as climate science, neuroscience, ecology, finance, and epidemiology [4].

Historically, one of the most relevant examples of identifying a physical law directly from observations is Kepler's discovery of planetary motion. Indeed, in the early 1600s, equipped with the most extensive and accurate planetary data of the era, he developed a data-driven model for the motion of the planets, resulting in his revolutionary elliptical orbits. However, this process took him more than four years, demonstrating how the transition from measurements to models is often long and complex, and sometimes requires significant interaction with the researcher in terms of intuition and problem-solving. Today, with the rapid developments in Machine Learning and Data Science, it is possible to analyze and extract information from highly complex datasets, going beyond human capabilities. Furthermore, in recent years, significant progress has been made in defining methods capable of automatically distilling physical models of dynamical processes directly from big data. Overall, the automatic discovery of physically consistent and interpretable laws is now a task that can often be solved in a short time, with only limited interaction with the scientist [5].

A relevant turning point in model discovery was the introduction of symbolic regression [28]. Since the very first applications, this approach has proven to be an effective method for solving an extremely complex problem: determining the complete underlying structure of a nonlinear dynamical system, entirely from observations. This result is achieved by taking advantage of genetic programming optimization [19]. By construction, symbolic regression does not require any prior assumption on the functional structure of the equation. For this reason, it represents a general and flexible method with a potentially wide

range of applications. However, symbolic regression is extremely expensive, as it requires evaluating a very large number of possible functional configurations. Moreover, it does not scale well to large systems, and it may be prone to overfitting. Later, innovative techniques based on sparse regression emerged, such as Sparse Identification of Nonlinear Dynamics (SINDy) [4]. In this case, the model discovery problem is formulated from an entirely new perspective: physical systems have only a few relevant terms that determine the dynamics, making the governing equations sparse in a high-dimensional nonlinear function space. Given a predefined library of candidate (possibly nonlinear) basis functions, SINDy formulates model discovery as a sparse linear regression problem to identify a small set of active terms and their coefficients. This reformulation substantially reduces the search space and scales more favorably with problem size compared to symbolic regression approaches. By construction, this method guarantees the identification of a parsimonious expression to model the dynamical system whose state has been observed. Conversely, prior knowledge is required to select a suitable set of basis functions and the stability of the model found heavily depends on the nonlinearities pre-specified by the user. Given the excellent results obtained in many scenarios, both approaches have subsequently inspired the development of other methods, trying to enlarge their applicability in model discovery and enhancing the available performance. Among these, there are extensions designed to handle noisy data [11, 14, 17], to employ specifically designed loss functions [23], to discover partial differential equations (PDEs) [18, 27] and last, but not least, to leverage the latest techniques from deep learning, such as deep-symbolic models [2, 10, 15, 16, 34, 37] and reduced order models (ROMs) [6, 8, 9]. For a comprehensive review of the main techniques, see, e.g., [5].

The combination of the two methods discussed above has recently produced promising results in the context of automated model discovery. Recently introduced, Symbolic-SINDy [24], is a hybrid startegy leveraging both symbolic regression and SINDy, designed so that the complementarity of the two approaches gives rise to a more effective and efficient implementation. Specifically, this method can be interpreted as an extension of SINDy, where the library of nonlinear functions is automatically populated based on the nonlinearities identified through the flexibility of symbolic regression.

To prove the advantages offered by this method, in this work Symbolic-SINDy is tested on more realistic and complicated problems, where both SINDy and symbolic regression show critical issues. In addition, an uncertainty quantification routine is also integrated, which is useful to evaluate both the stability of the model and the impact of the automatically inserted building block. Then Symbolic-SINDy, combined with an appropriate change-point detection algorithm, is applied to online-learning problems using time-varying systems as

test cases. This type of setup allows us to show the stability of the method when facing sudden changes in dynamics. Finally, the proposed method is integrated into the context of reduced order modeling for dynamical systems, when the observations belong to high-dimensional dynamics. In such a situation, the already proposed solution [24] turns out to be ineffective, requiring a novel integration of symbolic regression at the latent level.

This thesis is structured as follows. First, Chapter 1 consists of a theoretical and methodological overview of symbolic regression and SINDy, discussing their strengths and limitations. Chapter 2 presents Symbolic-SINDy in its new framework from an implementation perspective. Subsequently, in Chapter 3, the new method is tested against two challenging test cases, and its performance is compared with that of symbolic regression and SINDy. In Chapter 4 Symbolic-SINDy is combined with a change-point detection method to address online learning and control problems. Then, in Chapter 5, the method is integrated at the latent level of a Convolutional Autoencoder to improve the identification of the dynamics of a high-dimensional vector field. Finally, in Chapter 6, some concluding remarks are reported.

1 | Data-driven Model Discovery: existing tools

This chapter provides a theoretical and methodological overview of the main tools employed throughout this thesis. First, the mathematical formalism of dynamical systems and ordinary differential equations (ODEs) is recalled. Subsequently, the two main approaches to data-driven model discovery are analyzed, specifically Symbolic Regression and Sparse Identification of Nonlinear Dynamics, describing their respective strengths and weaknesses.

1.1. Dynamical systems & state-space representation

A **dynamical system** is a mathematical model used to describe the evolution of a process over time, which can be represented in different ways. A comprehensive formulation is the so-called state-space representation, characterized by the explicit relations between measures and time.

1.1.1. State-space representation and autonomous formulation

The state-space representation is based on the concept of state, defined as the smallest set of variables needed to fully determine the condition of a system at a specific time instant t. These variables are collected in a vector $\mathbf{x}(t) = [x_1(t)\ x_2(t)\ \cdots\ x_n(t)]^\top \in \mathbb{R}^n$, called the *state vector*. Once the state vector at a given time is known, describing the evolution of the system simply means describing how the state evolves. Mathematically speaking, to address this problem in the continuous-time setting, the notion of **Ordinary Differential Equations (ODEs)** can be employed. Indeed, a dynamical system can then be represented by a differential equation

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t; \boldsymbol{\beta}), \tag{1.1}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state vector that evolves in time t, $\boldsymbol{\beta} \in \mathbb{R}^p$ is the vector of constant parameters which the system depends on, and $\mathbf{f}(\cdot;\boldsymbol{\beta})$ is the vector field that determines the dynamics of the system. Generally, $\mathbf{f}(\cdot;\boldsymbol{\beta})$ is Lipschitz continuous in $\mathbf{x}(t)$ to guarantee the existence and uniqueness of solutions [3]. Throughout this thesis, $\dot{\mathbf{x}}(t)$ denotes the time derivative of the vector $\mathbf{x}$ evaluated at time t. Moreover, when no ambiguity arises, $\mathbf{f}$ refers to the vector field without explicitly indicating its arguments.

The state vector solution $\mathbf{x}(t) \in \mathbb{R}^n$, obtained by solving (1.1) over the time interval $[t_0,\ T]$ with a given initial condition $\mathbf{x}(t_0) \in \mathbb{R}^n$, can be interpreted as a curve in space, evolving over time. This curve is called a *trajectory*, and the space in which it lies is called the *state space* or *phase space* of the system. Because this space is n-dimensional, (1.1) can be referred to as an *n-dimensional* system or an n^{th}*-order* system [30].

It is worth noting that the system (1.1) is generally formulated in non-autonomous form, since time dependence explicitly appears in $\mathbf{f}$. It is possible to transform a system from a non-autonomous formulation to an autonomous one simply, by introducing an additional state variable $x_{n+1} = t$ whose time derivative is $\dot{x}_{n+1} = 1$. By defining the extended state

$$\tilde{\mathbf{x}}(t) = \begin{bmatrix} \mathbf{x}(t) \\ t \end{bmatrix} \in \mathbb{R}^{n+1},$$

and

$$\tilde{\mathbf{f}}(\tilde{\mathbf{x}}(t);\boldsymbol{\beta}) = \begin{bmatrix} \mathbf{f}(\mathbf{x}(t), t;\boldsymbol{\beta}) \\ 1 \end{bmatrix} \in \mathbb{R}^{n+1},$$

the system (1.1) can be rewritten in the equivalent autonomous form:

$$\dot{\tilde{\mathbf{x}}}(t) = \tilde{\mathbf{f}}(\tilde{\mathbf{x}}(t);\boldsymbol{\beta}), \tag{1.2}$$

which depends only on the extended state and not explicitly on time. Moreover, given the important role that the system parameters $\boldsymbol{\beta}$ play in the description of the dynamics behavior, it is possible to include them, or eventually a subset of them, in the modeling procedure as additional variables to obtain more general models. This is done with the same technique, by extending the state vector with the desired parameters and appending to the dynamics the corresponding missing trivial ODEs ($\dot{\boldsymbol{\beta}} = 0$).

1.1.2. Linear and nonlinear systems

The formulation of $\mathbf{f}(\cdot;\boldsymbol{\beta})$ is crucial in determining the properties and equilibria of a dynamical system. The simplest dynamics to deal with are those for which $\mathbf{f}$ is a linear

time-invariant map: $\mathbf{f}(\mathbf{x}(t), t; \boldsymbol{\beta}) = \mathbf{A}\mathbf{x}(t)$. For example, electrical circuits and many systems in solid and fluid mechanics can be represented as linear time-invariant systems [30]. These dynamical systems are called *linear systems* in the sense that if $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ are solutions, then any linear combination $c_1\mathbf{x}_1(t)+c_2\mathbf{x}_2(t)$ is also a solution. This property goes under the name of *superposition principle*: in this way, a linear system can be broken down into parts, then each part can be solved separately and finally recombined. This property allows for a powerful simplification of complex problems, since a linear system is precisely equal to the sum of its parts. Furthermore, for these systems, unless $\mathbf{A}$ is singular, equilibria correspond only to fixed points, whose stability can be assessed by examining the eigenvalues of $\mathbf{A}$.

However, not all dynamics can be adequately described by a linear map. Typically, $\mathbf{f}$ can be characterized by nonlinear terms such as products, powers, and composite functions of the state variables. Most complex phenomena are, in fact, governed by nonlinear systems. There are several examples in physics [35], chemistry [12], ecology [21, 36], just to mention a few examples [30]. Nonlinear systems are generally much more difficult – or even impossible – to solve analytically. In some cases, the system can be locally linearized to study its dynamics. Conversely, for other systems, nonlinearities cannot be ignored when considering the global behavior. It should also be noted that the study of equilibria becomes considerably more complicated in the presence of nonlinearities that can lead to rich and complex dynamics, including limit cycles, bifurcations, chaotic behavior, and attractors [30].

1.1.3. Inverse problem: systems identification

Numerical methods to solve ODEs, given the function $\mathbf{f}(\cdot; \boldsymbol{\beta})$ and an initial condition $\mathbf{x_0} = \mathbf{x}(t_0)$, allow to simulate the dynamic of the system over a selected time interval $[t_0,\ T]$ with both high efficiency and accuracy. The presence of nonlinear terms has of course an impact in this operation, implying the solution of nonlinear system of equations at each time step if implicit schemes are adopted.

On the other hand, the problem of finding $\mathbf{f}$ given some observed trajectory of the dynamical system is definitely more involved and challenging. Indeed, frequently it is desirable to identify the vector field $\mathbf{f}$ of an unknown system in closed-form, using only measurements of the state. After collecting a finite sample of snapshots of $\mathbf{x}_t$ at N_t time instants, the objective is to symbolically estimate $\mathbf{f}$ by setting up a suitable optimization routine. Not only, the identified model must be interpretable: this allows, first of all, to explain the observed phenomenon in detail, then to understand how the variables interact with

each other, and also to make highly accurate forecasts at inference time, for time windows that fall outside the observation range. In this context, the presence of nonlinearity greatly complicates the problem of model discovery, significantly enlarging the search domain, since there are potentially infinite many expressions capable of representing the dynamics.

Different solutions can be adopted to tackle this problem. A valid method is to search within a more general set of functions, taking advantage of the effectiveness of evolutionary minimization algorithms, which are very useful to solve complex optimization problems, while still promoting parsimony of the fit [19, 23]. Another strategy is to constrain the search within a library of nonlinear functions pre-specified by the user and promote sparsity in the solution [4]. To further complicate the problem, in addition to the theoretical difficulties in finding $\mathbf{f}$, there are also difficulties related to the design of the experiment. It is useful to bear in mind that, in practice, it is not known which states of the system are truly significant, nor whether all the significant states of the system can actually be measured. Moreover, measurement noise is amplified in the estimation of the unseen time derivatives. Furthermore, it is difficult to collect data uniformly and densely over time.

1.2. Symbolic Regression

A possible approach to address the task of model discovery is provided by **Symbolic Regression (SR)** [28]. Consider the nonlinear dynamical system written in autonomous form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t); \boldsymbol{\beta}). \tag{1.3}$$

Unlike classical regression, where one seeks to estimate the optimal parameters of a pre-specified model, symbolic regression aims to discover both the model structure of $\mathbf{f}$ and its parameter values entirely from data. The basic idea is to take advantage of the increasing computational power to explore a large space of candidate expressions, typically produced through evolutionary strategy [19], and then select a parsimonious model that provides the best fit on the observations. This approach has the advantage of requiring only limited *a priori* knowledge about the underlying system. However, it often comes with significant computational costs.

1.2.1. Expression trees & genetic programming

A possible way to represent a mathematical expression is through *expression trees*. Let us denote by $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ the set of symbolic state variables and by $\mathcal{F}$ the set

of allowed basic operators, called the functions set. This latter potentially contains basic operations such as $+, -, \times, \div, \wedge$ and arithmetic functions such as $\sin, \cos, \log, \exp, \ldots$. Note that binary operations are not necessarily symmetric (e.g. $a^b \neq b^a$), while functions usually take a single input argument. The functions set is specified by the user, e.g., $\mathcal{F} = \{+, -, \times, \sin, \log\}$. Finally, let us denote by $\mathcal{C} = (a, b)$ where $a, b \in \mathbb{R}$, the real interval in which the constant parameters are defined. Also in this case, the user specifies a and b. An **expression tree** is *an incomplete binary tree* whose nodes are either state variables, constants, or basic operators. Specifically, each non-leaf node is filled with an allowed basic operator, whose number of children is either two or one, depending on whether the operation is binary or unary. Leaf nodes contain either a state variable or a constant. For example's sake, the expression tree of $g(\mathbf{x}) = \cos(x_1^2 + x_2)$ is provided in Figure 1.1.

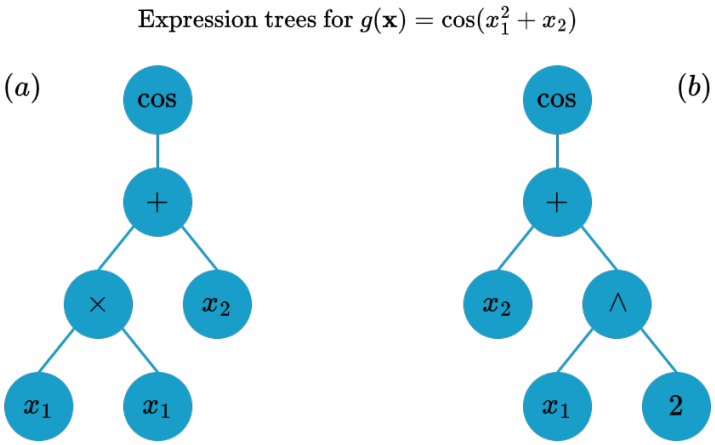


Figure 1.1: Tree representation of $g(\mathbf{x}) = \cos(x_1^2 + x_2)$. Note that both (a) and (b) represent the same mathematical equation.

It is worth noticing that for one mathematical expression there might exist many different equivalent expression trees able to represent it, as in the example reported in Figure 1.1. Consider also that if $(\mathcal{X}, \mathcal{F}, \mathcal{C})$ is not sufficiently expressive, a mathematical expression may not be represented by an expression tree. In other words, the relationship between a mathematical expression and its expression tree is not bijective and strongly depends on how $(\mathcal{X}, \mathcal{F}, \mathcal{C})$ is defined.

The tree-based representation of a mathematical equation is quite flexible, enabling the discovery of a function using **Genetic Programming (GP)** [19]. Genetic programming is a powerful generalization of evolutionary algorithms that simultaneously optimizes both the structure and the parameters of an unknown input–output map. The procedure starts from a randomly generated large population of expression trees. At each iteration, candi-

date expression trees are transformed to generate new individuals of the next generation through a set of stochastic perturbations. A tree can undergo three types of perturbation: *reproduction*, *mutation*, and *crossover*. Reproduction simply copies an individual without modifying its structure. This operator is usually the least frequent, but it allows well-performing expressions to reproduce. Mutation, instead, acts with a random local change in the expression tree, either by replacing a single node or by recreating a new portion of tree. Finally, crossover exchanges subtrees of two different individuals in the population. This operation is particularly important, as it allows the most informative building blocks of the expressions to propagate across the population. A schematic representation of these perturbations is shown in Figure 1.2.

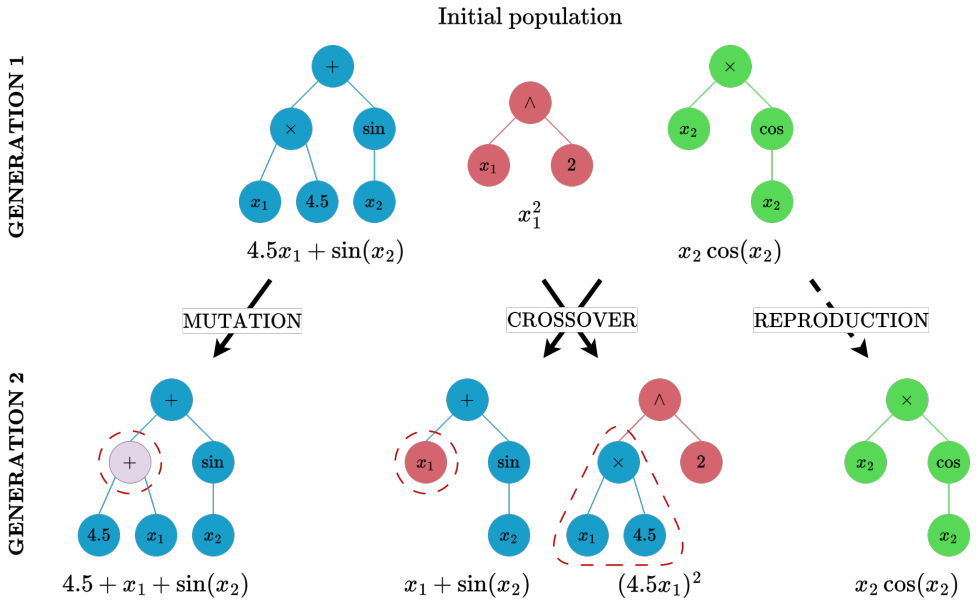


Figure 1.2: Candidate expression trees of the current iteration are used to generate the ones for the next generation through reproduction, mutation, and crossover.

The new mathematical expressions generated are then evaluated using the training data and only the top performers are retained to initialize the next generation. This implementation mimics the biological evolution process of nature to create the genetic material of living organisms, and only the most adapted characteristics survive and propagate through the population.

Denote as $\mathcal{D} = \{\mathbf{x}^{(k)}, y^{(k)}\}_{k=1}^{N}$ a general training dataset of N measured multivariate inputs and corresponding scalar outputs. The selection of expression trees used in the production of the next generation is based on the prediction error computed on the training data $\mathcal{D}$. However, it is important to consider that, during this minimization, in general, a more complex expression will be able to fit the given dataset at least as well as a less complex

expression. In other words, as the complexity of the expression increases, the error within the training set tends to decrease. This phenomenon is known as *overfitting*. Indeed, performance measured on a hypothetical test set tends to worsen as complexity increases. For this reason, the loss metric should incorporate structural constraints or penalties to control the complexity of the generated trees and guarantee a good generalization of the estimated expression. Therefore, the general form of the objective function is as follows:

$$f(\cdot, \boldsymbol{\theta}) = \arg \min_{f'(\cdot,\boldsymbol{\beta}')} C\left(f'(\cdot, \boldsymbol{\beta}'), \mathcal{D}\right) + \lambda \mathcal{L}\left(f'(\cdot, \boldsymbol{\beta}')\right), \tag{1.4}$$

where C measures the goodness of fit of the estimated $f(\cdot, \boldsymbol{\theta})$ with respect to the training data $\mathcal{D}$, while $\mathcal{L}$ measures the complexity of the expression, e.g., the number of nodes in the corresponding tree. Here, λ is called the parsimony coefficient, and larger values of λ promote the selection of simpler expressions.

The core mechanism underlying most symbolic regression algorithms is obtained by combining the tree representation of a mathematical expression and the genetic programming optimization strategy.

By construction, symbolic regression serves as a natural tool for the model discovery task, as it is capable of simultaneously identifying the functional form of the dynamics $\mathbf{f}$ and the constant parameters $\boldsymbol{\beta}$ that govern the evolution of the state $\mathbf{x}(t)$.

1.2.2. SR-T and D-CODE

To identify the form of the function $\mathbf{f}$ in (1.3) from the data, a history of snapshots of the state $\mathbf{x}(t)$ and its time derivative $\dot{\mathbf{x}}(t)$ is collected at N_t time instances $t_1, t_2, \ldots, t_{N_t} \in [0, T]$. Note that time derivatives $\dot{\mathbf{x}}(t)$ are not required to be measured as they can be computed directly from state observations using finite differentiation or, in the case of noisy data, numerical differentiation [7]. A dataset $\mathcal{D} = \{\mathbf{x}(t_k), \dot{\mathbf{x}}(t_k)\}_{k=1}^{N_t}$ is then defined.

As in the standard symbolic regression framework, also for the identification of an ODE the goodness of fit C in (1.4) can be measured simply by the mean-squared error (MSE) between the mapped input and its corresponding time derivative:

$$C\left(f_i'(\cdot, \boldsymbol{\beta}_i'), \mathcal{D}\right) = \frac{1}{N_t} \sum_{j=1}^{N_t} \|\dot{x}_i(t_j) - f_i'(\mathbf{x}(t_j), \boldsymbol{\beta}_i')\|^2, \qquad \forall i \in 1, \ldots, n.$$

Although this formulation is basic, it immediately generated considerable interest because of its success in several physical applications, particularly in the estimation of conservation

laws [28]. In this thesis, symbolic regression equipped with this loss function will be referred to as **SR-T**.

A practical limitation of SR-T in the ODE discovery is that the state derivatives explicitly appear in the loss function. Therefore, one needs to compute them directly from noisy measurements, and derivative estimation becomes unstable, degrading the model accuracy. For ODE identification, **D-CODE** [23] is a more robust method, as the objective function is based on the variational formulation of ODEs to bypass the computation of unobserved time derivatives. Specifically, the goodness of fit C in (1.4) is formulated as:

$$C(f_i'(\cdot, \boldsymbol{\beta}_i'), \mathbf{x}, g) = \int_0^T f_i'\big(\mathbf{x}(t), \boldsymbol{\beta}_i'\big) g(t)\, dt \; + \; \int_0^T x_i(t)\, \dot{g}(t)\, dt \qquad \forall i \in 1, \dots, n,$$

where $g(\cdot)$ is a testing function (e.g. a spline basis). The integrals can be computed numerically based on the discretization of the observed time domain $[0, T]$. D-CODE has been shown to be more accurate and robust than SR-T while retaining comparable computational costs. See, e.g., [23] for further details.

1.2.3. Strengths & limitations

Symbolic regression is an extremely powerful method in the context of model discovery. Unlike neural networks or black-box approaches, it is able to return a closed-form mathematical equation that links the quantity of interest with the predictors. Therefore, it enables one to interpret the phenomenon, understand causality, interactions, and the impact of each measured state on the dynamics. Furthermore, symbolic regression is more flexible than classical parametric regression models. By simultaneously searching for both the form of the equation and the associated parameters, it is possible to properly identify more complex structures that better describe a general unknown system. In addition to these advantages, prior knowledge of the problem is extremely limited. In fact, interaction with the user is reduced to a minimum, and is limited to criticism of the model found and tuning of the hyperparameters. However, these advantages are offset by a series of important limitations. The first concerns the search within the function space, which is an NP-hard problem: the computational times increase exponentially as the search space grows. Moreover, if the function set is too broad, some dynamics could be mistaken for others (e.g., nonlinear higher-order polynomial oscillations may be interpreted as trigonometric combinations). For correct identification, the dynamics must also be sufficiently informative over the observation interval. A further limitation is that each expression is solved individually, making the algorithm unusable for medium-to high-dimensionality problems. Additionally, in the context of automated model discovery, not all equations

found may have a physical meaning [28], and regularization tuning is essential to avoid overfitting. Finally, with SR-T, the temporal derivatives computed from noisy measurements are potentially unstable, affecting the accuracy of the discovery. On the other hand, D-CODE, although more robust, may sometimes exhibit the "*bloat*" phenomenon, identifying overly complex ODEs whose dynamics deviate significantly from the true ones.

1.3. Sparse identification of nonlinear dynamics

Consider the nonlinear dynamical system written in autonomous form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t); \boldsymbol{\beta}). \tag{1.5}$$

Another popular approach to identify the underlying governing equation from collected experimental data is based on sparse regression [31]. Indeed, for many systems of interest, the function $\mathbf{f}$ often consists of only a few terms, making it sparse in the space of possible functions. The identification strategy is provided by the so-called **Sparse Identification of Nonlinear Dynamics (SINDy)** [4], where a sparsity-promoting linear regression on a pre-specified nonlinear function library is performed. This method allows determining which terms in the library are non-zero without performing a computationally intractable brute-force search.

1.3.1. SINDy algorithm

To identify dynamics $\mathbf{f}$ in (1.5) from the data, a history of snapshots of the state $\mathbf{x}(t)$ and its time derivative $\dot{\mathbf{x}}(t)$ are collected at N_t time instances $t_1, t_2, \ldots, t_{N_t} \in [0, T]$. The data are then arranged into two matrices:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^\top(t_1) \\ \mathbf{x}^\top(t_2) \\ \vdots \\ \mathbf{x}^\top(t_{N_t}) \end{bmatrix} = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_{N_t}) & x_2(t_{N_t}) & \cdots & x_n(t_{N_t}) \end{bmatrix},$$

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}^\top(t_1) \\ \dot{\mathbf{x}}^\top(t_2) \\ \vdots \\ \dot{\mathbf{x}}^\top(t_{N_t}) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \cdots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \cdots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_{N_t}) & \dot{x}_2(t_{N_t}) & \cdots & \dot{x}_n(t_{N_t}) \end{bmatrix},$$

where $x_i(t_j)$ corresponds to the i-th entry of $\mathbf{x}$, evaluated at the j-th time instant, and the same notation for the matrix of the time derivatives. Note that in practice it is not required to measure $\dot{\mathbf{X}}$, since this matrix can be computed directly from the data in $\mathbf{X}$, using either finite differentiation or, in the case of noisy data, numerical differentiation [7].

Next, a library of candidate functions is chosen, consisting of a finite number of elements $\boldsymbol{\Theta}(\mathbf{x}) = [\theta_1(\mathbf{x})\ \theta_2(\mathbf{x})\ \dots\ \theta_l(\mathbf{x})] \in \mathbb{R}^l$. $\boldsymbol{\Theta}$ is called the *Candidate Function Library* (CFL). In principle, the CFL can be populated by any kind of function of the state vector $\mathbf{x}$, including nonlinear and complex functions. The choice of the library is problem-dependent and must be specified by the user. Generally, a standard choice to build an informative CFL is to include the intercept, a few polynomial functions of the state vector (e.g. up to the third degree), and optionally the base trigonometric expressions. Once the library $\boldsymbol{\Theta}$ is defined, the design matrix of the linear regression is obtained by applying $\boldsymbol{\Theta}$ to $\mathbf{X}$. For example $\boldsymbol{\Theta}(\mathbf{X})$ can be defined as:

$$\Theta(\mathbf{X}) = \begin{bmatrix} | & | & | & | & & | & | & \\ \mathbf{1} & \mathbf{X} & \mathbf{X}^2 & \mathbf{X}^3 & \cdots & \sin(\mathbf{X}) & \cos(\mathbf{X}) & \cdots \\ | & | & | & | & & | & | & \end{bmatrix},$$

where higher-order polynomials are denoted as $\mathbf{X}^2, \mathbf{X}^3, \dots$. Here, $\mathbf{X}^2$ denotes the quadratic nonlinearities in the state variable:

$$\mathbf{X}^2 = \begin{bmatrix} x_1^2(t_1) & x_1(t_1)x_2(t_1) & \cdots & x_2^2(t_1) & x_2(t_1)x_3(t_1) & \cdots & x_n^2(t_1) \\ x_1^2(t_2) & x_1(t_2)x_2(t_2) & \cdots & x_2^2(t_2) & x_2(t_2)x_3(t_2) & \cdots & x_n^2(t_2) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_1^2(t_{N_t}) & x_1(t_{N_t})x_2(t_{N_t}) & \cdots & x_2^2(t_{N_t}) & x_2(t_{N_t})x_3(t_{N_t}) & \cdots & x_n^2(t_{N_t}) \end{bmatrix},$$

and the definition of higher-order nonlinearities can be done similarly.

Each column of $\boldsymbol{\Theta}(\mathbf{X})$ represents a candidate function to describe the evolution of the state variables. Under the assumption that the system's behavior is governed by only a few dominant terms, the goal is to find a sparse matrix of coefficients $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1\ \boldsymbol{\xi}_2\ \dots\ \boldsymbol{\xi}_n]$ that selects the most relevant functions from the library. Specifically, the k-th column $\boldsymbol{\xi}_k$ contains the coefficients that determine which terms in $\boldsymbol{\Theta}(\mathbf{X})$ best reconstruct the time derivative of the k-th state variable. In matrix form, this relationship is expressed as:

$$\dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi} \tag{1.6}$$

where the equality sign is used under the assumption that the library $\boldsymbol{\Theta}(\mathbf{X})$ is sufficiently expansive to potentially allow for a perfect reconstruction of the observed dynamics.

To estimate the (sparse) coefficient matrix $\boldsymbol{\Xi}$ in (1.6), a least-squares framework can be employed for each column, augmented with a sparsity-promoting regularizer $\mathcal{L}$ in the objective function:

$$\boldsymbol{\xi}_i = \arg\min_{\boldsymbol{\xi}_i'} \|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\xi}_i'\|_2 + \mathcal{L}(\boldsymbol{\xi}_i') \qquad \forall i = 1, \ldots, n. \tag{1.7}$$

A natural way to solve (1.7) is to employ ℓ_1-regularization on the columns of $\boldsymbol{\Xi}$ (i.e. $\mathcal{L} = \lambda\|\boldsymbol{\xi}_i\|_1$). With this choice of $\mathcal{L}$ the problem in (1.7) is reduced to a LASSO regression [31] on $\boldsymbol{\Theta}(\mathbf{X})$. The parameter λ plays a key role in the estimation: when $\lambda = 0$, the coefficients $\boldsymbol{\xi}_i$ are estimated simply using least-squares, with no sparsity. As λ increases, an increasing number of coefficients are shrunk toward zero. Although this procedure is effective, it can be computationally expensive for very large datasets. An efficient alternative is to use the so-called Sequential Thresholded Least Squares (STLSQ). It consists of iteratively solving (1.7) in the least squares sense, and setting to zero the entries of $\boldsymbol{\xi}_i'$ with magnitude below a threshold L at each iteration. Here, the sparsity is induced by the threshold coefficient L. In both LASSO and STLSQ, the hyperparameters can be tuned using cross-validation on a subset of the collected data.

To this end, SINDy provides a practical implementation for dynamics identification, as the system behavior can be approximated as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t); \boldsymbol{\beta}) \approx \boldsymbol{\Theta}(\mathbf{x}(t))\boldsymbol{\Xi}. \tag{1.8}$$

In this framework, the function $\mathbf{f}$ is approximated by a linear combination of nonlinear candidate functions from the library, whose coefficients $\boldsymbol{\Xi}$ are determined through sparse regression. It is important to note that, while the resulting model is provided in closed-form, it generally represents an approximation of the true dynamics. Indeed, the equality in (1.8) holds only under the assumption that the library $\boldsymbol{\Theta}$ is sufficiently expressive to include all the necessary nonlinearities that define the system. In such a case, the true underlying model can be perfectly discovered, the constant parameters $\boldsymbol{\beta}$ correspond exactly to the non-zero coefficients in $\boldsymbol{\Xi}$, and the SINDy algorithm effectively performs their numerical estimation. Figure 1.3 shows how SINDy can be used to discover, e.g., the Lorenz system from data.

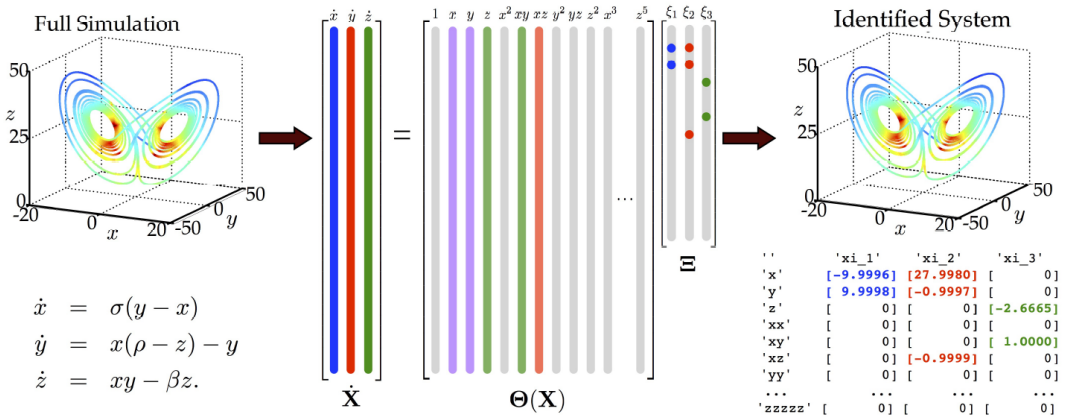


Figure 1.3: Schematic SINDy algorithm. Image sourced from [4].

1.3.2. Ensemble SINDy for uncertainty quantification

When dealing with noisy data, it is also crucial to provide uncertainty estimates for the discovered models. In this direction, recent extensions of SINDy use sparse Bayesian inference for probabilistic model discovery, namely UQ-SINDy [14]. Such methods employ Markov Chain Monte Carlo (MCMC) sampling, which is, however, extremely computationally intensive, taking several hours of runtime to identify a model. A popular alternative widely used in practice is the so-called **Ensemble-SINDy (E-SINDy)** [11]. Ensembling is a well-established machine learning technique that combines the output of multiple models to improve prediction. For model discovery, the ensembling strategy improves robustness and naturally provides uncertainty estimates for the identified model coefficients and their inclusion probability, allowing active selection of the library terms.

A possible way to produce an ensemble of SINDy models is by leveraging data bootstrapping. From the N_t rows of the data matrices $\dot{\mathbf{X}}$ and $\mathbf{\Theta}(\mathbf{X})$, corresponding to N_t samples in time, q bootstrapped data samples are generated to produce q SINDy models. Each of these q data bootstraps is composed by N_t rows, randomly sampled with replacement from the original N_t rows of the data matrices. Note that some of the entries might appear multiple times in a single data bootstrap. In principle, there are also other random data subsampling approaches that may be used, however, bootstrapping based on selection with replacement is the most standard procedure. Alternatively, to generate an ensemble of SINDy models, one can sample library terms instead of data pairs. In this case, k columns out of l columns of $\mathbf{\Theta}(\mathbf{X})$ are sampled without replacement, to produce at most $q = \binom{l}{k}$ different SINDy models. As before, different subsampling strategies are also allowed. The aggregation can be done by taking either the mean of the identified model coefficients (*bagging*) or the median for a more robust estimation (*bragging*).

E-SINDy by construction provides inclusion probabilities and uncertainty estimates for the coefficients of the discovered model. Indeed, the identified ensemble of model coefficients can be used to compute coefficient probability density functions, which approximate the posterior distribution $p(\boldsymbol{\Xi} \mid \mathbf{X})$.

This ensemble approach is extremely useful in the estimation of a robust and parsimonious model, providing inclusion probabilities and uncertainty estimates for its coefficients, especially in the presence of noisy observations and limited data availability. In contrast with other technique, E-SINDy requires only minimal computational cost, as the ensemble is produced by solving q independent sparse regression problems. This strategy applies under the same conditions as the standard SINDy algorithm, where it is assumed that all relevant variables are measured at a sufficient temporal and spatial resolution to properly approximate time derivatives. Finally, the post-processing phase can be highly flexible, since the summary statistics of the ensemble can be used to actively select terms in the library, for example by thresholding the inclusion probabilities, effectively filtering out spurious terms and reducing the variance of the estimated coefficients, thus improving the accuracy of the model discovery process.

1.3.3. Strength & limitations

Due to its effectiveness and simplicity of implementation, the SINDy algorithm represents the gold standard technique in model discovery. Through a relatively simple sparse regression on a set of nonlinear building blocks, SINDy efficiently estimates the underlying dynamics in closed-form. Unlike black-box models and neural network architectures, it is, in fact, much easier to train and, above all, completely interpretable, returning a parsimonious model in symbolic form. This allows users to deduce the causality, interactions, and specific role of each predictor in the system. Furthermore, by interpreting SINDy as a linear regularization, the method can be easily combined with more complex approaches, either as a final regularizer or as a way to provide an optimal initial guess for other methods. However, despite its simplicity, SINDy has some obvious limitations. The main limitation is that it requires prior knowledge of the problem in order to construct the CFL. Indeed, for correct identification, it is necessary to define a concise and sufficiently informative library of functions. Having a library that is too large can lead to two significant issues: increased computational complexity and estimation difficulties. As the library grows, the sparse regression problem significantly increases computational costs. Furthermore, the presence of irrelevant blocks may be misleading in the estimation of the final model. In practice, it is often impossible to include the nonlinearities characteristic of unknown systems in the library *a priori*, especially in the case of composite functions,

non-integer powers, or fractional and more exotic functions. For all these reasons, despite its numerous strengths, the SINDy algorithm may be limited for more complex real-world applications if applied without prior knowledge of the system.

2 | A new Symbolic-SINDy framework

This chapter describes a new method for automated model discovery: Symbolic-SINDy. Previously introduced in [24], it immediately showed promising results, overcoming the main limitations of both existing methods: symbolic regression and SINDy. The goal of this analysis is to adapt the method so that it can handle more complex and realistic scenarios. Moreover, the method has also been integrated with an uncertainty quantification (UQ) procedure, which allows us to assess, in a post-processing phase, the stability of both the identified model and the retained building blocks, providing the inclusion probabilities and uncertainty estimates for the model coefficients.

2.1. Symbolic-SINDy methodology

As discussed in detail in Chapter 1, both symbolic regression and SINDy have fundamental limitations for more complex real-world applications in the context of automated model discovery. Although symbolic regression is totally suitable for this task, since it requires limited knowledge of the problem, it nevertheless presents two significant critical issues. First the computational overhead of the operation is significant because it requires a lengthy and costly minimization. Furthermore, this method is extremely prone to overfitting, returning closed-form equations that are sometimes more complex than necessary. On the other hand, SINDy, despite relying on an extremely efficient algorithm, requires a concise and sufficiently informative library of functions in order to correctly identify the underlying dynamics. This library is chosen by the user, incorporates prior knowledge of the problem, and is key to the success of the identification procedure. However, it is often impossible to include those nonlinearities that are peculiar of unknown systems in the library *a priori*, especially in the case of composite functions, non-integer powers, or fractional – and even more involved functions.

Symbolic-SINDy is conceived as a new model discovery approach, able to overcome the limitations shown by symbolic regression and SINDy, as well as to extend model

identification scope. The idea behind Symbolic-SINDy is to combine the *complementary* strengths of both existing approaches. Specifically, the proposed method is an extension of SINDy, with a new automatic mechanism of selection of candidate functions. This framework leverages the flexibility of symbolic regression to search for promising nonlinearities within a large functional space [24].

2.1.1. Symbolic-SINDy algorithm

The Symbolic-SINDy algorithm is conceptually divided into two main phases. Initially, a preliminary execution of symbolic regression is run. The goal of this execution is to analyze and extract some suitable building blocks from the most promising expression trees obtained during genetic programming optimization. It is worth noting that, generally speaking, at this stage, one is not interested in the convergence of symbolic regression towards a definitive and stable solution, but rather in gathering a group promising building blocks from the population of trees. Each of these small mathematical expressions extracted is then integrated into a polynomial library, effectively becoming an automatically added nonlinearity. Sparse regression acts as a regularizer, ensuring the identification of a parsimonious model in closed-form. Among the various SINDy models fitted, one for each extracted building block, the best one is then chosen, i.e. the one that minimizes the error with respect to the training data. This model is then saved and returned. The procedure is summarized in detail in Algorithm 2.1, and a schematic representation is provided in Figure 2.1.

Algorithm 2.1 Symbolic-SINDy algorithm (pseudo-code)

Require: Training data $\mathbf{X}, \dot{\mathbf{X}}$; symbolic regression method (either `SR-T` or `D-CODE`) with genetic configuration; SINDy threshold L.
Ensure: Symbolic-SINDy model $\hat{\mathcal{M}}$.

```
Run symbolic regression method according to the genetic configuration
Extract and filter promising building blocks B from selected generations
for each b ∈ B do
    Augment library Θ_b(X) ← A(Θ_polynomial(X), b(X))[1]
    Fit M_b via SINDy with STLSQ with threshold L
    Compute training RMSE_b
end for
Select model M̂ = M_b with minimum RMSE_b
return M̂
```

[1]The operator $\mathcal{A}(\cdot,\cdot)$ constructs the augmented library from the polynomial library and the extracted building block. It can either concatenate the features or perform a tensor (Kronecker) product, thus including interaction terms.

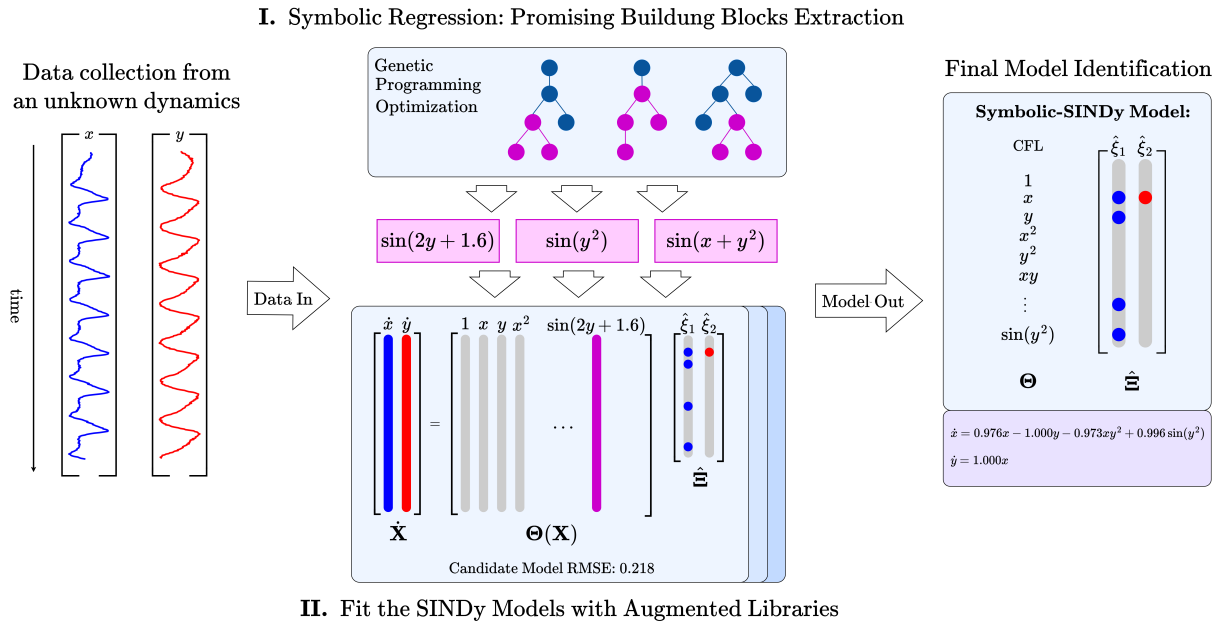


Figure 2.1: Schematic Symbolic-SINDy algorithm.

In practical terms, the implementation offers great flexibility, allowing the user to manage:

- the complexity of the extracted building blocks, as overly complex expressions are filtered out and discarded;
- the integration of the additional block into the library, which can be either concatenated to the polynomial library or combined through multiplicative interactions;
- the complexity of the identified model, by excluding overly complex SINDy models.

By leveraging these degrees of freedom, the proposed method can be made both more efficient and more effective. The theoretical advantages offered by this procedure arise from exploiting the *complementarity* of the two methods employed. In this context, it is possible to reduce the computational overhead of symbolic regression, as genetic programming is mainly used as engine for the exploration and the extrapolation of promising building blocks that survive during evolution. As convergence of the symbolic regression is not required, it is also possible to prevent overfitting to overly complex solutions. The final model is estimated using SINDy, which ensures the identification of a parsimonious and interpretable dynamics through sparse regression. The automatic integration of blocks within the nonlinear library allows, in principle, to define a CFL informed of the specific problem, ideally providing SINDy with the building blocks required for the correct identification of the dynamical system under observation, without requiring the user to have prior knowledge. In this way, it is therefore possible to construct an effective and flexible framework for data-driven model discovery.

2.1.2. Symbolic-SINDy with uncertainty quantification

With Symbolic-SINDy, which is essentially an extension of SINDy, it is possible to efficiently quantify the uncertainty in the coefficients of the discovered dynamical system using an ensemble of models, following the framework proposed in E-SINDy, as discussed earlier in Section 1.1.2. Initially, the model is estimated using the standard Symbolic-SINDy procedure. This allows the most informative building block to be identified and incorporated into the nonlinear library. Following this operation, an ensemble of SINDy models is trained using the augmented CFL previously obtained.

The resulting ensemble can be used to understand whether the added building block is truly decisive and the stability of the identified model, as well as the uncertainty associated with each of its coefficients. In particular, by observing the probability of inclusion and the distribution of the added building block, its reliability is qualitatively accessible. A low probability of inclusion and/or a distribution localized in multiple values indicates the low likelihood of the added block, suggesting that the identified model is incorrect.

In principle, model ensembles can be generated using different subsampling approaches, combining sampling of data pairs with sampling of library terms and, eventually, using also active learning on summary statistics. However, the standard procedure is based on simple data bootstrapping, i.e. selection with replacement of the data pairs. It is also possible to include into the CFL a larger number of promising building blocks, to explore a wider domain of potential nonlinearities, bearing in mind, however, that this would increase computational costs and could lead to numerical instability due to collinearity between blocks. The choice to use E-SINDy as an uncertainty quantification tool is due to its high efficiency compared to other probabilistic model discovery methods such as UQ-SINDy [14].

Finally, it should be noted that this method of uncertainty quantification is based entirely on the structure of the final model identified by Symbolic-SINDy in the first instance, completely ignoring the uncertainty associated with the blocks extracted in such a phase. Indeed, while tailoring E-SINDy is simple and efficient, quantifying the uncertainty associated with genetic programming optimization is not a straightforward operation. An ensemble or Bayesian approach could, in principle, be applied to symbolic regression [17] as well, however, both strategies would be extremely costly and potentially unstable.

3 | Numerical experiments

In previous work [24], Symbolic-SINDy proved its effectiveness with respect to common (but quite easy) benchmarks in model discovery. Specifically, it correctly identified logarithmic growth models, such as the Gompertz model, models with irrational powers, such as the generalized logistic model, and nonlinear oscillatory models with higher powers, such as the Sel'kov model. The encouraging results, in which Symbolic-SINDy totally outperformed existing methods, justify a more in-depth analysis with respect to more involved and general scenarios. In the current section, the performance of the four methods (SR-T, D-CODE, SINDy, Symbolic-SINDy) is evaluated and compared in relation to structurally more complex situations.

Benchmark models. The aforementioned methods are assessed on two benchmarks:

1. Hill equation in biochemistry;
2. revised version of van der Pol's nonlinear oscillator, including a sinusoidal forcing.

Measurement settings. To ensure a fair comparison, the same measurement settings are used for each experiment. Specifically:

- Trajectory generation: each experiment is conducted by simulating $N = 50$ trajectories over the time interval $[0, T]$, each initialized from a randomly sampled initial condition. For parametrized systems, a model parameter is also randomly sampled and kept fixed along the evolution of each trajectory;
- Sampling frequency: each trajectory is sampled at a frequency of 10 observations per time unit, corresponding to a temporal discretization $dt = 0.1$;
- Noise contamination: the data are corrupted with additive white noise at a 1% level. Specifically, zero-mean Gaussian noise with standard deviation $\sigma = \sigma_R \, \mathrm{std}(\dot{\mathbf{X}})$ is added to the observations, where $\sigma_R = 0.01$ and $\mathrm{std}(\dot{\mathbf{X}})$ denotes the standard deviation of the noise-free data;

- Random seeds: experiments are repeated using different random seeds in order to assess the stability and robustness of the employed methods.

Evaluation metrics. To assess the methods' performance, we rely on three indicators:

- Success probability: the fraction of times the exact functional form is recovered;
- RMSE: the root mean squared error of the estimated trajectories on training data;
- Execution time: the computational time required by a single execution of the algorithm.

Moreover, the generalization of each model outside the training set and its predictive capacity of will be discussed.

3.1. Hill equation

The Hill equation was first introduced by Hill to describe the equilibrium relationship between oxygen tension and the saturation of hemoglobin. In pharmacology, the Hill equation has been extensively used to analyze quantitative drug–receptor relationships. Many pharmacokinetic–pharmacodynamic models have used the Hill equation to describe nonlinear saturating drug dose–response relationships [12]. The Hill equation therefore describes the response in terms of the quantity of a macromolecule x, such as an enzyme or protein, in relation with the concentration y of an introduced ligand, such as a hormone or drug. Mathematically, this relationship is expressed by the equation

$$\dot{x} = \frac{k\, y^h}{k_a + y^h}, \tag{3.1}$$

$$\dot{y} = -k_y\, y, \tag{3.2}$$

where $h \in \mathbb{R}^+$, $k \in \mathbb{R}^+$, $k_a \in \mathbb{R}^+$, $k_y \in \mathbb{R}^+$ and:

- h is the Hill coefficient and controls the cooperativity of the system, determining the steepness of the transition between the inactive and the maximum response state.
- k represents the maximum response of the system, i.e. the maximum growth rate of x achievable for high concentrations of y.
- k_a is the half-saturation constant and identifies the drug concentration at which the system reaches half of the maximum response.
- k_y is the drug decay rate, governing the decrease of concentration y over time.

Equation (3.1) is a generalization of classic Michaelis-Menten kinetics [12], in which the concept of cooperativity between reactants emerges, increasing or decreasing their affinity. Equation (3.2) models the natural reduction over time of the concentration of the ligand responsible for the reaction, until it asymptotically disappears.

The mathematical structure of the dynamical system is rather complex, since a fraction appear in $\mathbf{f}$, where the numerator contains a power that is generally not an integer, and the denominator contains the exact same power shifted by an additive term. Typically, when observing nonlinear saturating curve phenomena in the context of drug dose–response, the Hill model is assumed as the underlying model and ad-hoc techniques are used to estimate the coefficients. Potentially, automated model discovery methods represent an appealing and more general alternative.

3.1.1. Problem setup

In the numerical experiments, the initial conditions are sampled as $(x_0, y_0) \in [0, 10] \times [0, 10]$, and the 50 trajectories are simulated over a time horizon $T = 15$ using $N_t = 150$ time steps. The Hill model parameters are fixed as $h = 2.8$, $k = 1$, $k_a = 4$, $k_y = 0.1$. In the k-parametrized setting, the parameter k is allowed to vary within the range $k \in [1, 2]$.

SINDy is run using a general polynomial CFL of the third degree, simulating a situation where no prior knowledge of the problem is available.

Symbolic regression is run using a maximum number of 20 generations and functions set $\mathcal{F} = \{+, -1, \times, \div, \wedge\}$ for both SR-T and D-CODE.

Symbolic-SINDy is run based on D-CODE for the standard case, and SR-T for the k-parametrized case, using a maximum number of 20 generations and functions set $\mathcal{F} = \{+, -1, \times, \div, \wedge\}$. The choice of using the same configuration of the symbolic regression methods is motivated and discussed in the following paragraph. The building blocks are then integrated into a general polynomial CFL of the third degree for the not parametrized problem, and of the second degree for the k-parametrized problem. A detailed description of the tuning parameters used for all the methods is provided in Appendix A.1.

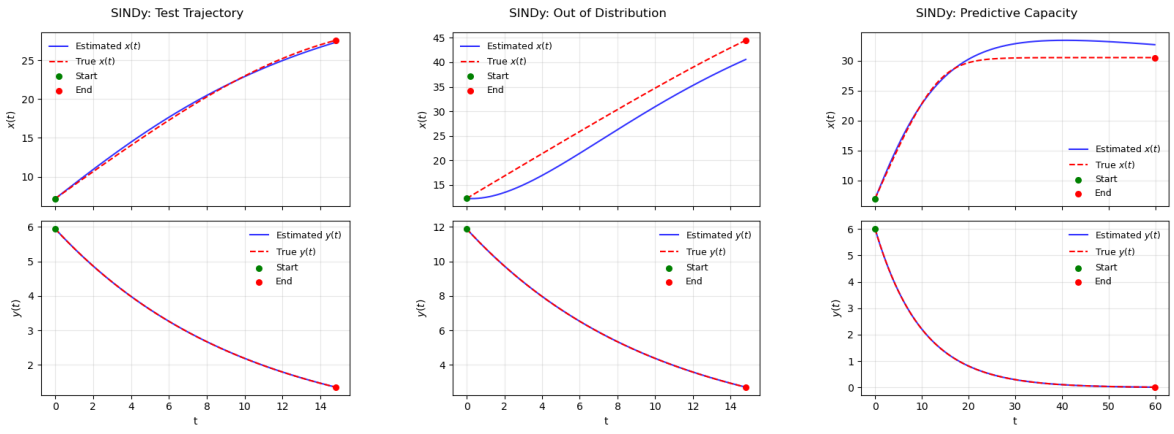
3.1.2. Numerical results

The numerical results obtained from the various experiments are summarized in Table 3.1. In addition, Figures 3.1, 3.2, and 3.3 are provided for support. A detailed version of the results for each experiment is provided in Appendix A.1.

Having used a value $h > 1$ as the Hill coefficient, the positive cooperativity of the reactants means that for high values of y, the response trajectory in x will grow linearly, until it saturates when the concentration of y drops significantly, indicatively when it is lower than the half-saturation constant k_a.

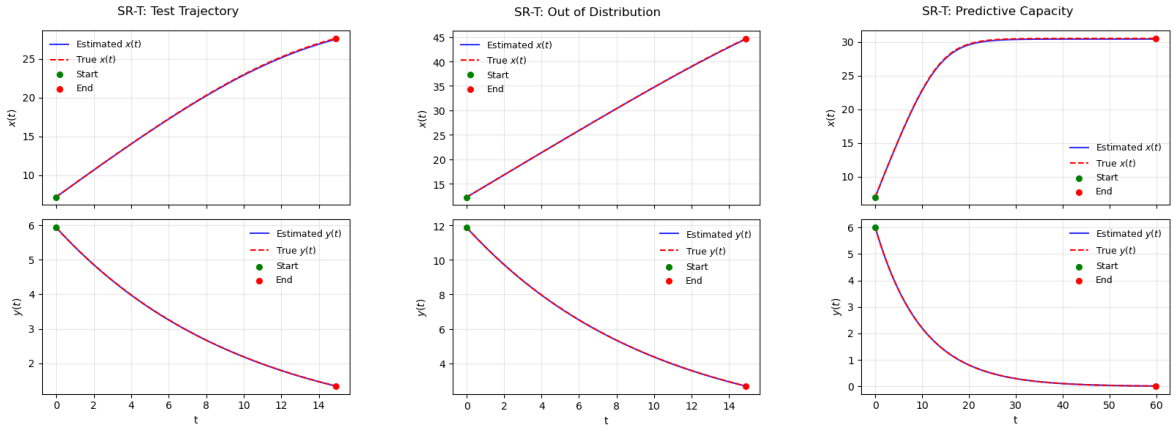
	Params.	Suc. Prob (3.1)	Suc. Prob (3.2)	RMSE	Time [m]
SINDy	–	0	1.00	0.206	$\simeq 0$
	k	0	1.00	0.279	$\simeq 0$
SR-T	–	0.80	1.00	0.098	9.81
	k	0.60	1.00	0.136	9.81
D-CODE	–	0.80	1.00	0.064	7.04
	k	0.50	1.00	0.055	7.31
Symbolic-SINDy	–	0.85	1.00	0.030	3.80
	k	0.80	1.00	0.101	5.15

Table 3.1: Performance comparison of the different model discovery methods for the Hill equation.



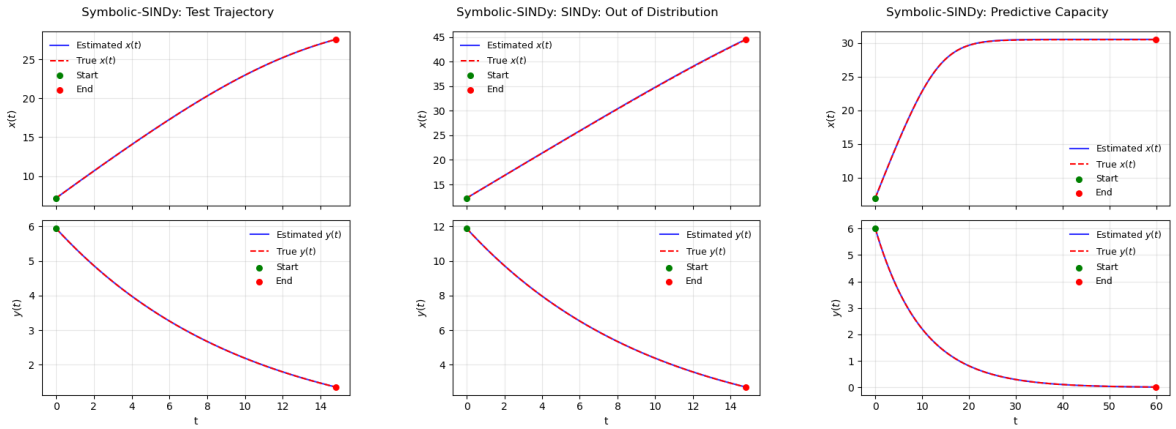
(a) $(x_0, y_0, k) = (7, 6, 1.8)$ (b) $(x_0, y_0, k) = (12, 12, 2.3)$ (c) $(x_0, y_0, k) = (7, 6, 1.8)$

Figure 3.1: SINDy for the k-parametrized Hill equation. In particular, (a) shows the estimated dynamics starting from a point within the training set; (b) shows the estimated dynamics starting from a point outside the training set; (c) shows the estimated dynamics starting from a point within the training set, with extrapolation time window $[15, 60]$.



(a) $(x_0, y_0, k) = (7, 6, 1.8)$ (b) $(x_0, y_0, k) = (12, 12, 2.3)$ (c) $(x_0, y_0, k) = (7, 6, 1.8)$

Figure 3.2: SR-T for the k-parametrized Hill equation, with `seed=102`. In particular, (a) shows the estimated dynamics starting from a point within the training set; (b) shows the estimated dynamics starting from a point outside the training set; (c) shows the estimated dynamics starting from a point within the training set, with extrapolation time window $[15, 60]$.



(a) $(x_0, y_0, k) = (7, 6, 1.8)$ (b) $(x_0, y_0, k) = (12, 12, 2.3)$ (c) $(x_0, y_0, k) = (7, 6, 1.8)$

Figure 3.3: Symbolic-SINDy for the k-parametrized Hill equation, `seed=102`. In particular, (a) shows the estimated dynamics starting from a point within the training set; (b) shows the estimated dynamics starting from a point outside the training set; (c) shows the estimated dynamics starting from a point within the training set, with extrapolation time window $[15, 60]$.

Since the functions' library of SINDy does not contain the unique building block that appears in the ground-truth dynamics, the best that SINDy can do is extrapolate the optimal polynomial combination that is locally capable of explaining the observed data.

We remark that the RMSE metric on the training data is not high compared to the order of magnitude of the problem. However, although the error is small, the model found by SINDy does not describe the dynamics appropriately and does not generalize to future times. Indeed, we can see a significant deviation of predicted solution with respect to the ground truth in the extrapolation time window $[15, 60]$, with $T = 15$ being the training time, as we can see on Fig. 3.1c The ground-truth dynamics, as described above, is characterized by convex trajectories for the variable x that saturate over time. In contrast, as shown in Figures 3.1b and 3.1c, the trajectories estimated with SINDy, for some initial conditions, change concavity several times and do not tend to saturate completely over time. For this problem, SINDy shows to be an unreliable option.

For symbolic regression, it is important to consider the structural complexity of the problem. In addition to a complicated mathematical form, there are several constants that need to be estimated in the expression. To enhance the use of genetic programming, and therefore model discovery, the function set used is reduced to the minimum necessary, leaving out even functions that are realistically plausible for a saturation model, such as the log function. SR-T proves to be an extremely effective method: in both cases, the probability of model identification is relatively high. The RMSE is low and depends strongly on the accuracy with which the constants are estimated. In fact, genetic programming quickly identifies the right mathematical form, however it estimates the constants more slowly, only through random mutation. In terms of generalization and predictive capabilities, SR-T is effective as it frequently identifies the closed-form dynamical system correctly. D-CODE achieves comparable results. It should be noted that D-CODE's RMSE is lower because, in this case, the method converges more quickly toward the correct mathematical expression, estimating the constants with greater accuracy. However, when it struggles to find the constants, it tends to fit a slightly more complex closed-form model, while still maintaining excellent generalization and predictive capabilities.

Symbolic-SINDy extracts promising mathematical blocks during the genetic programming phase and finally adds the final solution to the set of potential blocks. Generally, the expressions that should be integrated into the CFL are short. In contrast, in this problem, to identify the dynamics correctly, the missing block that needs to be extracted is long, and it must be recovered entirely to properly describe the system. Observe that, for this reason, the optimal choice is to tune the filter so that the extracted building blocks are neither too short expressions nor excessively long. Note that the filter controls the number and the complexity of the potential building blocks considered. It plays a key role in the accuracy and efficiency of the proposed method, as overly complex expressions may overfit the dynamics, and a wide filter would lead to an increase in computational costs. Another

important observation is that, in this problem, it is necessary to estimate all coefficients with high accuracy. For this reason, it is required to take advantage of all possible generations of symbolic regression. This choice ensures that the most accurate blocks obtained during genetic programming can be integrated into the polynomial CFL. The results show that, by doing so, Symbolic-SINDy performs better than symbolic regression methods. The probability of success is higher in both cases. When correctly identified, the discovered models also have a smaller RMSE, thanks to compensation provided by the multiplicative coefficient estimated with sparse regression. Since for Symbolic-SINDy the genetic programming is used only in the first expression, the overall execution is less expensive than a complete symbolic regression estimation. The comparison with SINDy shows a clear improvement in performance. The incorrect identifications are mainly due to either low accuracy in the estimation of the coefficients or, alternatively, to a slight overfitting of the dynamics, where the selected block is slightly too complex. The risk of overfitting is caused by the high number of generations used, but its effect is mitigated by a proper tuning of the filter.

Comparison: SINDy vs. Symbolic-SINDy. The model identified by SINDy is compared with the best model identified with Symbolic-SINDy in Table 3.2. As highlighted previously, SINDy proves to be totally unsuited for a problem such as this one: equipped only with a polynomial library, the best it can do is extrapolate the optimal polynomial combination that is locally capable of explaining the observed data. In this way, SINDy does not describe the dynamics appropriately, neither generalize over time. However, it is worth noting that *a priori* it would be impossible to think of integrating the required block with high precision. Indeed, in addition to a complex nonlinear structure, there are several constants that must be estimated accurately. Symbolic regression is used for this purpose. With Symbolic-SINDy, by finding and automatically inserting the missing block into the CFL, the model is identified in closed-form, greatly *extending* the applicability of

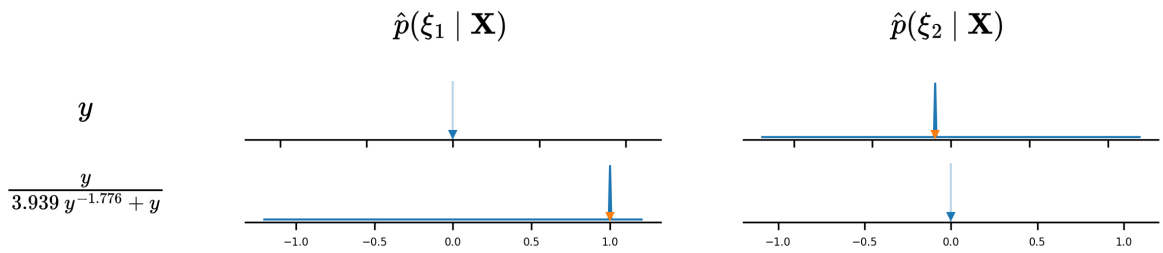
	SINDy	**Symbolic-SINDy**
–	$\dot{x} = -0.052 + 0.375\,y - 0.032\,y^2$ $\dot{y} = -0.100\,y$	$\dot{x} = 1.001\dfrac{y}{y + 3.939\,y^{-1.776}}$ $\dot{y} = -0.100\,y$
k	$\dot{x} = 0.026\,y - 0.111\,k + 0.357\,ky + 0.040\,k^2 - 0.032\,ky^2$ $\dot{y} = -0.100\,y$	$\dot{x} = 1.004\dfrac{k\,y}{y + 3.988\,y^{-1.764}}$ $\dot{y} = -0.100\,y$

Table 3.2: Comparison between SINDy and Symbolic-SINDy (with `seed=102`) for the Hill equation.

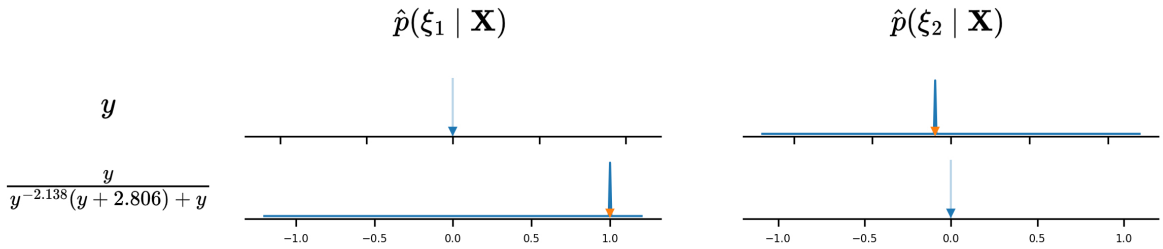
SINDy as a reliable method for automated model discovery. In this specific problem, the contribution provided by symbolic regression is decisive. The flexibility to identify characteristic nonlinearities allows us to automatically include all the necessary information into the library, thus enabling an effective recovery of the underlying dynamics.

A similar problem has already been addressed in [24]. In that case, Symbolic-SINDy has been used to identify a Gompertz-like model, and only a few generations of symbolic regression were employed. The substantial difference with that problem arises from the structural complexity of the Hill equation: estimating the mathematical structure of the equation and its constants accurately requires a greater number of iterations.

Uncertainty quantification on the discovered model. Through E-SINDy, it is possible to quantify the uncertainty of the discovered model, as well as the stability of the extrapolated building block. A bootstrap ensemble of 200 models is used. It can be seen in Figures 3.4a and 3.5a that, when the block integrated into the CFL is correct, both in form and in its parameters, the uncertainty associated with the SINDy coefficients is extremely low. As shown, in those cases, there are localized spikes around the true values of the coefficients, with an extremely low standard deviation, of the order of 10^{-4}. This analysis shows that, when SINDy's nonlinear library contains all the blocks necessary for model identification, the uncertainty associated with the coefficients estimated via sparse regression is low. It is interesting to note what happens when the added block is only partially correct, because the parameters of the problem are estimated with low precision. From Figure 3.5b, it can be seen that the resulting model is more uncertain. In this case, more spikes appear around different values of the coefficients, and there is greater variability, with a standard deviation of the order of 10^{-1}. This situation indicates that the added block is less informative because it is less stable, suggesting that the search should be refined. Finally, it is interesting to consider what happens when the identified block is incorrect but slightly overfits the dynamics. In the previous paragraph, it was highlighted how the high number of generations used in the extraction phase could lead to the identification of blocks that are slightly too complex but are not excluded by the filter. Those blocks might be preferred because they could have a lower RMSE value on the training data. Although incorrect, these blocks are very informative and very similar to the true one. Consequently, as shown in Figure 3.4b, the uncertainty associated with the SINDy coefficients is limited, with a standard deviation of the order of 10^{-3}. This result is specific to the problem addressed because many generations of symbolic regression are required, and it is realistically impossible to build a filter capable of removing nonlinearities that are slightly too complex.

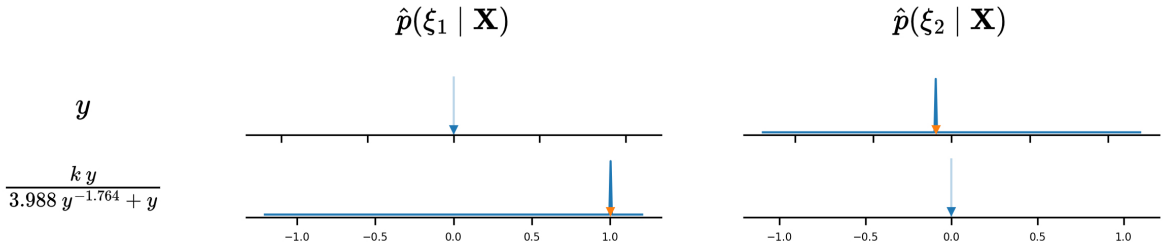


(a) Estimated coefficient distribution with `seed=102`: building block recovered.

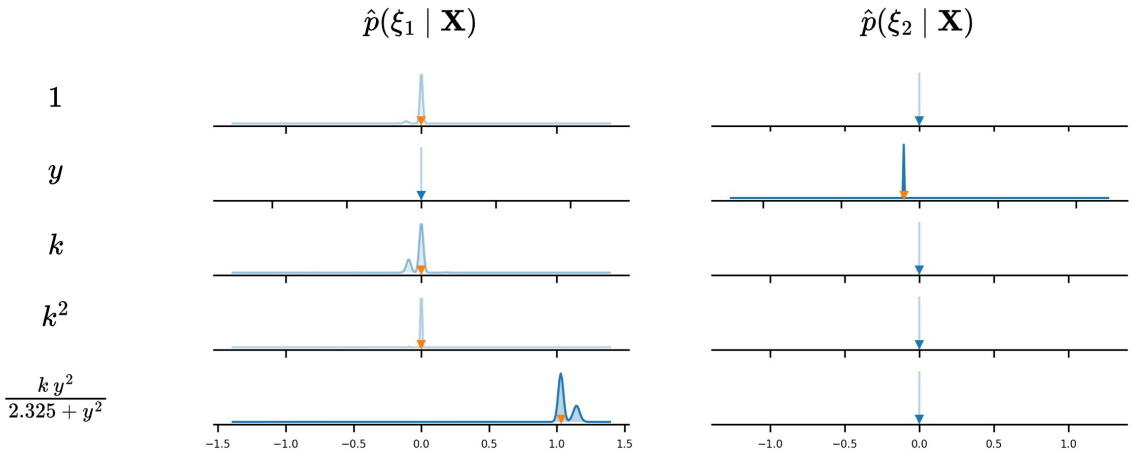


(b) Estimated coefficient distribution with `seed=101`: wrong building block.

Figure 3.4: Uncertainty quantification on Symbolic-SINDy identified model for the Hill equation.



(a) Estimated coefficient distribution with `seed=102`: building block recovered.



(b) Estimated coefficient distribution with `seed=101`: inaccurate parameter estimates.

Figure 3.5: Uncertainty quantification on Symbolic-SINDy identified model for the k-parametrized Hill equation.

3.2. Revised van der Pol oscillator

The van der Pol oscillator was first introduced by van der Pol to describe the oscillations in a vacuum-tube triode circuit used in early radio systems. A defining feature of this dynamics is that trajectories originating from different initial conditions converge to the same periodic orbit of finite amplitude [35]. In the presence of self-maintenance and amplitude stability, van der Pol proposed a nonlinear second-order differential equation to describe the phenomenon:

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0, \quad \mu \in \mathbb{R}^+,$$

where x is the state variable and μ controls the intensity of the nonlinearity and the damping. In particular, when μ is small, the observed oscillations are almost sinusoidal, while for larger values the system produces *relaxation oscillations*. This discovery was later applied to the modeling of heartbeat dynamics, marking the beginning of modern biophysics. In addition, the van der Pol equation has been simplified in the FitzHugh–Nagumo model to describe neuronal excitability. Much attention was also devoted to investigating the peculiar behavior of the Van der Pol oscillator in the presence of a periodic forcing [33].

We rewrite the van der Pol oscillator problem as a first-order dynamical system:

$$\dot{x} = \mu(1 - y^2)x - y + A\sin(\omega y^2), \tag{3.3}$$

$$\dot{y} = x, \tag{3.4}$$

where $A \in \mathbb{R}$, $\mu \in \mathbb{R}^+$, and $\omega \in \mathbb{R}^+$.

With this definition, Equation (3.3) explicitly introduces an additional nonlinearity into the van der Pol oscillator, described by a sinusoidal force composed with a quadratic power of the state. Note that the resulting dynamics is therefore defined by the contribution of two interacting oscillations. With a suitable choice of parameters, it is possible to ensure that the effects of the two oscillations partially overlap, making the resulting dynamics more complex to identify. Moreover, the introduced forcing is unconventional and would be difficult to be selected *a priori*. For these reasons, classical methods can face significant difficulties when attempting to identify the closed-form model from observations of such dynamics, justifying the use of a method capable of going beyond their limits.

3.2.1. Problem setup

In the numerical experiments, the initial conditions are sampled as $(x_0, y_0) \in [0, 1] \times [0, 1]$, and the 50 trajectories are simulated over a time horizon $T = 10$ using $N_t = 100$ time steps. The model parameters are fixed as $\mu = 1$, $A = 1$, $\omega = 1$. In the ω-parametrized setting, the parameter ω is allowed to vary within the range $\omega \in [1, \pi]$.

SINDy is run using a general polynomial CFL of the third degree, simulating a situation where no prior knowledge of the problem is available.

Symbolic regression is run using a maximum number of 20 generations. For the standard case the functions set is defined as $\mathcal{F} = \{+, -, \times, -1, \wedge, \sin, \cos\}$, while for the ω-parametrized problem it is reduced to $\mathcal{F} = \{+, -, \times, -1, \sin\}$, to simplify the identification. These specifications are the same for SR-T and D-CODE.

Symbolic-SINDy is run based on SR-T. For the standard case, the functions set is enlarged including also the log operation as $\mathcal{F} = \{+, -, \times, -1, \wedge, \sin, \cos \log\}$, using a maximum number of 6 generations. Also for the ω-parametrized problem, the functions set is enlarged including the log operation as $\mathcal{F} = \{+, -, \times, -1, \sin, \log\}$, using a maximum number of 11 generations. In this example, the function sets are deliberately selected as large as possible, with the aim of maximizing symbolic regression's flexibility and user-independence. The building blocks are then integrated into a general polynomial CFL of the third degree.

A detailed description of the tuning parameters used for all the methods is provided in Appendix A.2.

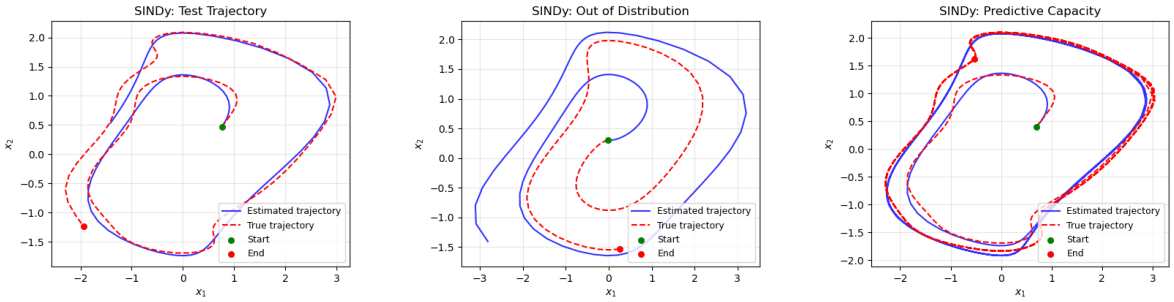
3.2.2. Numerical results

The numerical results obtained from the various experiments are summarized in Table 3.3. In addition, Figures 3.6, 3.7, and 3.8 are provided for support. A detailed version of the results for each experiment is provided in Appendix A.2.

Choosing $\mu = 1$, the observed oscillations of the van der Pol equation are almost sinusoidal. Adding a further sinusoidal term, for some points along the trajectory, this oscillation is amplified. When ω is small, the forcing contribution stretches the trajectory. In the ω-parametrized problem, when the frequency is allowed to be higher, this second source of oscillation is more evident.

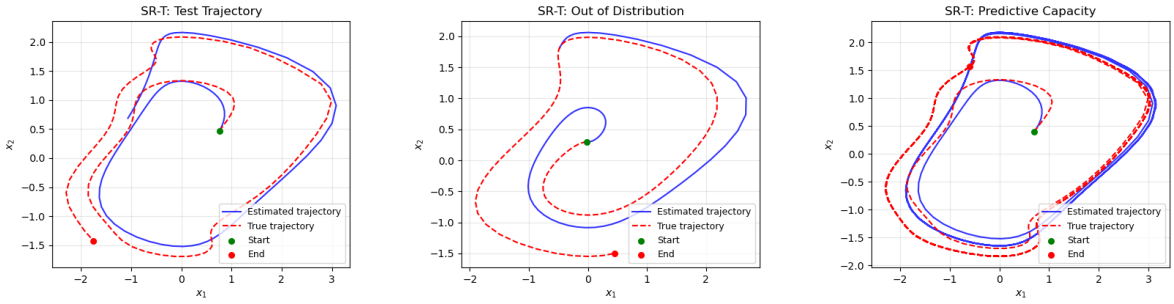
	Params.	Suc. Prob (3.3)	Suc. Prob (3.4)	RMSE	Time [m]
SINDy	–	0	1.00	0.534	$\simeq 0$
	ω	0	1.00	0.840	$\simeq 0$
SR-T	–	0.40	1.00	0.305	7.13
	ω	0.40	1.00	0.394	7.80
D-CODE	–	0.35	1.00	0.360	5.33
	ω	0.05	1.00	0.791	5.53
Symbolic-SINDy	–	0.80	1.00	0.124	2.27
	ω	0.40	1.00	0.555	4.83

Table 3.3: Performance comparison of the different model discovery methods for the revised van der Pol oscillator.



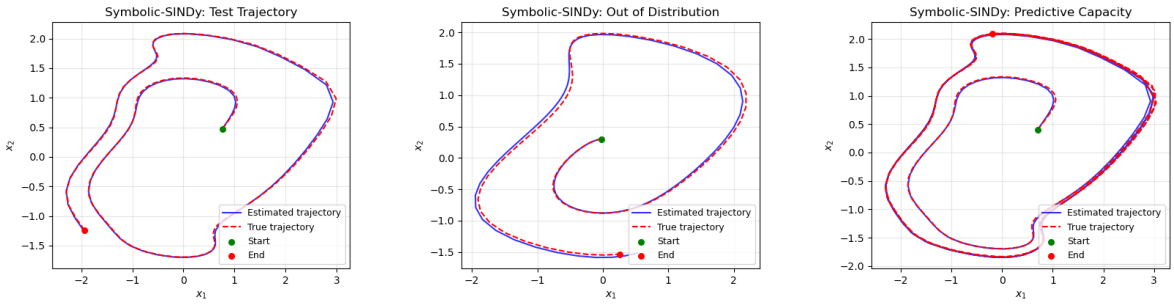
(a) $(x_0, y_0, \omega) = (0.7, 0.4, 2.5)$ (b) $(x_0, y_0, \omega) = (0.0, 0.3, 0.8)$ (c) $(x_0, y_0, \omega) = (0.7, 0.4, 2.5)$

Figure 3.6: SINDy for the ω-parametrized revised van der Pol oscillator. In particular, (a) shows the estimated dynamics starting from a point within the training set; (b) shows the estimated dynamics starting from a point outside the training set; (c) shows the estimated dynamics starting from a point within the training set, with extrapolation time window $[10, 40]$.



(a) $(x_0, y_0, \omega) = (0.7, 0.4, 2.5)$ (b) $(x_0, y_0, \omega) = (0.0, 0.3, 0.8)$ (c) $(x_0, y_0, \omega) = (0.7, 0.4, 2.5)$

Figure 3.7: SR-T for the ω-parametrized revised van der Pol oscillator, `seed=103`. In particular, (a) shows the estimated dynamics starting from a point within the training set; (b) shows the estimated dynamics starting from a point outside the training set; (c) shows the estimated dynamics starting from a point within the training set, with extrapolation time window $[10, 40]$.



(a) $(x_0, y_0, \omega) = (0.7, 0.4, 2.5)$ (b) $(x_0, y_0, \omega) = (0.0, 0.3, 0.8)$ (c) $(x_0, y_0, \omega) = (0.7, 0.4, 2.5)$

Figure 3.8: Symbolic-SINDy for the ω-parametrized revised van der Pol oscillator, `seed=103`. In particular, (a) shows the estimated dynamics starting from a point within the training set; (b) shows the estimated dynamics starting from a point outside the training set; (c) shows the estimated dynamics starting from a point within the training set, with extrapolation time window $[10, 40]$.

Using SINDy, model identification is partial. Indeed, since the CFL does not contain the required nonlinearity, SINDy correctly identifies only a few terms of the dynamics and uses additional nonlinearities to approximate the observed trajectory as closely as possible. In the ω-parametrized case, the best identified model uses almost all the terms in the library. Although it cannot identify the model in closed-form, SINDy still exhibits some important features. In general, it captures only the main behavior and cannot reconstruct the trajectory in its full complexity, as shown in Figure 3.6a. Moreover, the absence of the

required term in the library leads to overfitting on the training data. Figure 3.6b shows that, for some initial conditions, the trajectories obtained by evolving the SINDy model are very similar to those on which the model is trained, while being completely different from the true trajectories, indicating poor generalization. Nevertheless, the asymptotic behavior is recovered. As shown in Figure 3.6c, over time the trajectory converges to a limit cycle with an amplitude similar to the real one.

Symbolic regression methods also highlight critical issues in model identification. Both SR-T and D-CODE struggle to identify the true underlying dynamics, since the estimated probabilities of success are relatively low. SR-T proves to be slightly more robust. When it does not identify the correct model in closed-form, it often fits dynamics that combine x and y through a trigonometric expression. This highlights the difficulty of correctly identifying the overall oscillation, which is the result of two distinct and dependent contributions. SR-T performs better than SINDy, tending to model the curve within the training set in greater detail. It also shows better generalization, although in some simulations it exhibits the same issues, as shown in Figures 3.7a and 3.7b. SR-T also asymptotically reaches a limit cycle over time, but with less precision, as shown in Figure 3.7c. A similar behavior is observed for D-CODE. The main difference with respect to SR-T concerns the convergence speed. D-CODE seems to converge more slowly, potentially identifying completely incorrect dynamics, despite the many generations used. This behavior is even more evident in the ω-parametrized case. The difficulty in convergence further confirms the complexity of model identification and suggests that existing symbolic regression methods may act as weak estimators of the dynamics.

Symbolic-SINDy is provided by a wider functions set with the aim of maximizing symbolic regression's flexibility and user-independence. Moreover, the number of generations employed is lower, to reduce computational costs of genetic programming. In the standard case, the performance gain is evident: the probability of success increases significantly and, simultaneously, the time required is considerably reduced. This example highlights the power of the method, which is capable of performing better than both SINDy and symbolic regression, solving a problem that proves to be extremely complex. A similar argument applies to the ω-parametrized case, where the results obtained with Symbolic-SINDy are not worse than those obtained with symbolic regression. It is important to note that the performance of symbolic regression is achieved in a simplified context, where the functions set is limited to only the necessary functions; otherwise, the identification probabilities would have been close to zero. Symbolic-SINDy proves to be more flexible, identifying the correct model with the same frequency despite a less informed functions set. Given the slower convergence of genetic programming, it was necessary to slightly

increase the number of generations for this problem. Furthermore, in order to extract the decisive building block, the filter was enlarged, allowing more blocks to be extracted. These two operations increase the computational cost of Symbolic-SINDy, but still keep it lower than symbolic regression methods.

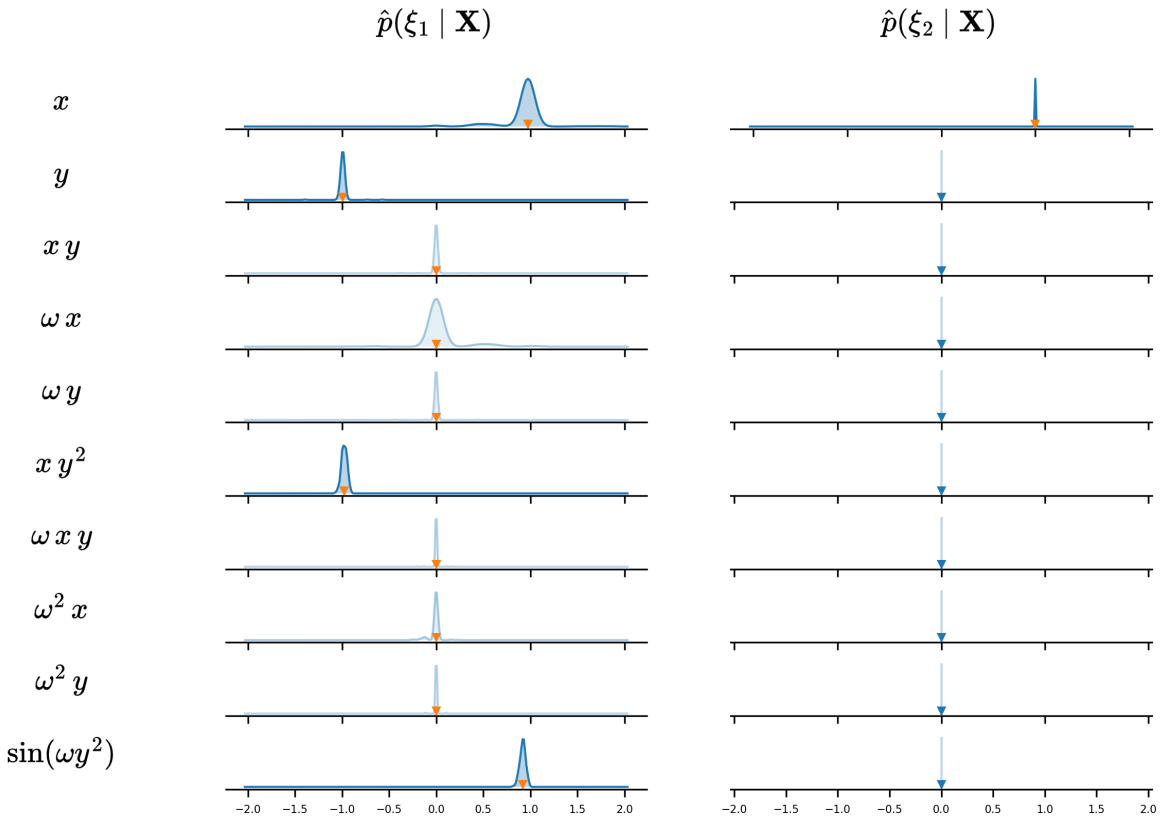
Comparison: SINDy vs. Symbolic-SINDy. The model identified with SINDy is compared with the best model identified with Symbolic-SINDy in Table 3.4. For this problem, the model identification with SINDy is only partial. Indeed, this method correctly identifies only a few terms of the dynamics and uses additional nonlinearities to approximate the observed trajectory as closely as possible. SINDy proves to be incapable of capturing the true complexity of the dynamics, simply because inside its CFL the nonlinearity characteristic of the system is missing. However, realistically, this term would hardly be taken into consideration *a priori*. Thanks to the automatic addition of this building block, Symbolic-SINDy succeeds in its intent, discovering the model correctly and with limited prior knowledge. In this problem, it is also interesting to note that both phases of Symbolic-SINDy, i.e., genetic programming for building block extraction and sparse regression, are decisive for the success of the experiment. Used separately, both methods prove ineffective for model discovery tasks. Indeed, if SINDy – given the incompleteness of the functions' library – can never correctly identify the underlying dynamics, symbolic regression on the other hand frequently converges to an incorrect model that combines x and y through a trigonometric expression. Symbolic-SINDy prevents the fit of a complex model by first extracting promising information and then regularizing with sparse regression. This results in a significant increase in performance, even in more complicated settings.

	SINDy	**Symbolic-SINDy**
–	$\dot{x} = 0.960\,x + 0.5357\,y^2 - 0.969\,xy^2 - 0.597\,y^3$ $\dot{y} = 1.000\,x$	$\dot{x} = 0.976\,x - 1.000\,y - 0.973\,xy^2 + 0.996\sin(y^2)$ $\dot{y} = 1.000\,x$
ω	$\dot{x} = 1.059 + 1.212\,x - 2.043\,y + 0.394\,\omega + \dots$ $\dot{y} = 1.000\,x$	$\dot{x} = 0.975\,x - 0.994\,y - 0.974\,xy^2 + 0.916\sin(\omega y^2)$ $\dot{y} = 1.000\,x$

Table 3.4: Comparison between SINDy and Symbolic-SINDy (with `seed=103`) for the revised van der Pol.

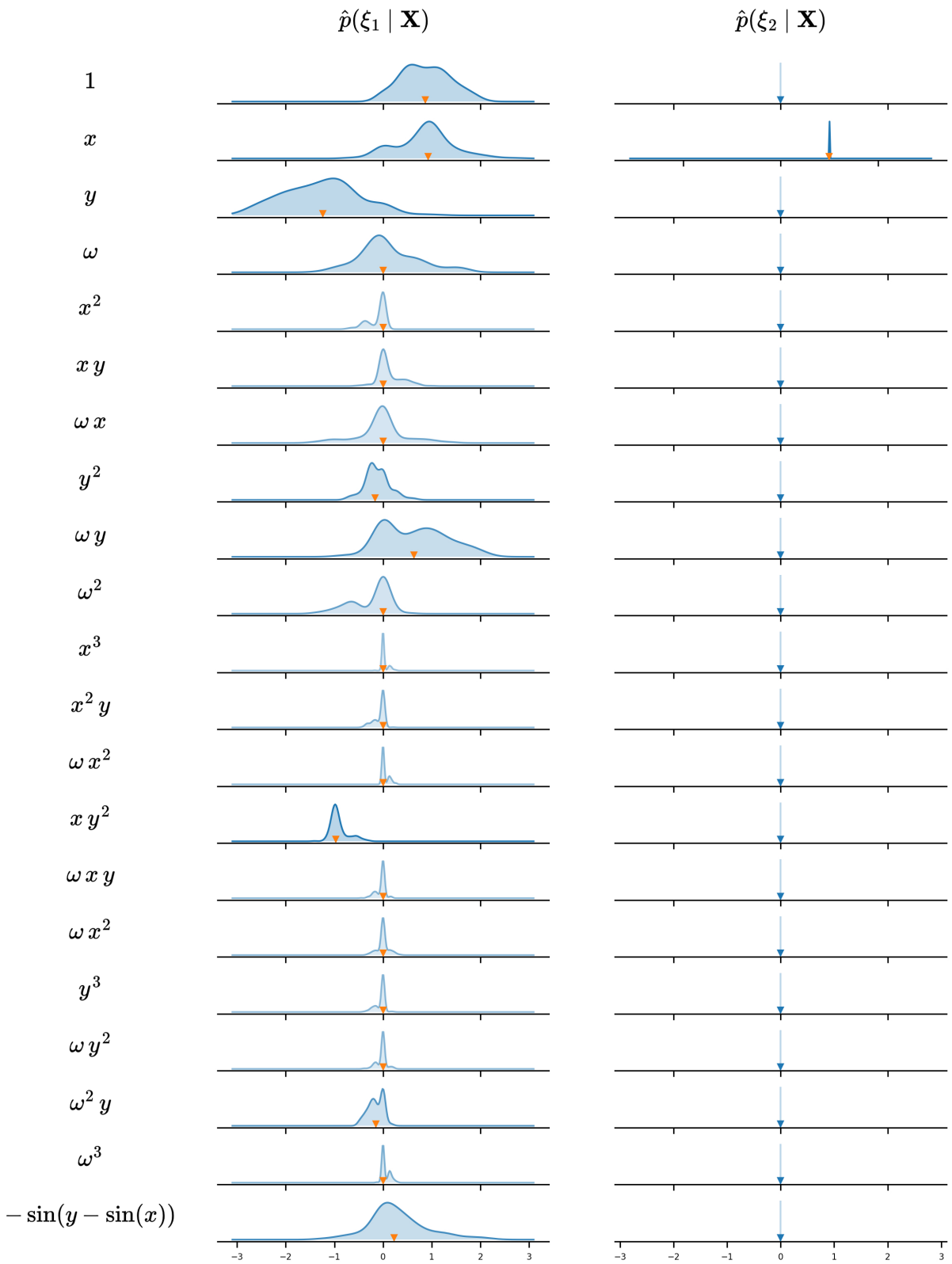
Uncertainty quantification on the discovered model. Through E-SINDy, it is possible to quantify the uncertainty of the discovered model, as well as the stability of

the extrapolated building block. A bootstrap ensemble of 200 models is used. It can be seen in Figure 3.9a that, when the block integrated into the CFL is correct, both in form and in its parameters, the uncertainty associated with the SINDy coefficients is low. As shown in Figure 3.9a, there are localized spikes around the true values of the coefficients, with a low standard deviation. This analysis shows that, when SINDy's nonlinear library contains all the blocks necessary for model identification, the uncertainty associated with the coefficients estimated via sparse regression is low. On the other hand, it is interesting to consider what happens when the identified block is incorrect. From Figure 3.9b, it can be seen that the resulting model is significantly more uncertain. In this case, the models in the ensemble are extremely different both in nonlinear terms used and in the associated coefficients. The high variability encountered indicates that the added building block is not informative, and therefore suggesting the need to search for another block. This result shows how the uncertainty quantification can be used as a support tool in the post-processing phase to assess the quality of the automatically integrated building block, providing further information to evaluate the reliability of the identified model.



(a) Estimated coefficient distribution with `seed=103`: building block recovered.

Figure 3.9: Uncertainty quantification on the Symbolic-SINDy identified model for the ω-parametrized van der Pol oscillator.



(b) Estimated coefficient distribution with `seed=109`: wrong building block.

3.3. Results analysis and comparative discussion

In the examples provided, Symbolic-SINDy proves to be an extremely valid method in the context of automated model discovery. Overall, the combination of symbolic regression and SINDy proves to be an effective approach for the purpose, overcoming the limitations associated with existing methods and enhancing close-form dynamics identification. The flexibility of symbolic regression is exploited to search within a large functional space for promising building blocks. These problem-specific nonlinear mathematical expressions automatically expand the SINDy library. Through sparse regression, it is finally possible to obtain a parsimonious estimate of the dynamics in a small functional space. In addition to the solid results obtained in [24], Symbolic-SINDy generally showed a higher recovery rate than the existing methods in the problems addressed, while also ensuring a reduction in execution times.

The first example we discussed is related with the Hill equation. In this case, the nonlinear system of equations has an extremely complex structural form, combining powers and fractions with different constants. Clearly, SINDy is unable to estimate the dynamics in closed form since it is difficult to include the entire expression required *a priori* in its library. Symbolic regression is suited to this problem, but requires the definition of a concise functions set since identification is very complicated. Fractional functions are indeed a class of expressions that are difficult to estimate symbolically. Symbolic-SINDy enhances the performance of symbolic-regression methods, reducing execution times: genetic programming is performed only with respect to the first equation. To achieve these results, it is necessary to take advantage of symbolic regression in all its generations. This choice is mainly motivated by the need to extract the block with high precision. Indeed, in addition to the complex mathematical expression to recover, all constants must also be estimated accurately. A second consideration concerns the block extraction process: in fact, the block to be recovered is long. To ensure the extraction of a block of that complexity the optimal choice is to tune the filter so that the extracted building blocks are neither too short expressions nor excessively long. The filter controls the number and the complexity of the potential building blocks considered. It plays a key role in the accuracy and efficiency of the proposed method, as overly complex expressions may overfit the dynamics, and a wide filter would lead to an increase in computational cost. The critical problem that can arise when a high number of generations is combined with a filter that selects long expressions, is the extraction of blocks that slightly overfit the observed data. This risk can be mitigated by properly defining the filter, but realistically it could be difficult to construct a perfect filter with limited prior knowledge. One possible

solution to the problem of overfitting is to use a slightly lower number of generations. This choice would still result in a significant recovery rate, albeit with a less accurate estimate of the Hill equation parameters. Overall, this example shows how Symbolic-SINDy proves to be an effective method when faced with a complex problem and, moreover, that sometimes a strong bias from symbolic regression is required to enhance the identification.

The second experiment highlights the potential of Symbolic-SINDy. In the chosen example, the revised van der Pol oscillator, the dynamics are characterized by two interacting oscillatory contributions: one is a nonlinear polynomial combination and the other is a trigonometric composite function. Both SINDy and symbolic regression prove to be ineffective. Typically, it would be unthinkable to include a composition between power and sine function in the SINDy library without prior knowledge, especially if the pulsation is not unitary. Symbolic regression also shows problems in correctly distinguishing the two contributions in the dynamics. In the ω-parametrized case, the functions set was reduced to facilitate identification, otherwise the recovery probability would have been really low. In contrast, Symbolic-SINDy, although equipped with a larger functions set, achieved a higher recovery rate and a reduction in execution times, with a clear gain especially in the unparametrized case. In this sense, the flexibility of genetic programming is exploited to the fullest, searching in a larger set of allowed operations, emulating a situation where prior knowledge is truly limited. In this experiment, a low number of generations is sufficient to identify the decisive block and automatically integrate it into the SINDy library. In the ω-parametrized case, identification was considerably more challenging, both at the symbolic regression level and at the Symbolic-SINDy level. As explained above, in both cases it was therefore necessary to restrict the functions set, even though the configuration used for Symbolic-SINDy still contains a larger set of expressions, demonstrating the greater flexibility of the method. Furthermore, in this case, the filter was also slightly relaxed to facilitate the extraction of the missing block, thus increasing the execution time. This choice was made because, during genetic programming, even if the block is identified, it is not always extracted because it is recovered in an equivalent but longer mathematical form, due to the parity of the sin function. Increasing further the complexity of the problem, allowing for more parameters to vary, or decreasing the impact of the sine forcing, lowering the driving frequency and amplitude, all methods proved to be substantially ineffective. In those cases also Symbolic-SINDy showed poor performance. Overall, this example shows that Symbolic-SINDy is an effective method, significantly increasing the identifiability of the model where classical methods fail, while maintaining a low computational cost.

In addition, the integration of a UQ framework proves useful. Indeed, by simply using an

E-SINDy after the block identification, it is possible to qualitatively access the stability of the model found and the uncertainty associated with its coefficients. As shown in the various situations proposed, when the block found is correct, great stability is observed. The empirical distributions associated with the coefficients have a low standard deviation, and few blocks have a significant inclusion probability. Conversely, greater uncertainty is observed when the block integrated into the library is incorrect or partially correct. In these cases, indeed, there are more spikes located at different values for the same coefficient, generally a higher standard deviation, and many terms have a high inclusion probability. A situation of this type suggests that the model found is unstable and, therefore, that the automatically added block is not informative for the purposes of the problem. A particular situation is shown in Figures 3.4b, where, although the block is mathematically incorrect, the uncertainty associated with the identified model is very low. This is in fact a special case, where the extracted block, although mathematically incorrect, describes the dynamics very well. As explained above, this block slightly overfits the dynamics but is expressive enough to capture the observed behavior.

In conclusion, Symbolic-SINDy proves to be an effective method in the context of automated model discovery, even in more complicated scenarios. In particular, the use of symbolic regression as a preliminary step to extract nonlinearities proves decisive in extending the applicability of SINDy, offering a concrete and efficient solution for this purpose. Together with the examples proposed in [24], the method has been tested on several examples, of different structure and complexity, systematically performing better than the existing methods. However, it is important to note that, as the complexity of the problem increases, in order for the performance to be significant, it is necessary to restrict the structure of the functions set of genetic programming. This is clearly due to the fact that the dynamics are more complex to identify and, therefore, symbolic regression needs to be simplified in order to remain effective. This inevitably impacts the practical applicability of Symbolic-SINDy, which requires at least minimal prior knowledge to be transferred inside the functions set for complex problems. The idea of using an auxiliary method to automatically extract information that is difficult to include *a priori* remains very appealing, and this implementation succeeds in this purpose.

4 | Online Symbolic-SINDy for time-varying systems

This chapter extends the framework presented in Chapter 2 by adapting Symbolic-SINDy to the case of time-varying dynamical systems. Many existing methods struggle to properly handle situations in which abrupt changes in dynamics occur due to variations of the underlying system over time. In particular, critical issues arise when new terms are introduced or existing ones are eliminated. The following section shows how symbolic regression and sparse identification can be effectively combined for online learning of the system in the case of on-the-fly variations in the system's behavior.

4.1. Online learning framework

In the context of online control of a dynamical system, it is necessary to define a method that, while a data stream is being acquired, is able to autonomously identify any change in the underlying dynamics and, consequently, update the estimated model. Due to its simplicity, Symbolic-SINDy can potentially be used for this purpose in its online version. In particular, it can be extended to the identification of time-varying systems by explicitly including time dependence directly in the library, through the addition of nonlinear functions of time. These nonlinearities can be defined *a priori* or added automatically through block extraction during the execution of symbolic regression. Furthermore, Symbolic-SINDy can be coupled with an appropriate change-point detection algorithm to locally estimate dynamics and adapt the model to sudden abrupt changes in system behavior. Formally, **Online Symbolic-SINDy** models the evolution of the system as follows:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t); \boldsymbol{\beta}) \approx \boldsymbol{\Theta}(\mathbf{x}, \boldsymbol{\beta}, t)\, \boldsymbol{\Xi}(t).$$

It is important to note that Online Symbolic-SINDy is essentially an extension of SINDy, in which both the CFL $\boldsymbol{\Theta}(\mathbf{x}, \boldsymbol{\beta}, t)$ and the coefficient matrix $\boldsymbol{\Xi}(t)$ are time-dependent. This allows the output model to change according to the ground-truth system. Furthermore,

since temporal variations in the system often occur in a nonlinear manner, the flexibility of symbolic regression can be exploited to identify promising temporal nonlinearities, which may then be integrated into the function library.

In principle, there are several change-point detection methods that can be used. Among them, a very simple and widely used method is the **Cumulative Sum (CUSUM)** [13]. CUSUM is a statistical technique that is used to detect deviations in the mean level of a process over time. In particular, in this context, the mean level of model error is monitored to detect changes in the underlying system. As the name suggests, the algorithm works by computing the cumulative sum of deviations of the observed error values from a reference value. When this statistic exceeds a predefined threshold, a potential change in the process is detected. Additionally, to promptly react to large initial errors occurring during the baseline calibration phase, a secondary threshold is introduced to detect an early change-point.

Online Symbolic-SINDy works as follows. In the first phase, the system is observed for a short interval, and its dynamics is identified using SINDy, equipped only with a polynomial library in states and time. This operation efficiently defines a good initial estimate of the dynamics. From this point onward, the CUSUM algorithm monitors in the background how much the current estimated model deviates from the measurements that are acquired over time. When a change-point is detected, a search for promising nonlinearities is performed locally using a symbolic regression method. This operation allows for further potential nonlinearities to be extracted, both in state and time, which could be the cause of the variation in dynamics. The extracted building blocks are then saved. At this point, a new model with new coefficients is locally identified using SINDy, this time equipped with an augmented nonlinear library combining the polynomial library with additional extracted blocks. In the following moments, the stability of the new model is evaluated by running SINDy again if needed, always informed of the saved blocks. It is important to note that it is not necessary to search for additional blocks at every change-point detected, especially if this search has just been performed. The system continues to monitor according to this logic, searching for additional blocks using symbolic regression when necessary, and using these nonlinearities to enlarge the SINDy library.

4.2. Numerical experiments

Among the most interesting test cases in the context of online control of time-varying systems are those in which new terms are introduced on-the-fly. This implies that the system evolves differently than initially estimated, and recovering the missing term that

was not initially included in the model might be challenging. A further complication can be obtained by varying the parameters and coefficients of the system over time. In this way, the impact of each component will change as the dynamics progress, continuously altering the behavior of the system.

Benchmark models. Online Symbolic-SINDy is tested against two problems:

1. forced van der Pol oscillator;
2. generalized Lotka-Volterra model.

Measurement settings. For both problems, the following settings are used:

- Trajectory generation: each experiment is conducted by simulating $N = 50$ trajectories over the time interval $[0, T]$, each initialized from a randomly sampled initial condition;
- Sampling frequency: each trajectory is sampled at a frequency of 20 observations per time unit, corresponding to a temporal discretization $dt = 0.05$;
- Noise contamination: the data are corrupted with additive white noise at a 1% level. Specifically, zero-mean Gaussian noise with standard deviation $\sigma = \sigma_R \,\mathrm{std}(\dot{\mathbf{X}})$ is added to the observations, where $\sigma_R = 0.01$ and $\mathrm{std}(\dot{\mathbf{X}})$ denotes the standard deviation of the noise-free data.

Evaluation metrics. To evaluate the performance of the method, the identified closed-form models over time and the corresponding RMSE are considered.

4.2.1. Forced van der Pol oscillator

As discussed in Section 3.2, the van der Pol oscillator is an interesting nonlinear dynamical systems because of its properties and applications. In particular, much attention was also devoted to its peculiar behavior in presence of periodic forcing. Consider the forced van der Pol oscillator written as a first-order dynamical system:

$$\begin{aligned} \dot{x} &= \mu(1 - y^2)x - y + A\cos(\omega t), \\ \dot{y} &= x, \end{aligned}$$

where $A \in \mathbb{R}$, $\mu \in \mathbb{R}^+$, $\omega \in \mathbb{R}^+$, and $t \in \mathbb{R}^+$. A fascinating phenomenon of this system is that, for an appropriate choice of parameters (typically $\mu \gg 1$), it evolves at steady state with stable oscillatory dynamics, but with a response frequency different from the forcing

frequency. This phenomenon is known as *frequency demultiplication* (or subharmonic response), in which the system responds by oscillating stably at a frequency that is an integer fraction of the external driving frequency. The global behavior of the solutions of the forced van der Pol oscillator depends on the choice of the parameters μ, A, and ω. Variations in the forcing amplitude A or in the forcing frequency ω may induce qualitative changes in the system dynamics. In particular, when A varies, two different situations can be distinguished. For values of the forcing amplitude below a critical threshold, the system is mostly characterized by periodic orbits with periods that are odd multiples of the driving period, because of frequency demultiplication. When the forcing amplitude becomes sufficiently large and reaches a critical value, the period of the solution becomes the driving one [33].

Problem setup. Consider the following time-varying forced van der Pol system:

$$\dot{x} = \mu(1 - y^2)x - y + A(t)\cos(\omega t), \tag{4.1}$$

$$\dot{y} = x, \tag{4.2}$$

where $\mu = 5,\ \omega = \pi,\ A(t) = 0.5t$ and $t \in \mathbb{R}^+$.

The initial conditions are sampled as $(x_0,\, y_0) \in [0, 2] \times [0, 2]$, and the system is simulated over a time horizon $T = 70$ using $N_t = 1400$ time steps. In this example, the behavior of the system when the parameter A varies is emphasized. In particular, here the forcing amplitude $A(t)$ linearly depends on time t.

Although the nonlinear forcing term that characterizes the dynamics is present in the model form the beginning, initially, when t is small, its effect on the trajectories is almost negligible or may be indistinguishable from noise. As time increases, the effect of the forcing term becomes more evident, and it is responsible for the frequency demultiplication effect. Finally, once the critical value is reached ($t \approx 42$), the system passes through a frequency-locking bifurcation, and the solution period coincides with the driving period. An example of the dynamics is shown in Figure 4.1.

The system is initially observed in the interval $[0, 15]$, and the dynamics are estimated using SINDy with polynomial CFL of degree 3. SR-T is employed as symbolic regression method. When called, symbolic regression is performed on a local time window of 10 instants, using 11 generations and function set $\mathcal{F} = \{+, -, \times, -1, \cos\}$. When called, SINDy is performed on a local time window of 15 instants. A detailed description of the tuning parameters used for all the methods is provided in Appendix B.1.

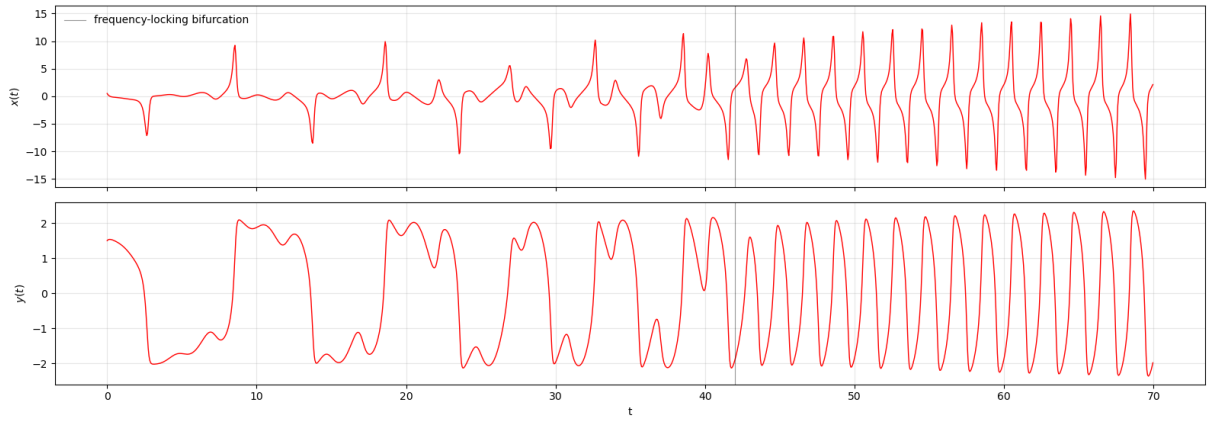
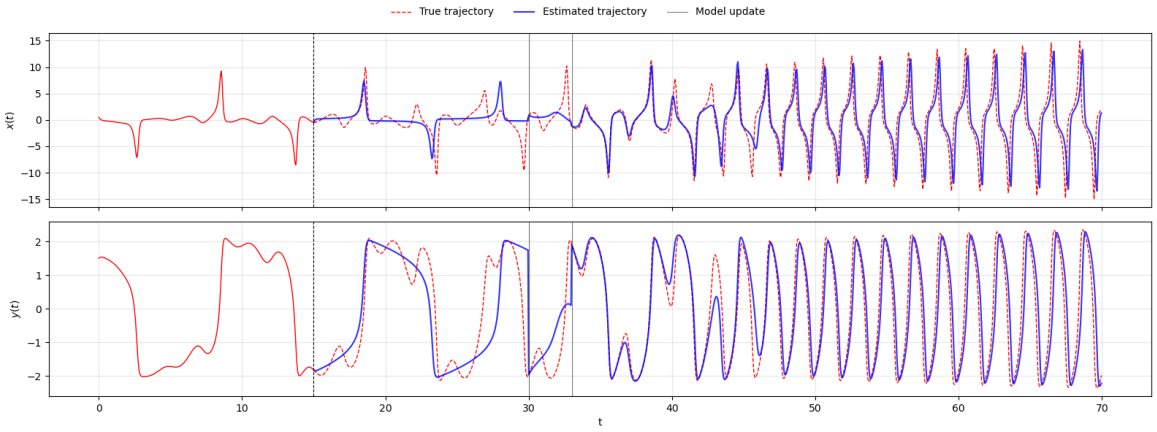


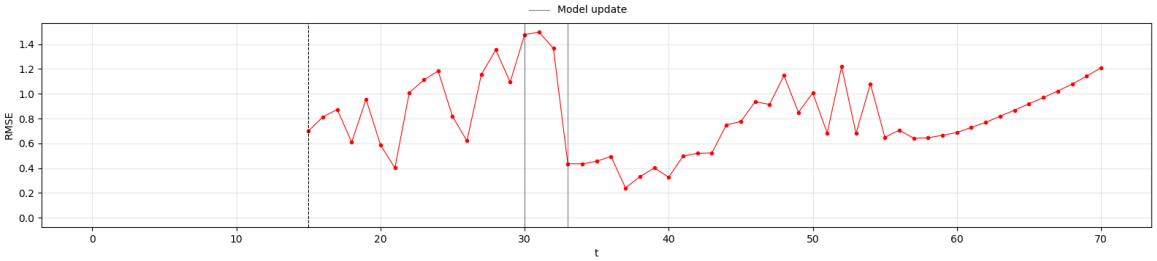
Figure 4.1: Noise-free ground-truth trajectories of the forced van der Pol oscillator.

Results. The results of the online experiment are shown in Figure 4.2 and Table 4.1. The total duration of the experiment is 47m53s, of which only 4m44s are used by symbolic regression for the extraction of promising build blocks.

The first identified model is obtained simply by applying SINDy to the 50 noisy trajectories in the time interval $[0, 15]$. SINDy captures the dynamics observed up to that point, essentially returning an unforced van der Pol oscillator. The identification is therefore only partial, and there are two reasons for this estimate. First, the considered function library does not contain the required nonlinearity, therefore, it cannot properly capture the dynamics in their complexity. Furthermore, the amplitude of the periodic forcing over the time interval is small, and its impact on the dynamics is marginal: although small peaks are noticeable, they can easily be mistaken for noise. After the calibration phase, at time $t = 30$, CUSUM detects the mismatch between the estimated model and the observed dynamics, which triggers the execution of symbolic regression. This operation extracts several potential blocks, including an excellent estimate of the required nonlinearity: $\cos(3.146\,t)$. SINDy is therefore potentially equipped with a complete CFL for the model discovery scope. Although SINDy can now correctly identify the model in closed-form, it initially fits the wrong model. Indeed, the estimated model with the correct library is discarded because of its excessive complexity and, therefore, the best acceptable model is returned. However, due to persistent instability in the fitted model, at $t = 33$, another update is made. Although the model found in this new call does not exactly match the closed-form ground-truth model, it captures all the terms that appear in the dynamics with sufficient accuracy, in particular, it correctly identifies the forcing term whose amplitude varies over time $t\,\cos(\pi\,t)$. An additional term appears in the estimated dynamics, which locally compensates for the numerical errors of the estimated building block. Initially, the model found proves to be stable and correctly undergoes the



(a) Online dynamics identification



(b) RMSE evolution between estimated model and acquired data.

Figure 4.2: Online model identification of the forced van der Pol oscillator with Symbolic-SINDy.

frequency-locking bifurcation between $t = 41$ and $t = 42$, as shown in Figure 4.2a. Right after the bifurcation, the error suddenly increases due to the contribution of the additional term. This leads to a search for a better model at $t = 48$, which is not found. As time increases, the contribution to the dynamics of the additional term becomes negligible and the model found is stably aligned with the ground-truth dynamics. The stability achieved can also be observed from the RMSE trend, reported in Figure 4.2b, where the error stabilizes over time, growing steadily in accordance with the growth in order of magnitude of the system.

In this experiment, it is possible to appreciate how SINDy, when provided with all the necessary terms, is an effective method for the online identification of a dynamic system. For this scope, symbolic regression plays a decisive role in automatically recovering missing terms, which would have been difficult to include in the SINDy library *a priori*. In conclusion, Symbolic-SINDy proves to be an effective method for the online control of a dynamic system, thanks to its ability to locally estimate dynamics, autonomously recover and integrate missing nonlinearities, and adapt over time at a relatively low cost. This makes it a flexible tool that can be widely used in online learning and control contexts.

Time	Identified model
$t = 15$	$\dot{x} = 4.725\,x - 1.223\,y - 4.695\,xy^2$ $\dot{y} = 1.006\,x$
$t = 30$	$\dot{x} = -1.055 - 0.852\,y + 0.567\,y^2 - 1.000\,xy^2 - 0.374\,y^3 - 1.007\cos(3.691\,y)$ $\quad - 0.506\,y\cos(3.691\,y) - 1.743\,xy^2\cos(3.691\,y)$ $\dot{y} = 1.006\,x$
$t = 33$	$\dot{x} = 5.214\,x - 1.548\,y - 4.557\,xy^2 + 0.425\,t\cos(3.146\,t) + 0.814\,xy\cos(3.146\,t)$ $\dot{y} = 1.005\,x$

Table 4.1: Online model identification process of the forced van der Pol oscillator with Symbolic-SINDy.

4.2.2. Non-autonomous Lotka-Volterra model

We now consider a second test case, the Lotka-Volterra model, that was independently introduced by Lotka and Volterra and describes the population dynamics of two interacting species, namely a prey and a predator. The dynamics of this system is defined by two coupled nonlinear differential equations:

$$\dot{x} = \alpha x - \beta xy, \tag{4.3}$$

$$\dot{y} = \delta xy - \gamma y, \tag{4.4}$$

where $\alpha \in \mathbb{R}^+$, $\beta \in \mathbb{R}^+$, $\gamma \in \mathbb{R}^+$, and $\delta \in \mathbb{R}^+$.

In Equations (4.3) and (4.4), x and y are respectively the amount of preys and predators in a given region. In this formulation, α and γ are their per-capita rates of change in the absence of each other, and β and δ are their respective rates of change due to interaction [1]. This model is widely applied to describe biological systems as it adequately captures the population oscillations characteristic of predator-prey ecologies. In its standard formulation, the parameters α, β, γ, and δ are fixed. However, in a more realistic scenario, the environment is not stationary, and the system may be driven by exogenous effects, such as seasonality and natural cycles [20]. To address these situations, the model can be extended into a non-autonomous system, where parameters are expressed as time-dependent functions.

Problem setup. Consider the following non-autonomous Lotka-Volterra model:

$$\dot{x} = \alpha(t)x - \beta(t)xy, \tag{4.5}$$

$$\dot{y} = \delta(t)xy - \gamma(t)y, \tag{4.6}$$

where

$$\alpha(t) = 1 + 0.4\sin(5t)\mathbb{1}_{\{t>20\}}(t), \quad \beta(t) = 0.1 - 0.01\mathbb{1}_{\{t>50\}}(t),$$

$$\gamma(t) = 1.5, \quad \delta(t) = 0.075, \quad t \in \mathbb{R}^+.$$

The initial conditions are sampled as $(x_0, y_0) \in [5, 20] \times [5, 20]$, and the system is simulated over a time horizon $T = 80$ using $N_t = 1600$ time steps. This test case is inspired by a problem addressed in [26]. Initially, the system evolves as a stationary Lotka-Volterra system. Subsequently, at $t = 20$, a sinusoidal forcing is introduced that acts on the coefficient α, simulating a seasonal effect on the rate of change of the prey. This nonlinearity must be recovered, as it is not initially observed. Finally, at $t = 50$, the interaction coefficient between prey and predator is slightly modified through a switching mechanism, introducing a transient regime into the system. An example of the dynamics is shown in Figure 4.3.

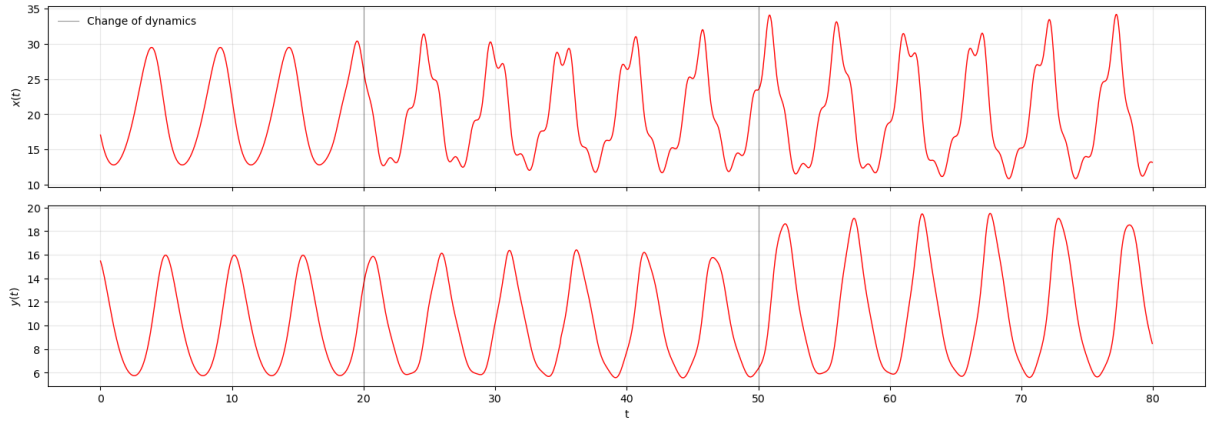
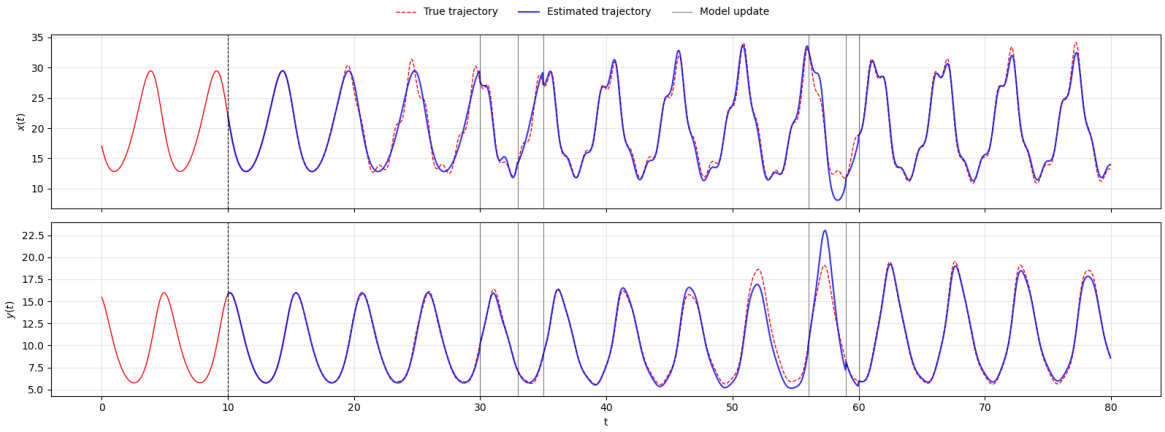


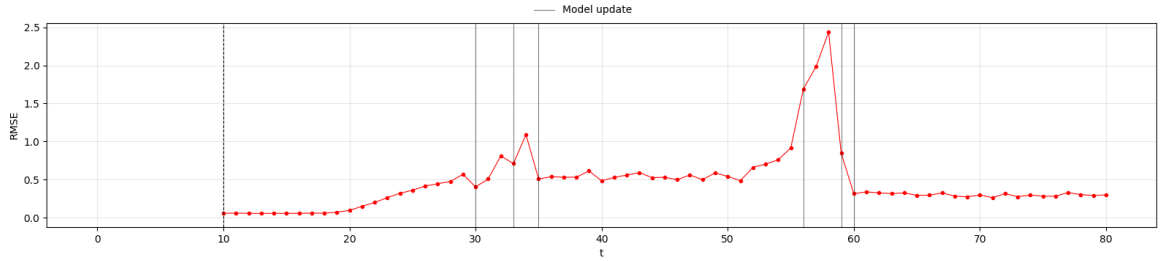
Figure 4.3: Noise-free ground-truth trajectories of the non-autonomous Lotka-Volterra model.

The system is initially observed in the interval $[0, 10]$, and the dynamics are estimated using SINDy with polynomial CFL of degree 2. D-CODE is employed as symbolic regression method. When called, symbolic regression is performed on a local time window of 10 instants, using 14 generations and function set $\mathcal{F} = \{+, -, \times, -1, \sin\}$. When called, SINDy is performed on a local time window of 10 instants. A detailed description of the tuning parameters used for all the methods is provided in Appendix B.2.

Results. The results of the online experiment are shown in Figure 4.4 and Table 4.2. The total duration of the experiment is 31m27s, of which only 5m02s are used by symbolic regression in two different calls for the extraction of promising build blocks.



(a) Online dynamics identification



(b) RMSE evolution between estimated model and acquired data.

Figure 4.4: Online model identification of the non-autonomous Lotka-Volterra model with Symbolic-SINDy.

The first identified model is obtained simply by applying SINDy to the 50 noisy trajectories in the time interval $[0, 10]$. The starting Lotka-Volterra model is hence immediately correctly estimated, since a simple polynomial library function is sufficient to recover the ground-truth dynamics. The change in system behavior occurs at $t = 20$, with the introduction of a sinusoidal external forcing term into the growth rate of the prey α, thus simulating a seasonal effect. The CUSUM detects this change at $t = 30$, and the symbolic regression is executed. Among the promising building blocks extracted, a good estimate of the missing nonlinearity is recovered: $\sin(5.007\,t)$. Right after the building block extraction, the first new identified model is unstable, even if it is partially correct. Although SINDy is equipped with a complete CFL for the problem, the estimation is influenced by a local transitional phase. Indeed, almost all the terms of the system are estimated with high accuracy by the model, but not the required nonlinearity: $x\,\sin(5\,t)$. In this case, the observed oscillations are mainly approximated by $\sin(5007\,t)$, neglecting the in-

Time	Identified model
$t = 10$	$\dot{x} = 0.999\,x - 0.100\,xy$ $\dot{y} = -1.497\,y + 0.075\,xy$
$t = 30$	$\dot{x} = -0.202 + 1.013\,x - 0.100\,xy + 1.819\,\sin(5.007\,t) + 0.174\,t\,\sin(5.007\,t)$ $\dot{y} = -1.498\,y + 0.075\,xy$
$t = 33$	$\dot{x} = -0.099 + 1.028\,x - 0.102\,xy + 0.265\,\sin(t)$ $\dot{y} = -1.498\,y + 0.075\,xy$
$t = 35$	$\dot{x} = -0.401 + 1.031\,x - 0.101\,xy + 0.264\,\sin(5.007\,t) + 0.324\,x\sin(5.007\,t)$ $\dot{y} = -1.497\,y + 0.075\,xy$
$t = 56$	$\dot{x} = 0.393 - 1.861\,x + 0.383\,t + 3.195\,\sin(5.007\,t) + 0.356\,x\sin(5.007\,t) - 0.072\,t\sin(5.007\,y)$ $\dot{y} = -1.498\,y + 0.075\,xy$
$t = 59$	$\dot{x} = -1.000 + 1.068\,x + 0.088\,y - 0.097\,xy - 0.091\,\sin(\sin(\sin t))$ $\dot{y} = -1.497\,y + 0.075\,xy$
$t = 60$	$\dot{x} = 0.186 + 0.993\,x - 0.090\,xy + 0.228\,\sin(4.997\,t) + 0.332\,x\,\sin(4.997\,t)$ $\dot{y} = -1.497\,y + 0.075\,xy$

Table 4.2: Online model identification process of the non-autonomous Lotka-Volterra model with Symbolic-SINDy.

teraction with x. CUSUM immediately detects the instability of the discovered model and attempts new updates. Only at $t = 35$ the new fitted model is close enough to the ground-truth system. With this update, all the terms of the dynamics are properly identified, even the characteristic nonlinearity $x\sin(5\,t)$. In this model, two additional terms also appear; however, their impact is only marginal, as they compensate for numerical errors in the building block estimation. This model proves his stability, remaining aligned with the observed dynamics for future intervals and maintaining a steady low error, as shown in Figures 4.4a and 4.4b. The second change in the dynamics occurs at $t = 50$ and does not introduce additional nonlinearities, but rather a sudden change in the value of a coefficient. When CUSUM intervenes, between $t = 56$ and $t = 59$, SINDy has a sufficiently informative CFL, but it does not identify the correct model. These failures, therefore, are not due to a lack of nonlinearities in the SINDy library, but rather to the fact that the models are estimated within a local time window containing the transition phase between the two dynamical systems. After several failed attempts, the execution of symbolic regression is triggered again, at $t = 60$, and further promising building blocks are extracted, including $\sin(4.997\,t)$. With this update, the new model obtained with SINDy

adequately describes the dynamics. As before, all terms of the system are recovered with high accuracy. Also in this case two additional terms appear to compensate for numerical errors in the building block estimation. As shown in Figures 4.4a and 4.4b, the found model is stably aligned with the ground-truth dynamic for future intervals and maintains a steady low error over time.

In this experiment, the combination of symbolic regression and SINDy proves to be effective in the context of online control of dynamical systems. In particular, it is highlighted how the method is capable of detecting changes in dynamics, recovering characteristic nonlinearities, and correctly identifying the true underlying system in the long term. It is worth noting that the second execution of symbolic regression would not have been necessary for the purposes of the experiment, since a good approximation of the missing nonlinearity was already available, but rather shows how it might be possible to retrieve other informative blocks at a later stage. Once again, Symbolic-SINDy proves to be a method that combines flexibility, accuracy, and low computational costs to locally estimate dynamics in online learning and control problems.

4.3. Online performances and applicability

These experiments yielded significant results regarding the use of Symbolic-SINDy for online learning and control problems in dynamic systems.

Indeed, the two proposed examples highlight the strengths of the method when faced with complex dynamical systems. As discussed above, although widely used for online problems, the SINDy method alone would not have been sufficient for a complete description of the systems tested. Ideally, coupling with the change-point detection method would have allowed SINDy to update its coefficients over time. However, the use of a static polynomial CFL would only identify local approximations of the dynamics. The estimated models would have low predictive capacity, which would have led to significant instability, resulting in a completely unsuitable method.

To overcome the natural weaknesses shown when dealing with online problems, SINDy can be coupled with an extended Kalman filter [25, 26], resulting in a setup (SINDy-EKF) that is ideally very practical in data assimilation contexts, allowing for the simultaneous update of status and parameters during the dynamics evolution. However, in the two scenarios presented, its use would have been ineffective for two reasons. Firstly, SINDy-EKF fails to recover terms that are not present in the starting model, unless explicitly specified otherwise. In a problem such as the forced van der Pol oscillator, SINDy-EKF would not have been able to update itself with respect to the initially unobservable external forcing.

Secondly, the method is particularly suited to slow variations in coefficients over time. In contrast, the proposed test cases present very sudden and rapid variations in dynamics, which would be challenging for a Kalman filter to track accurately.

The combination with symbolic regression, on the other hand, proves to be a clever choice. Thanks to the flexibility of the latter, it is possible to identify characteristic nonlinearities in the system under observation. The local extraction of promising building blocks provides SINDy with a dynamic CFL automatically constructed in relation to the problem. Furthermore, the use of the change-point detection method allows the coefficients to be updated promptly, even when the model found turns out to be unstable. As shown in the examples, whenever the decisive non-linearity is recovered, the method aligns itself properly with the true underlying dynamics in the long run, allowing the system to be completely described and its future behavior to be anticipated. In the two problems shown, changes in dynamics emerge over time in different ways. In the forced van der Pol oscillator, a term that was not initially estimated must be entirely recovered. In the non-autonomous Lotka-Volterra model, instead, the coefficients are changing the behavior differently over time. The success of these two experiments shows how Symbolic-SINDy can also be used effectively to deal with a wide class of online problems thanks to its flexibility. The costs associated with this method are relatively low, since the only additional operation consists of occasionally performing symbolic regression to save promising nonlinearities.

Despite the promising results, some critical aspects and inherent limitations of the proposed approach must be addressed to provide a comprehensive evaluation. In the problems considered, symbolic regression is decisive for the extraction of trigonometric functions with non-unit pulsation that are not included in principle within the SINDy function library. However, the pulsation estimation errors, although very low, cause the addition of compensatory terms in the identified dynamics. When these errors increase, the final model obtained, although largely correct, has a high error because of the presence of these compensatory terms. In fact, in different time instants than those considered, the missing building blocks are still estimated with good accuracy (e.g. $\cos(3.163\,t)$ or $\sin(5.028\,t)$), but they give rise to models with different dynamics. A critical issue with symbolic regression is precisely the slow updating of constants, an aspect that must therefore be taken into account, as it can potentially limit the applicability of the method. Moreover, both symbolic regression and SINDy are used to estimate the dynamics locally. For this reason, the dynamics has to be *clearly* observable inside the time window considered for the identification. For this reason, a periodic forcing term was employed in both examples to ensure that the system remains persistently excited, preventing the dynamics from

decaying toward a stable equilibrium point.

One final observation concerns the configuration of symbolic regression used: in both problems considered, it was necessary to limit the functions set to only the necessary functions. Although the idea behind Symbolic-SINDy is to automatically find promising nonlinearities from a large set of allowed operations, for complex problems a concise definition of the function set is essential. Note that in both problems, the nonlinearity to be recovered is partially masked by an interaction with the state or with time, thus simulating a realistic situation. Therefore, for more complicated problems, the applicability of the method is limited for its purpose, since the functions' set must be defined effectively for proper building block recovery.

5 | Symbolic-SINDy for high-dimensional problems

In many real-world applications, we are faced with physical phenomena that are distributed in space and evolve over time. For this type of problems, the direct application of traditional model discovery techniques might be prohibitive, since their inability to scale efficiently as dimensionality increases makes them computationally unfeasible. However, the dynamics of these high-dimensional systems is frequently governed by the evolution of a small number of latent variables. In this context, combining SINDy with an Autoencoder for the sake of dimensionality reduction has proven to be an extremely promising strategy, capable of identifying the governing equations in the latent domain in closed-form [6]. Since these variables are not directly observable, building a suitable library for the problem *a priori* is a real challenge, overall limiting the applicability of the method. The introduction of an intermediate step based on Symbolic-SINDy appears to be an appealing choice to overcome this limitation, and thus enhance the automatic discovery of the most suitable coordinate system.

5.1. Latent dynamics identification

Autoencoders (AEs) are neural networks that are often used to extrapolate low-dimensional features in high-dimensional data. Indeed, they allow a generally high-dimensional space to be mapped into a nonlinear manifold embedding, typically of lower dimension, called the *latent space*. Mathematically speaking, an autoencoder maps a high-dimensional input vector $\mathbf{x} \in \mathbb{R}^n$ ($n \gg 1$), to a low dimensional latent variable $\mathbf{z} \in \mathbb{R}^r$ ($n \gg r$), and then back to the high-dimensional space $\tilde{\mathbf{x}} \in \mathbb{R}^n$. The goal of the autoencoder is to reconstruct the input vector, i.e. $\|\tilde{\mathbf{x}} - \mathbf{x}\|_2 \approx 0$. The autoencoder is composed by two nonlinear maps, the *encoder* $\phi(\cdot\,; \mathbf{W}_\phi) : \mathbb{R}^n \to \mathbb{R}^r$ and the *decoder* $\psi(\cdot\,; \mathbf{W}_\psi) : \mathbb{R}^r \to \mathbb{R}^n$, respectively, where $\mathbf{W} = [\mathbf{W}_\phi, \mathbf{W}_\psi]$ are the weights of the associated network. Since

$$\mathbf{z} = \phi(\mathbf{x}\,; \mathbf{W}_\phi)$$
$$\tilde{\mathbf{x}} = \psi(\mathbf{z}\,; \mathbf{W}_\psi)$$

the neural network weights are optimized so that:

$$\arg\min_{\mathbf{W}} \|\mathbf{x}-\tilde{\mathbf{x}}\|_2^2 = \arg\min_{[\mathbf{W}_\phi,\mathbf{W}_\psi]} \|\mathbf{x}-\psi(\phi(\mathbf{x};\mathbf{W}_\phi);\mathbf{W}_\psi)\|_2^2.$$

Generally, r is made as small as possible until the autoencoder performance starts to fail. This choice allows the autoencoder to discover the intrinsic dimensionality of the data. Many scientific and engineering applications leverage low-dimensional coordinate systems to build parsimonious models that characterize a physical process [3].

5.1.1. SINDy Autoencoder

Although the coordinate system defined by an autoencoder is optimal for reconstructing the system under consideration, it does not guarantee the preservation of the structure of the dynamics of the underlying phenomenon. In order for the latent space to be consistent with the associated high-dimensional dynamics, it is necessary to inform the model, during training, of the temporal evolution of the system. To achieve this consistency, the loss function of the neural network is integrated with additional terms. In particular, assuming that the latent dynamics can be described by a sparse combination of terms within a library of nonlinear functions, dynamic consistency is imposed via SINDy. The total loss function is therefore defined as:

$$\begin{aligned}\mathcal{L}(\mathbf{x},\dot{\mathbf{x}};\mathbf{W},\boldsymbol{\Xi}) = &\underbrace{\|\mathbf{x}-\psi(\mathbf{z})\|_2^2}_{\text{reconstruction}} + \lambda_1 \underbrace{\|\dot{\mathbf{x}}-(\nabla_{\mathbf{z}}\psi(\mathbf{z}))(\boldsymbol{\Theta}(\mathbf{z}^T)\boldsymbol{\Xi})\|_2^2}_{\text{SINDy in }\dot{\mathbf{x}}} \\ &+ \lambda_2 \underbrace{\|(\nabla_{\mathbf{x}}\mathbf{z})\dot{\mathbf{x}}-\boldsymbol{\Theta}(\mathbf{z}^T)\boldsymbol{\Xi}\|_2^2}_{\text{SINDy in }\dot{\mathbf{z}}} + \lambda_3 \underbrace{\|\boldsymbol{\Xi}\|_1}_{L_1\text{ reg}}.\end{aligned} \tag{5.1}$$

The optimal parameters are then found by solving the following problem:

$$\arg\min_{\mathbf{W},\boldsymbol{\Xi}} \mathcal{L}(\mathbf{x},\dot{\mathbf{x}};\mathbf{W},\boldsymbol{\Xi}).$$

Thanks to this formulation, the coordinate system discovered by the autoencoder allows an accurate reconstruction of the state and also a closed-form description of its evolution. It should be noted that the coefficients of the library $\boldsymbol{\Xi}$ become trainable parameters, which are sparsified in presence of L_1 regularization. This model is commonly known as **SINDy Autoencoder** (AE+SINDy) [6]. A schematic visualization of the network is provided in Figure 5.1.

SINDy has a significant impact on the coordinate system. On the one hand, latent coordinates and high-dimensional dynamics contribute to the selection of active library

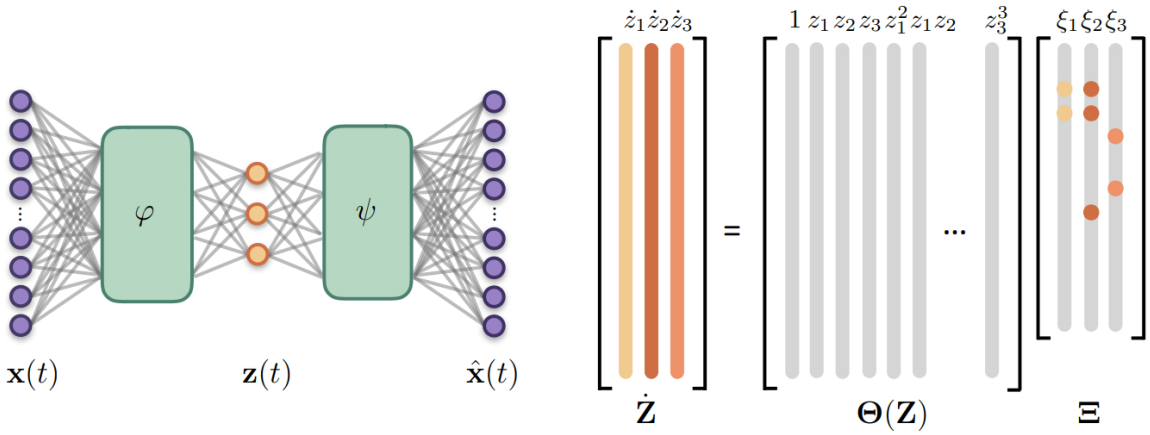


Figure 5.1: Schematic AE+SINDy architecture. The network is trained with (5.1) as objective function. Image sourced from [6].

terms; on the other hand, SINDy's loss influences the latent trajectory, forcing it to adapt to the imposed functional structure. During training, an increasing number of terms are discarded, making the space more regular.

Finally, AE+SINDy's ability to identify the latent dynamics of high-dimensional systems can also be exploited in the context of Reduced Order Modeling (ROM). Once trained, the model can evolve the system at the latent level, while the decoder is used to reconstruct the corresponding high-dimensional behavior. In this sense, AE+SINDy provides an effective surrogate model for complex dynamics that would otherwise require the use of computationally expensive solvers [8].

5.1.2. Limitations and Symbolic-SINDy integration

While the AE+SINDy framework has demonstrated remarkable results and significant flexibility across various scenarios, it inherits a fundamental limitation from the standard SINDy approach: the requirement to define a suitable library *a priori*. By construction, the loss constraints enforce the alignment between the latent trajectories and the library functions. However, relevant latent inconsistencies can make this matching process impossible. Typically, the library is constructed from polynomial and elementary trigonometric functions of the latent variables, with the possible inclusion of explicit time dependence. Nevertheless, such a library may prove insufficient when dealing with high-frequency oscillations or highly irregular dynamics.

Since latent dynamics cannot be directly observed, building a library of functions suited to the problem is a very complex task. As shown in previous chapters, the automatic extrac-

tion of building blocks through symbolic regression appears to be an appealing strategy to overcome this limitation, and automatically build a CFL specific to the phenomenon, enhancing the dynamics identification.

In cases where the standard AE+SINDy framework fails in its scope, the proposed approach involves partial execution of the architecture to generate an intermediate latent representation on which to apply Symbolic-SINDy. At this stage, the coordinate system does not need to fit the library, but rather to accurately reconstruct the state with a minimum degree of regularization imposed by the dynamics. The training set is then projected into this intermediate space, where Symbolic-SINDy extracts the best block that describes the evolution of the data. This expression is then integrated into the SINDy library of the AE+SINDy architecture, with the aim of identifying a more appropriate coordinate system and facilitating the reconstruction of the latent dynamics. In the following section we demonstrate the feasiblity of this idea on a large-scale dynamical system representing an idealized model of high-vorticity fluid flow.

5.2. Double gyre flow

The double gyre flow is a time-dependent model for two counter-rotating vortices in a rectangular domain. When time is introduced via a periodic perturbation, the central dividing line between the two gyres oscillates left and right, creating a time-varying velocity field that can lead to chaotic particle trajectories [29]. The velocity field $\mathbf{v} = [u, v]^T$ in the domain $[0, L_x] \times [0, L_y]$ and in the time interval $[0, T]$ is governed by:

$$u(x, y, t) = \frac{\partial x}{\partial t} = -\pi A \sin\left(\pi f(x, t)\right) \cos\left(\pi y\right) \tag{5.2}$$

$$v(x, y, t) = \frac{\partial y}{\partial t} = \pi A \cos\left(\pi f(x, t)\right) \sin\left(\pi y\right) \frac{\partial f(x, t)}{\partial x} \tag{5.3}$$

and the time dependency is introduced by

$$f(x, t) = a(t)x^2 + b(t)x,$$

with time dependent coefficients

$$a(t) = \epsilon \sin(\omega t), \qquad b(t) = 1 - 2\epsilon \sin(\omega t).$$

Here, $A \in \mathbb{R}$, $\epsilon \in \mathbb{R}$, $\omega \in \mathbb{R}$ are the model parameters in the flow pattern. In particular, A controls the velocity magnitude, while ϵ and ω respectively determine the amplitude and frequency of the oscillation in the x-direction.

Note that this system is inherently high-dimensional, as it derives from a simplified analytical description of the double-gyre pattern, a recurring phenomenon in geophysical flows [29]. The system evolves spatially under the influence of a periodic external forcing that influences its behavior over time. The resulting dynamics are purely oscillatory in nature, resulting from the multiplication of spatial and temporal trigonometric functions. In this setting, the standard AE+SINDy framework may encounter significant limitations in describing the latent dynamics when using a purely polynomial library. In contrast, a problem-informed library can lead to a more suitable coordinate representation.

5.2.1. Problem setup

The numerical simulation of the double gyre flow is performed using the implementation provided in [32]. The simulation is carried out within a rectangular domain of size $[0, L_x] \times [0, L_y] = [0, 2] \times [0, 1]$. The spatial structures of the flow is discretized using a regular grid with $n_x = 64$ and $n_y = 32$ points along the respective axes. The physical parameters of the system are set to $A = 0.1$, $\epsilon = 0.25$, and $\omega = 5.0$. The dynamics of the system are simulated over a time horizon $T = 100$ with a constant temporal resolution $dt = 0.05$. This time discretization ensures that the periodic oscillations of the gyres are captured accurately throughout the simulation interval $[0, T]$. In this experiment, the simulated system is not affected by noise contamination. An example of the simulated dynamics is shown in Figure 5.2.

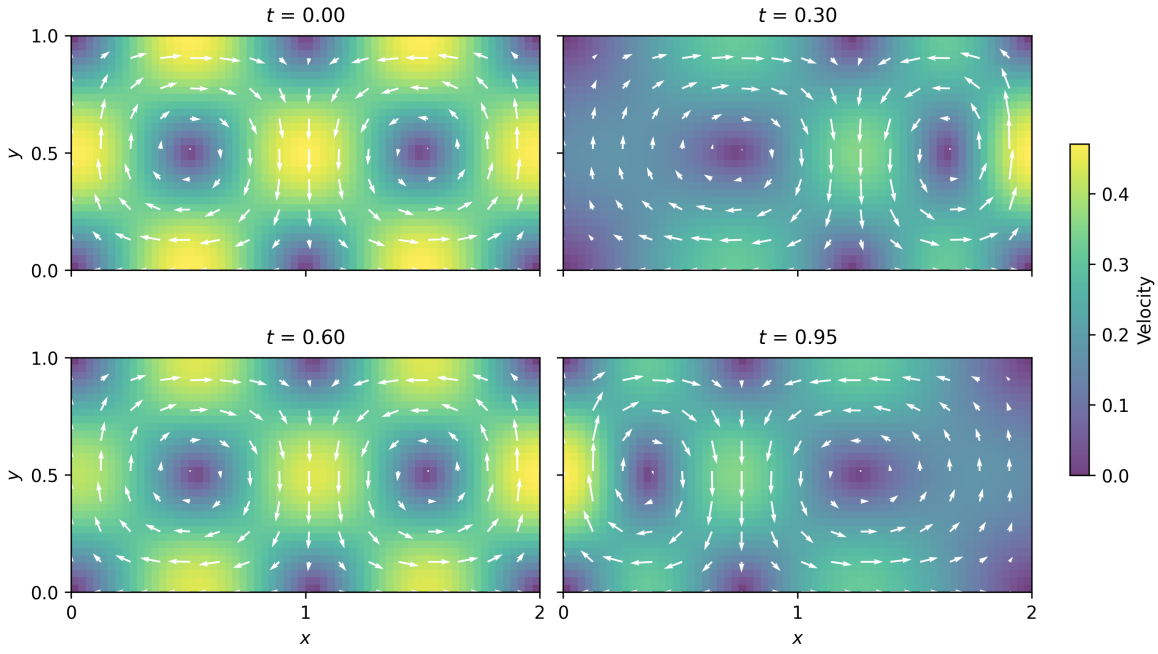


Figure 5.2: Snapshots of the simulated double gyre at times $t = \{0, 0.30, 0.60, 0.95\}$.

5.2.2. Methodology

To identify the latent dynamics of the double gyre flow, two autoencoder models are employed: one equipped with a standard SINDy library, and one using a library constructed according to the intermediate Symbolic-SINDy execution. The performance of the two methods is then compared in terms of the quality of the identified latent space and their ability to reproduce the high-dimensional dynamics. In particular, in the experiments, the following framework is employed.

Neural network structure. Given the time-dependent two-dimensional vector field nature of double gyre, the optimal strategy for data compression and latent feature extraction lies in the use of a **Convolutional Autoencoder** (CAE) [22]. In contrast with a traditional dense autoencoder, in this architecture convolutional layers are used because they allow for more effective preservation and characterization of the spatial correlations of the field. Moreover, the multi-channel structure of convolutional networks enables a better handling of the two vector components u and v, allowing the model to learn the inter-dependencies between the two scalar fields in a coherent manner [3]. The latent space is chosen to be one-dimensional ($r = 1$). This choice is motivated by the fact that, when higher-dimensional latent spaces are considered, the learned dynamics rapidly collapse onto a one-dimensional manifold, indicating that the system exhibits intrinsically a lower-dimensional behavior. The convolutional encoder consists of two layers with 16 and 32 filters respectively. The decoder is constructed as the symmetric counterpart of the encoder.

Training procedure. The dataset consists of 2001 snapshots, generated with a constant temporal and spatial discretization of $dt = 0.05$, $n_x = 64$, $n_y = 32$ respectively, for both scalar velocity components u and v. Accordingly, data are arranged in tensors of size $[N, C, H, W] = [2001, 2, 64, 32]$. The first 80% of the dataset is used for training the network, the subsequent 10% is used for validation, and the remaining 10% is used for testing the learned dynamics. To preserve the dynamics at a latent level, additional penalties are included in the loss function of the convolutional autoencoder, following the AE+SINDy methodological approach. The overall loss function can be expressed as:

$$\mathcal{L} = \underbrace{\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2}_{\text{reconstruction}} + \lambda_1 \underbrace{\|\dot{\mathbf{v}} - \hat{\dot{\mathbf{v}}}\|_2^2}_{\text{SINDy (decoded) in } \dot{\mathbf{v}}} + \lambda_2 \underbrace{\|\dot{z} - \boldsymbol{\Theta}(z)\boldsymbol{\Xi}\|_2^2}_{\text{SINDy in } \dot{z}} + \lambda_3 \underbrace{\|\boldsymbol{\Xi}\|_1}_{L_1 \text{ reg}}, \tag{5.4}$$

where $\mathbf{v}$ is the high-dimensional velocity field and z its one-dimensional latent represen-

tation. Note that the strategy of introducing the three penalty components to preserve the dynamics in the latent representation is completely general, and is independent of the choice of network, and therefore perfectly adaptable to a convolutional autoencoder.

CAE+SINDy configuration. In the standard configuration, a polynomial library up to the third degree is employed. Moreover, a sinusoidal function of the state is also included, motivated by the presence of periodicity in the high-dimensional system. Since the latent dimension is one, the resulting library is given by

$$\boldsymbol{\Theta}(z) = [1,\ z,\ z^2,\ z^3,\ \sin(z)]. \tag{5.5}$$

The network is trained for 1001 epochs, during which coefficient thresholding is active. This allows for a more accurate selection of the active terms in the library. Subsequently, an additional 201 epochs are used to refine the remaining coefficients. A detailed description of the neural network architecture and its hyperparameters is provided in the Appendix C.

CAE+Symbolic-SINDy configuration. In order to apply Symbolic-SINDy, a preliminary dimensionality reduction that takes into account the dynamics is necessary. For this purpose, a convolutional autoencoder is used, employing the standard library configuration defined in (5.5). In order to avoid identifying an intermediate latent space that is overly influenced by the presence of the initial library, the autoencoder undergoes partial training along with appropriate tuning of the penalty terms. The training data are then projected into the identified intermediate latent space. Symbolic-SINDy is performed on this trajectory, extracting a block $b(z,t)$ function of time and space. In particular, Symbolic-SINDy is executed based on D-CODE, with a functions set $\mathcal{F} = \{+, -, \times, -1, \sin, \cos\}$, using a maximum number of 11 generations. The found block is then integrated into a third-order polynomial library which will be used within a new complete training of the convolutional autoencoder. The resulting library is given by

$$\boldsymbol{\Theta}(z,t) = [1,\ z,\ z^2,\ z^3,\ b(z,t)]. \tag{5.6}$$

This network is trained for 1001 epochs, during which coefficient thresholding is active. Subsequently, an additional 201 epochs are used to refine the remaining coefficients. A detailed description of the neural network architecture and its hyperparameters is provided in the Appendix C.

5.2.3. Numerical results

As explained in the previous section, the latent space is chosen to be one-dimensional, since when considering a larger bottleneck, the high-dimensional projected observations systematically lie on a one-dimensional manifold.

CAE+SINDy. Applying the double gyre flow training data to the convolutional autoencoder augmented with the SINDy library in (5.5), the identified latent dynamics is:

$$\dot{z} = 0.422z \; - \; 0.425\sin(z). \tag{5.7}$$

Although the metrics obtained through training appear satisfactory and the discovered model is parsimonious, a more in depth analysis reveals some critical issues. Specifically, the results show an extremely low percentage error in the reconstruction of the velocity field on the validation set (0.4%) and a reasonable error on the derivative (9%). The main limitation emerges when comparing the SINDy loss value in the latent space with the order of magnitude of the latent variables themselves. Indeed, the observations projected in the latent space have an order of magnitude of approximately 2.6×10^{-2}, while the error associated with the dynamics $\dot{z}$ is 1.7×10^{-2}. This implies a relative error of more than 60%, indicating a profound inconsistency in the identified dynamics, despite the small absolute value of the relative loss. It is possible to leverage that term by increasing the associated penalty (λ_2) in the overall loss function. With this choice, the latent SINDy loss decreases in absolute value, but the gain is only apparent. Indeed, this choice does not entail substantial changes in the identified dynamics, but affects the order of magnitude of the latent space, contracting it further. In this way, the relative error remains high and the inconsistency persists. The inability to learn the dynamics of the problem can be better appreciated by encoding the test set into latent level and evolving its first snapshot according to the identified dynamics in (5.7), to compare the obtained trajectory with the ground-truth behavior. In Figure 5.3, it can be seen that the latent governing equation discovered is ineffective. Indeed, the predicted curve is essentially constant over time. The estimated latent dynamics does not evolve and is totally different from the real trajectory. This explains why an increase in λ_2, which reduces the latent SINDy loss, results in a latent space that is significantly smaller in order of magnitude. In fact, by forcing the latent space to follow the identified dynamics, it is necessary to keep the oscillations of the real trajectory less pronounced and attenuated. Obviously, the latent mismatch consequently impacts the quality of the reconstruction of the high-dimensional dynamics. By employing the decoder to project the latent trajectory from Figure 5.5 back into the

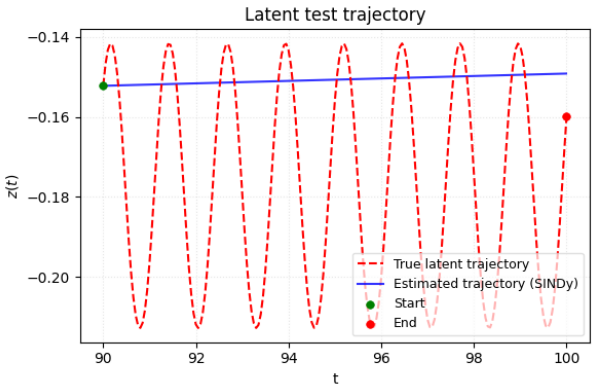


Figure 5.3: Comparison between the true latent trajectory and the estimated trajectory with (5.7) starting from the first test snapshot.

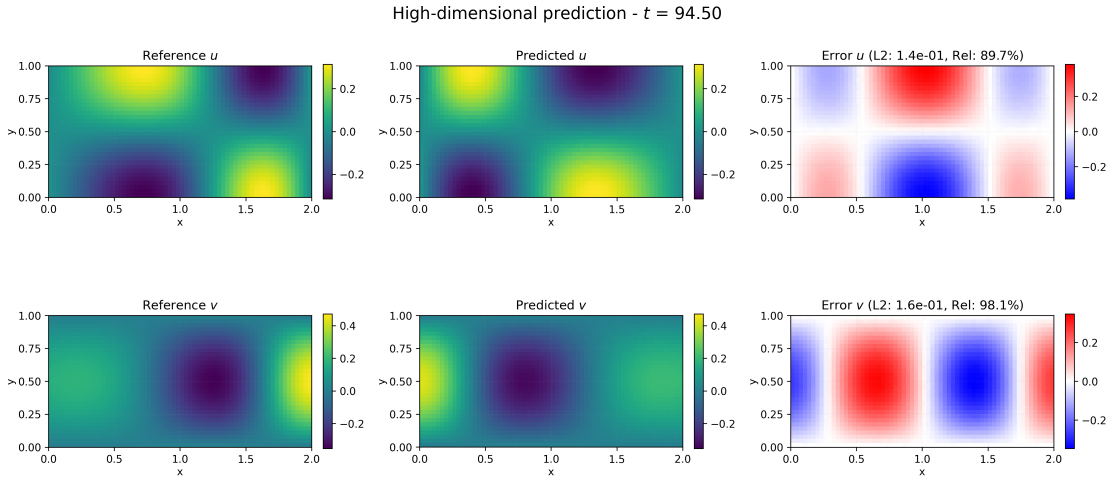
high-dimensional space, a substantial inconsistency can be observed, as shown in Figure 5.4. It can be seen that the predicted reconstructions in Figures 5.4a, 5.4b, and 5.4c remain the same and do not evolve. Given the oscillatory nature of the problem, there are moments when the reconstruction exhibits a smaller relative error in the L_2 norm with respect to the true high-dimensional field, of around 10%, and moments when the error increases, reaching peaks of over 100%. In this sense, it is clear that the library used is not representative of the system under observation, requiring the formulation of a new set of nonlinear latent basis functions, which are consistent with the specific problem.

CAE+Symbolic-SINDy. Before Symbolic-SINDy can be executed, a suitable latent space must be identified. For this purpose, a partial training of a standard CAE+SINDy is used. To reduce the bias induced by the preliminary SINDy library, the term λ_2 in the total loss function is relaxed. As previously discussed, the component associated with that penalty, i.e., the SINDy loss in $\dot{z}$, affects the order of magnitude of the latent space, restricting it. Once the training data are encoded into these intermediate latent coordinates, Symbolic-SINDy is performed. With this run $b(z,t) = \cos(4.998\,t)$ is extracted and the new nonlinear library to feed the convolutional autoencoder is defined as follows:

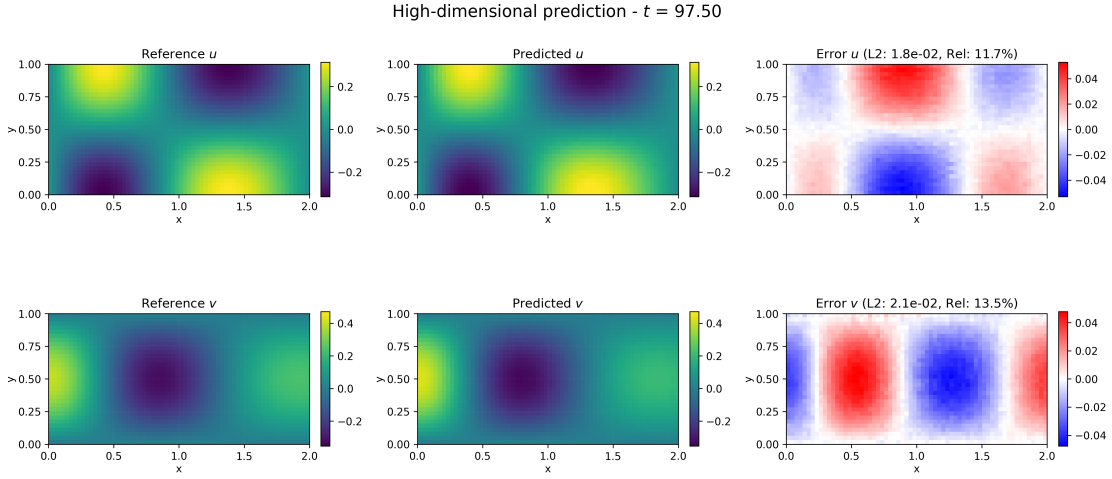
$$\boldsymbol{\Theta}(z,t) = [1,\ z,\ z^2,\ z^3,\ \cos(4.998\,t)]. \tag{5.8}$$

Applying the double gyre flow training data to the convolutional autoencoder augmented with the SINDy library in (5.8), the identified latent dynamics becomes:

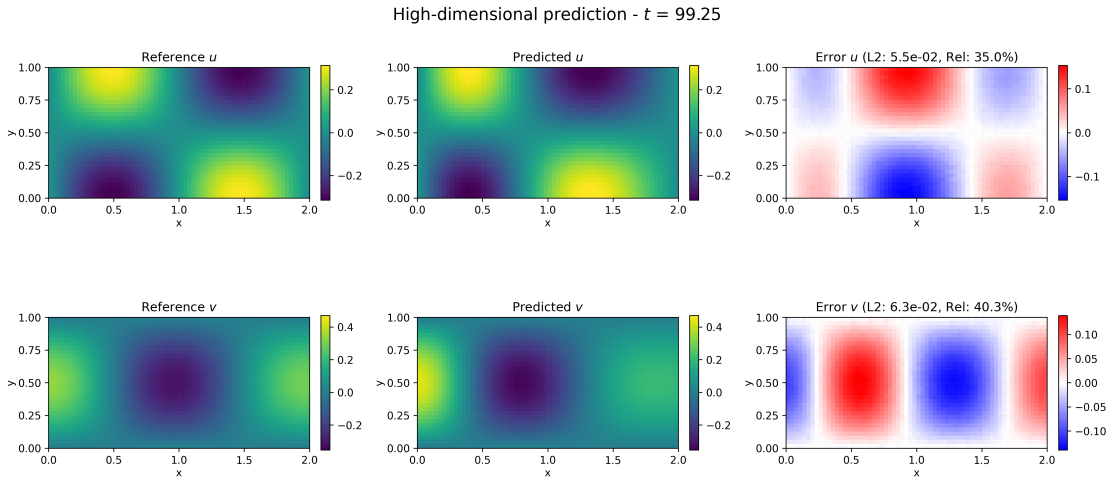
$$\dot{z} = -0.052 - 0.311z - 0.639\cos(4.998\,t). \tag{5.9}$$



(a) Estimated high-dimensional reconstruction and absolute error at $t = 94.50$.



(b) Estimated high-dimensional reconstruction and absolute error at $t = 97.50$.



(c) Estimated high-dimensional reconstruction and absolute error at $t = 99.25$.

Figure 5.4: Comparison between the true high-dimensional observation and the predicted reconstruction with (5.7) starting from the first test snapshot, at $t = \{94.50, 97.50, 99.25\}$.

Unlike in the previous case, the training is considerably more robust. The results show an extremely low percentage error on the validation set both in the reconstruction of the velocity field (0.01%) and in its derivative (0.3%). The SINDy loss value in the latent space is also smaller, with an absolute value of 1.0×10^{-3}. The order of magnitude of the constructed low-dimensional space is 1.0×10^{-1}, therefore getting a relative error of about 1%. Not only is this value significantly lower than in the standard case, but it also demonstrates the consistency of the identified latent model with respect to the adopted library. Indeed, by encoding the test data into the latent space and comparing the predicted trajectory evolved from the first snapshot according to (5.9) with the ground-truth latent curve, a significant improvement emerges. As shown in Figure 5.5, the discovered governing equation is closely aligned with the actual physical behavior of the system.

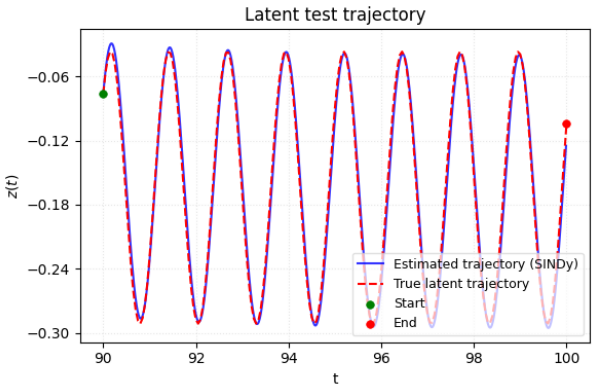
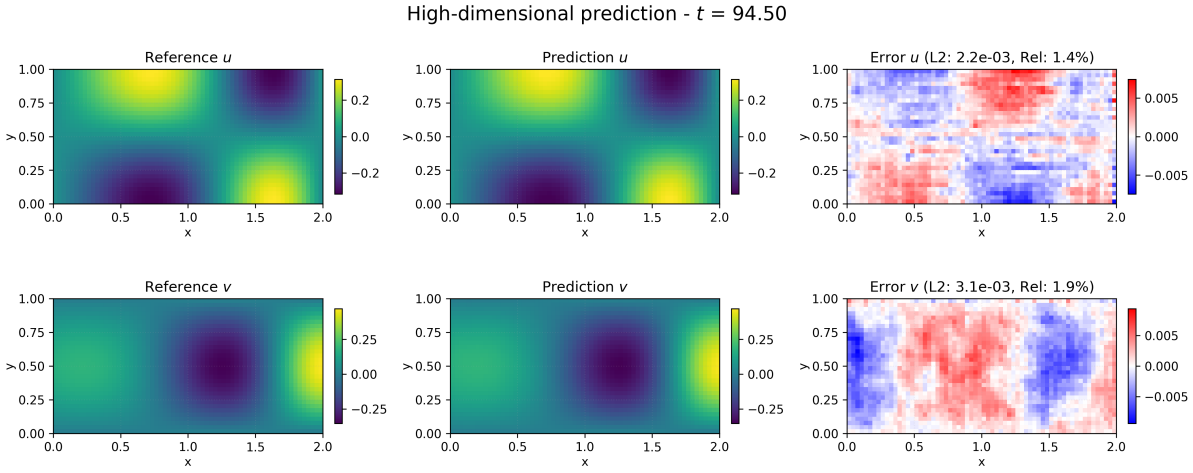


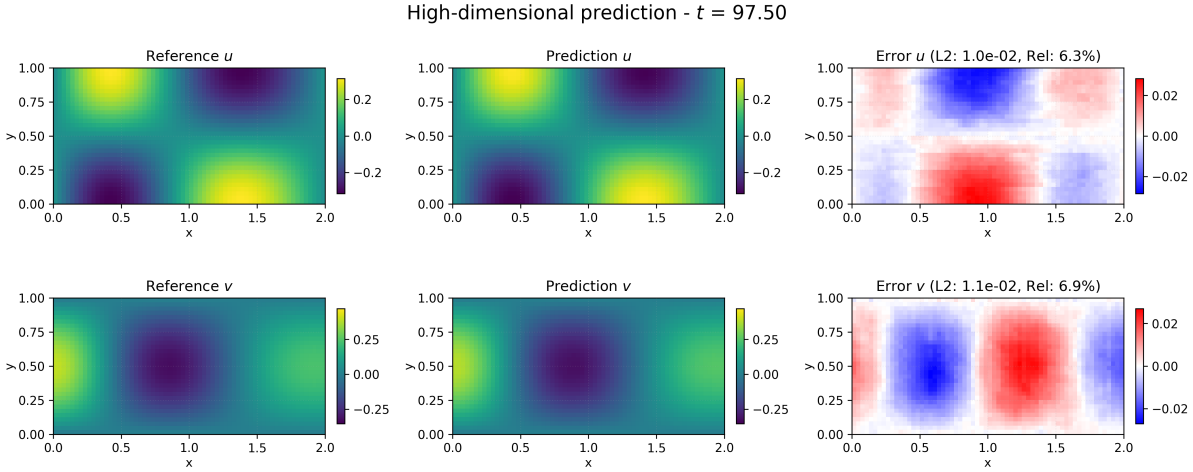
Figure 5.5: Comparison between the true latent trajectory and the estimated trajectory with (5.9) starting from the first test snapshot.

By employing the decoder to project the latent trajectory from Figure 5.5 back into the high-dimensional space, the preservation of the original dynamics is clearly observed, as illustrated in Figure 5.6. Throughout the simulation, the relative reconstruction error in L_2 for the velocity field components remains low, never exceeding 8%, and demonstrating a robust capacity to reproduce the underlying physical phenomenon.

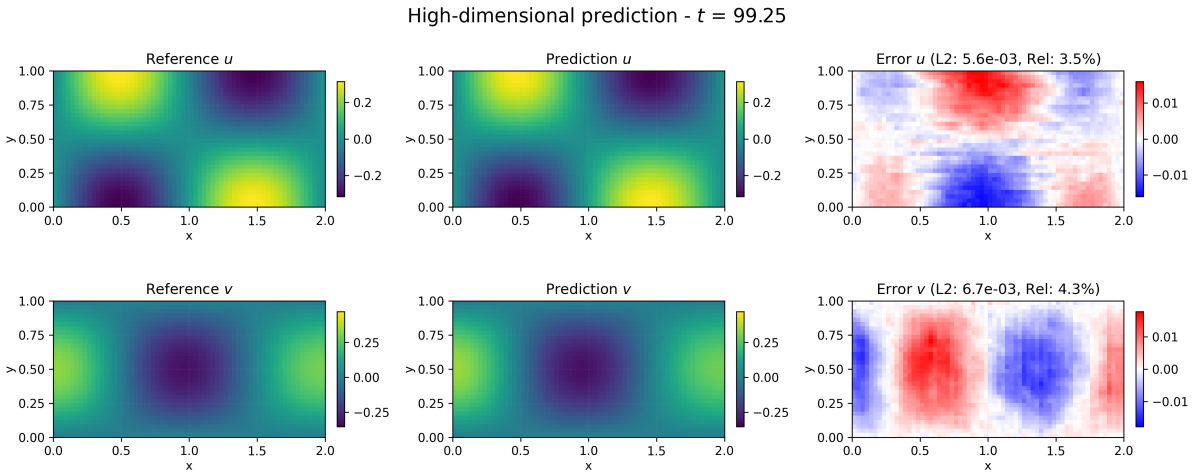
This example clearly shows how the use of a standard library is insufficient to adequately capture the complexity of the dynamics of a double gyre flow. In contrast, the addition of an intermediate step, in which Symbolic-SINDy extracts a group of promising building blocks at a latent level and chooses the best one, is essential for the construction of an effective library for the problem. Note that, in general, it is always possible to include time-dependency inside the starting library; however, it would be difficult to consider a trigonometric function with non-unit pulsation, especially in this example, where the dynamics exhibits evident oscillations both in time and space with different angular veloc-



(a) Estimated high-dimensional reconstruction and absolute error at $t = 94.50$.



(b) Estimated high-dimensional reconstruction and absolute error at $t = 97.50$.



(c) Estimated high-dimensional reconstruction and absolute error at $t = 99.25$.

Figure 5.6: Comparison between the true high-dimensional observation and the predicted reconstruction with (5.9) starting from the first test snapshot, at $t = \{94.50, 97.50, 99.25\}$.

ity. The combination of the new informed library and the autoencoder thus manages to identify a better coordinate system capable of adequately describing the latent dynamics consistently with the high-dimensional behavior.

5.3. Results discussion and considerations

This experiment shows encouraging results regarding the use of Symbolic-SINDy within the latent coordinates of an autoencoder. Its role proves to be decisive in identifying fundamental nonlinearities to enhance the discovery of the governing equations of a high-dimensional system. For this problem, the building block extracted with Symbolic-SINDy is $\cos(4.998\,t)$. Note that the standard library (5.5) used initially cannot capture this type of oscillation over time, even by increasing the degree of the polynomial considered. Moreover, given the oscillatory behavior both in state and time of the double gyre, it would be difficult to consider without prior knowledge a trigonometric function with non-unit pulsation inside such a library. In contrast, once Symbolic-SINDy extracts this block and integrates it into the library, the model discovery task is considerably simplified, converging rapidly towards a model consistent with the dynamics. It should also be noted that the structure of the identified block is coherent with the pulsation of the sinusoidal forcing from which the system is perturbed. Although there is a very small estimation error in the angular velocity, the identified model describes the dynamics of the system with high efficiency and accuracy. This is due to the presence of the two additional terms in (5.9), a small intercept, and a linear contribution of the state, which compensate for the error in the estimated frequency, mitigating the small time shift of the oscillation period. In this sense, both the contribution of the integrated nonlinearity found by Symbolic-SINDy and the terms already present inside the standard SINDy library are essential to enhance model discovery. The results obtained in this experiment are further confirmed by additional tests, carried out by considering different angular velocities for the forcing term, demonstrating the robustness of the method when tackling the task of identifying the dynamics of a large-scale, distributed system.

The results obtained in this experiment are extremely encouraging. The integration of Symbolic-SINDy applied to an inherently high-dimensional system allows the dynamics to be effectively identified, solving a problem that would otherwise be very complex – or even impossible – to tackle for standard approaches.

6 | Conclusions and future developments

This thesis consolidates and extends the Symbolic-SINDy framework, originally introduced by [24], in the context of data-driven model discovery for dynamical systems. The flexibility of symbolic regression is exploited to search within a large functional space for promising building blocks. These problem-specific nonlinear mathematical expressions automatically expand a polynomial SINDy library. Through sparse regression within this augmented library of candidate functions, it is finally possible to obtain a parsimonious estimate of the dynamics. This approach exploits the complementarity of symbolic regression and SINDy, overcoming their respective practical limitations. Indeed, through their synergy, it is possible to increase user independence while maintaining computational tractability and time efficiency.

First, Symbolic-SINDy is tested on more realistic and complicated problems, where both symbolic regression and SINDy struggle with the model discovery scope. Specifically, in the two proposed cases, the Hill equation and a revised version of the van der Pol oscillator, the current implementation outperformed existing methodologies. Moreover, the integration of an uncertainty quantification framework based on ensemble methods proves useful in the post-processing phase for validating the identified models. Together with the examples proposed in [24], the method has been tested against several examples, of different structure and complexity, systematically demostrating its effectiveness. Subsequently, Symbolic-SINDy is combined with the CUSUM change-point algorithm to address the online learning of dynamical systems. By leveraging the flexibility of symbolic regression, it is possible to extract local nonlinearities in both state and temporal domains, thus adapting the model discovered to abrupt changes in the underlying dynamics over time. In particular, the developed method proves capable of recovering nonlinear terms that were not initially estimated, as demonstrated in the forced van der Pol oscillator, and robustly adapting to sudden variations in the coefficients' behaviors, as shown in the time-varying Lotka-Volterra model. Finally, Symbolic-SINDy is integrated within the latent space of a convolutional autoencoder to enhance the discovery of the govern-

ing equations for high-dimensional systems. In the double gyre flow, this framework is fundamental in defining a new library of nonlinear functions, specific for the problem, and therefore identifying a coordinate system capable of describing the latent dynamics consistently with the high-dimensional behavior. In this sense, Symbolic-SINDy allows for the resolution of problems that would otherwise be complex or even impossible for standard approaches.

Despite the promising results, certain limitations of the proposed method need to be addressed. In complex scenarios, in order for Symbolic-SINDy to be effective, the genetic programming step has often been configured with a constrained functions set, limited to the operations that are truly necessary for the problem. In this sense, the potential of symbolic regression as a flexible tool for discovering arbitrary nonlinearities is partially compromised by the need for such prior knowledge. Furthermore, as the complexity of the problem increases, the number of generations required to extract accurate and promising building blocks grows significantly. These two factors impact the real applicability of the method in complicated model discovery settings. To overcome these limitations, future developments could involve replacing genetic programming-based approaches with emerging deep-symbolic models, such as Deep Generative Symbolic Regression [15]. These architectures could preserve the logic of the current implementation while enhancing its applicability.

Finally, extremely promising results emerge from the application of Symbolic-SINDy within the context of Reduced Order Modeling (ROM). In this framework, the method proves to be effective in the identification of a latent dynamics that remains consistent with the corresponding high-dimensional behavior. These preliminary and encouraging results justify further investigation of the real potential of the method with respect to more complex high-dimensional dynamical systems, starting with, for instance, a parametrized version of the double gyre problem.

Bibliography

[1] A. A. Berryman. The orgins and evolution of predator-prey theory. *Ecology*, 73(5): 1530–1535, 1992.

[2] L. Biggio, T. Bendinelli, A. Neitz, A. Lucchi, and G. Parascandolo. Neural symbolic regression that scales. In *International Conference on Machine Learning*, pages 936–945. Pmlr, 2021.

[3] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2 edition, 2022.

[4] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[5] G. Camps-Valls, A. Gerhardus, U. Ninad, G. Varando, G. Martius, E. Balaguer-Ballester, R. Vinuesa, E. Diaz, L. Zanna, and J. Runge. Discovering causal relations and equations from data. *Physics Reports*, 1044:1–68, 2023.

[6] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[7] R. Chartrand. Numerical differentiation of noisy, nonsmooth data. *International Scholarly Research Notices*, 2011(1):164564, 2011.

[8] P. Conti, G. Gobat, S. Fresca, A. Manzoni, and A. Frangi. Reduced order modeling of parametrized systems through autoencoders and SINDy approach: continuation of periodic solutions. *Computer Methods in Applied Mechanics and Engineering*, 411: 116072, 2023.

[9] P. Conti, J. Kneifl, A. Manzoni, A. Frangi, J. Fehr, S. L. Brunton, and J. N. Kutz. VENI, VINDy, VICI: a generative reduced-order modeling framework with uncertainty quantification. *Neural Networks*, page 108543, 2026.

[10] S. d'Ascoli, S. Becker, A. Mathis, P. Schwaller, and N. Kilbertus. ODEFormer:

Symbolic regression of dynamical systems with transformers. *arXiv preprint arXiv:2310.05573*, 2023.

[11] U. Fasel, J. N. Kutz, B. W. Brunton, and S. L. Brunton. Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A*, 478(2260):20210904, 2022.

[12] S. Goutelle, M. Maurin, F. Rougier, X. Barbaut, L. Bourguignon, M. Ducher, and P. Maire. The Hill equation: a review of its capabilities in pharmacological modelling. *Fundamental & clinical pharmacology*, 22(6):633–648, 2008.

[13] P. Granjon. The CUSUM Algorithm: A Small Review. `https://inria.hal.science/hal-01442693/file/RR-9010.pdf`, 2013. Available online; accessed: 2026-01-21.

[14] S. M. Hirsh, D. A. Barajas-Solano, and J. N. Kutz. Sparsifying priors for Bayesian uncertainty quantification in model discovery. *Royal Society open science*, 9(2):211823, 2022.

[15] S. Holt, Z. Qian, and M. van der Schaar. Deep generative symbolic regression. *arXiv preprint arXiv:2401.00282*, 2023.

[16] S. Holt, Z. Qian, T. Liu, J. Weatherall, and M. van der Schaar. Data-driven discovery of dynamical systems in pharmacology using large language models. *Advances in Neural Information Processing Systems*, 37:96325–96366, 2024.

[17] Y. Jin, W. Fu, J. Kang, J. Guo, and J. Guo. Bayesian symbolic regression. *arXiv preprint arXiv:1910.08892*, 2019.

[18] K. Kacprzyk, Z. Qian, and M. van der Schaar. D-CIPHER: discovery of closed-form partial differential equations. *Advances in Neural Information Processing Systems*, 36:27609–27644, 2023.

[19] J. R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, 1994.

[20] Y. A. Kuznetsov, S. Muratori, and S. Rinaldi. Bifurcations and chaos in a periodic predator-prey model. *International Journal of Bifurcation and Chaos*, 2(01):117–128, 1992.

[21] A. J. Lotka. *Elements of physical biology*. Williams & Wilkins, 1925.

[22] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-

encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pages 52–59. Springer, 2011.

[23] Z. Qian, K. Kacprzyk, and M. van der Schaar. D-CODE: Discovering Closed-Form ODEs from Observed Trajectories. In *International Conference on Learning Representations (ICLR)*, 2022.

[24] G. Romano. Model Discovery integrating Symbolic Regression into Sparse Identification of Nonlinear Dynamics. Master's thesis, Politecnico di Milano, 2024.

[25] L. Rosafalco, P. Conti, A. Manzoni, S. Mariani, and A. Frangi. EKF–SINDy: Empowering the extended Kalman filter with sparse identification of nonlinear dynamics. *Computer Methods in Applied Mechanics and Engineering*, 431:117264, 2024.

[26] L. Rosafalco, P. Conti, A. Manzoni, S. Mariani, and A. Frangi. Online learning in bifurcating dynamic systems via SINDy and Kalman filtering. *Nonlinear Dynamics*, pages 1–21, 2025.

[27] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Data-driven discovery of partial differential equations. *Science advances*, 3(4):e1602614, 2017.

[28] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[29] S. C. Shadden, F. Lekien, and J. E. Marsden. Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena*, 212(3-4):271–304, 2005.

[30] S. H. Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. Chapman and Hall/CRC, 2024.

[31] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

[32] M. Tomasetto, J. P. Williams, F. Braghin, A. Manzoni, and J. N. Kutz. SHRED-ROM: Reduced order modeling with shallow recurrent decoder networks. `https://github.com/MatteoTomasetto/SHRED-ROM`, 2025. Accessed on: 2026-02-14.

[33] M. Tsatsos. The Van der Pol equation. *arXiv preprint arXiv:0803.1658*, 2008.

[34] S.-M. Udrescu and M. Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science advances*, 6(16):eaay2631, 2020.

[35] B. Van der Pol. On relaxation-oscillations. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):978–992, 1926.

[36] V. Volterra. Fluctuations in the abundance of a species considered mathematically. *Nature*, 118:558–560, 1926.

[37] K. Yu, E. Chatzi, and G. Kissas. Grammar-based ordinary differential equation discovery. *Mechanical Systems and Signal Processing*, 240:113395, 2025.

A | Appendix A

A.1. Hill equation: supplementary information

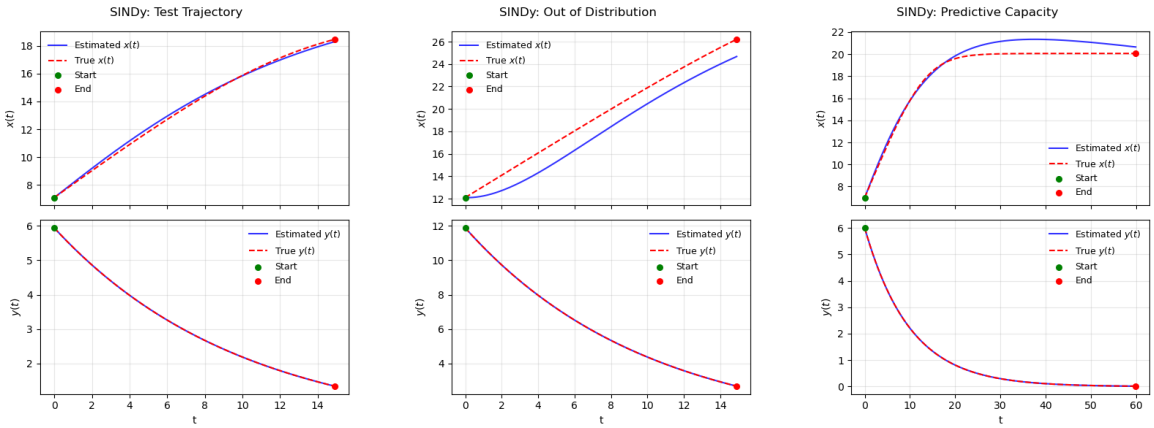
Hill – SINDy. Here are reported the hyperparameters used and the result of each experiment using SINDy for the discovery of the Hill equation.

Parameter	Description	Value
`degree`	Degree of the polynomial library	3
L	Sparsity threshold in STLSQ	0.03

Table A.1: Hill – SINDy hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
–	0	1	0.206	$\simeq 0$	$\simeq 0$

Table A.2: Hill – SINDy results for different seeds.



(a) $(x_0, y_0) = (7, 6)$ (b) $(x_0, y_0) = (12, 12)$ (c) $(x_0, y_0) = (7, 6)$

Figure A.1: SINDy for the Hill equation.

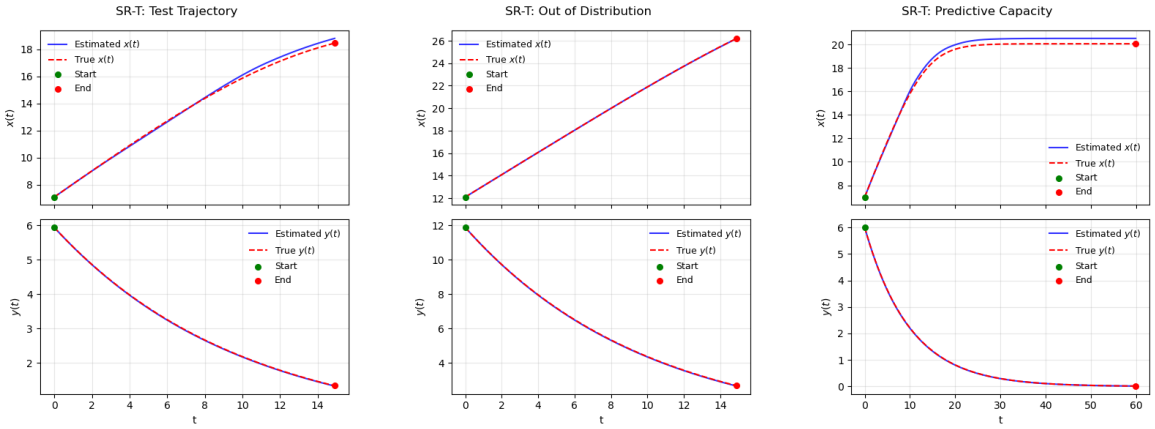
Hill – SR-T. Here are reported the hyperparameters used and the result of each experiment using SR-T for the discovery of the Hill equation.

Parameter	Description	Value
n_{gen}	Maximum number of generations	20
$\mathcal{F}$	Function set	$\{+, -1, \times, \div, \wedge\}$
$\mathcal{C}$	Constant range	$[1, 4]$
λ	Parsimony coefficient	0.01

Table A.3: Hill – SR-T hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
100	1	1	0.090	426.068	334.408
101	1	1	0.040	327.127	260.936
102	1	1	0.080	381.556	215.437
103	1	1	0.073	328.694	394.246
104	1	1	0.062	338.570	334.058
105	1	1	0.217	330.885	300.451
106	1	1	0.133	332.968	359.787
107	1	1	0.077	315.380	298.076
108	0	1	0.252	221.179	368.908
109	1	1	0.044	227.139	298.695
110	1	1	0.074	315.522	245.443
111	1	1	0.089	327.718	315.194
112	1	1	0.077	240.839	306.118
113	0	1	0.173	213.413	238.106
114	0	1	0.239	169.994	299.226
115	1	1	0.050	297.542	240.003
116	1	1	0.034	305.410	241.817
117	1	1	0.034	310.028	263.856
118	0	1	0.039	241.751	298.174
119	1	1	0.089	200.093	311.278

Table A.4: Hill – SR-T results for different seeds.



(a) $(x_0, y_0) = (7, 6)$ (b) $(x_0, y_0) = (12, 12)$ (c) $(x_0, y_0) = (7, 6)$

Figure A.2: SR-T for the Hill equation, with `seed=102`.

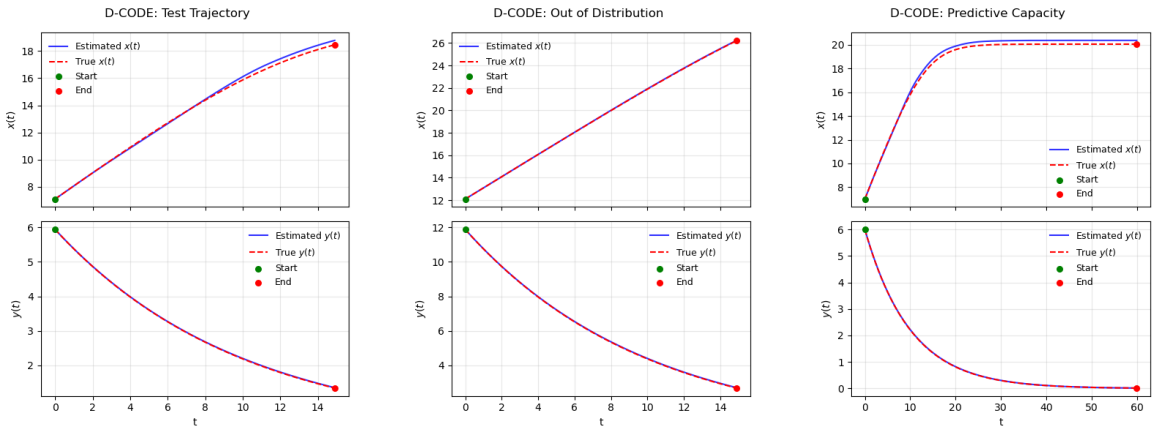
Hill – D-CODE. Here are reported the hyperparameters used and the result of each experiment using D-CODE for the discovery of the Hill equation.

Parameter	Description	Value
n_{gen}	Maximum number of generations	20
$\mathcal{F}$	Function set	$\{+, -1, \times, \div, \wedge\}$
$\mathcal{C}$	Constant range	$[1, 4]$
λ	Parsimony coefficient	0.01

Table A.5: Hill – D-CODE hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
100	1	1	0.049	230.641	155.171
101	0	1	0.061	239.690	175.415
102	1	1	0.036	233.689	222.623
103	0	1	0.094	207.546	194.723
104	1	1	0.072	210.417	222.983
105	1	1	0.077	189.189	162.068
106	1	1	0.086	241.674	127.113
107	1	1	0.053	263.079	221.854
108	0	1	0.084	173.088	292.144
109	1	1	0.044	168.163	236.448
110	1	1	0.048	217.637	214.528
111	1	1	0.060	196.542	197.702
112	1	1	0.058	252.938	199.818
113	1	1	0.076	185.426	207.250
114	1	1	0.092	191.118	185.497
115	1	1	0.057	227.728	250.589
116	1	1	0.060	223.274	162.885
117	1	1	0.059	218.662	271.168
118	0	1	0.036	168.726	224.470
119	1	1	0.076	246.165	241.429

Table A.6: Hill – D-CODE results for different seeds.



(a) $(x_0, y_0) = (7, 6)$ (b) $(x_0, y_0) = (12, 12)$ (c) $(x_0, y_0) = (7, 6)$

Figure A.3: D-CODE for the Hill equation, with `seed=102`.

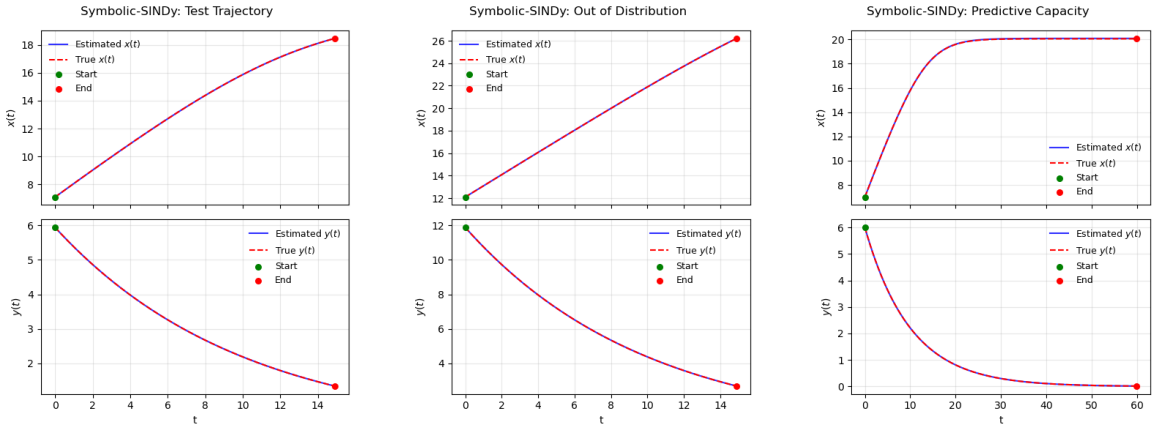
Hill – Symbolic-SINDy. Here are reported the hyperparameters used and the result of each experiment using Symbolic-SINDy for the discovery of the Hill equation.

Parameter	Description	Value
`method`	Symbolic regression method	`D-CODE`
`degree`	Degree of the polynomial library	3
$\otimes$	Tensor product library	`False`
n_{gen}	Maximum number of generations	20
$\mathcal{F}$	Function set	$\{+, -1, \times, \div, \wedge\}$
$\mathcal{C}$	Constant range	$[1, 4]$
λ	Parsimony coefficient	0.01
L	Sparsity threshold in STLSQ	0.05

Table A.7: Hill – Symbolic-SINDy hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
100	1	1	0.014	228.005	13.977
101	0	1	0.023	220.944	11.018
102	1	1	0.005	224.376	12.803
103	0	1	0.018	218.068	17.656
104	1	1	0.029	206.289	17.822
105	1	1	0.037	200.337	12.159
106	1	1	0.024	243.429	8.111
107	1	1	0.023	216.648	14.522
108	1	1	0.049	176.333	8.611
109	1	1	0.007	183.878	2.666
110	1	1	0.007	225.466	12.390
111	1	1	0.022	183.245	8.068
112	1	1	0.035	258.745	12.394
113	1	1	0.012	198.532	7.538
114	1	1	0.164	182.544	7.478
115	1	1	0.031	261.051	19.795
116	1	1	0.015	227.327	9.692
117	1	1	0.018	220.704	19.735
118	0	1	0.013	172.842	20.724
119	1	1	0.053	257.815	12.168

Table A.8: Hill – Symbolic-SINDy results for different seeds.



(a) $(x_0, y_0) = (7, 6)$ (b) $(x_0, y_0) = (12, 12)$ (c) $(x_0, y_0) = (7, 6)$

Figure A.4: Symbolic-SINDy for the Hill equation, `seed=102`.

Hill k-parametrized – SINDy. Here are reported the hyperparameters used and the result of each experiment using SINDy for the discovery of the Hill k-parametrized equation.

Parameter	Description	Value
`degree`	Degree of the polynomial library	3
L	Sparsity threshold in STLSQ	0.02

Table A.9: Hill k-parametrized – SINDy hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
–	0	1	0.279	$\simeq 0$	$\simeq 0$

Table A.10: Hill k-parametrized – SINDy results for different seeds.

Hill k-parametrized – SR-T. Here are reported the hyperparameters used and the result of each experiment using SR-T for the discovery of the Hill k-parametrized equation.

Parameter	Description	Value
n_{gen}	Maximum number of generations	20
$\mathcal{F}$	Function set	$\{+, -1, \times, \div, \wedge\}$
$\mathcal{C}$	Constant range	$[1, 4]$
λ	Parsimony coefficient	0.01

Table A.11: Hill k-parametrized – SR-T hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
100	1	1	0.078	321.314	285.395
101	0	1	0.345	226.920	329.569
102	1	1	0.016	318.705	299.035
103	1	1	0.238	208.899	247.838
104	1	1	0.029	317.097	262.329
105	0	1	0.104	228.562	314.334
106	1	1	0.331	299.903	342.970
107	0	1	0.060	360.751	318.173
108	1	1	0.065	353.957	253.201
109	1	1	0.275	199.195	302.935
110	0	1	0.099	252.363	328.334
111	0	1	0.122	317.771	287.552
112	1	1	0.014	338.329	328.774
113	1	1	0.140	310.282	264.774
114	1	1	0.014	303.804	302.505
115	0	1	0.349	237.551	300.861
116	1	1	0.059	305.479	267.303
117	0	1	0.027	301.410	318.153
118	0	1	0.337	274.615	309.497
119	1	1	0.023	306.523	319.188

Table A.12: Hill k-parametrized – SR-T results for different seeds.

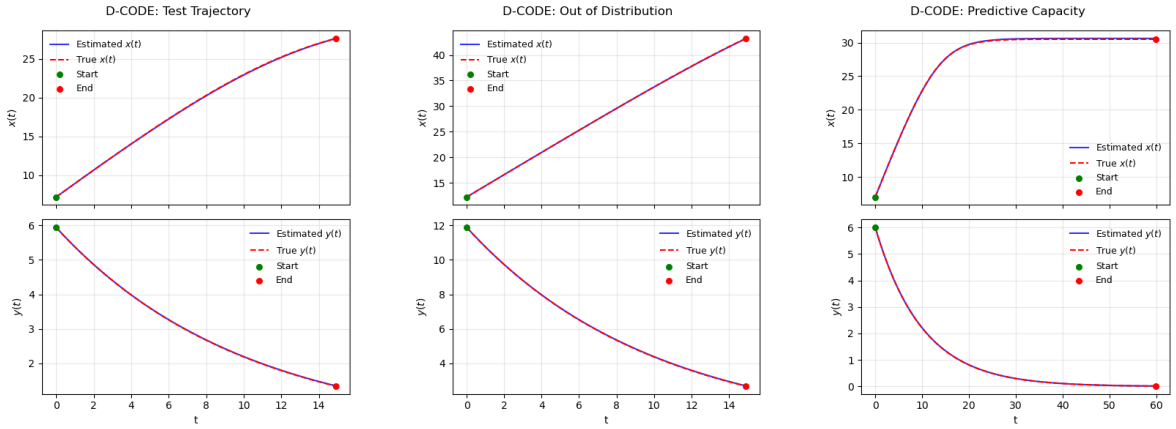
Hill k-parametrized – D-CODE. Here are reported the hyperparameters used and the result of each experiment using D-CODE for the discovery of the Hill k-parametrized equation.

Parameter	Description	Value
n_{gen}	Maximum number of generations	20
$\mathcal{F}$	Function set	$\{+, -1, \times, \div, \wedge\}$
$\mathcal{C}$	Constant range	$[1, 4]$
λ	Parsimony coefficient	0.01

Table A.13: Hill k-parametrized – D-CODE hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
100	0	1	0.037	203.169	207.635
101	1	1	0.061	213.625	177.891
102	1	1	0.016	207.991	221.309
103	0	1	0.098	202.229	223.018
104	1	1	0.061	219.225	209.442
105	0	1	0.032	221.896	216.473
106	0	1	0.072	252.423	209.170
107	1	1	0.056	229.853	189.968
108	0	1	0.052	216.568	189.738
109	1	1	0.065	206.788	181.901
110	1	1	0.075	215.451	211.813
111	0	1	0.022	255.633	183.040
112	1	1	0.020	188.040	227.417
113	1	1	0.041	217.725	216.629
114	0	1	0.091	206.878	223.227
115	0	1	0.051	246.528	251.070
116	0	1	0.087	323.413	263.670
117	1	1	0.059	234.306	213.664
118	1	1	0.037	226.266	185.653
119	0	1	0.075	214.860	258.966

Table A.14: Hill k-parametrized – D-CODE results for different seeds.



(a) $(x_0, y_0, k) = (7, 6, 1.8)$ (b) $(x_0, y_0, k) = (12, 12, 2.3)$ (c) $(x_0, y_0, k) = (7, 6, 1.8)$

Figure A.5: D-CODE for the k-parametrized Hill equation, with `seed=102`.

Hill k-parametrized – Symbolic-SINDy. Here are reported the hyperparameters used and the result of each experiment using Symbolic-SINDy for the discovery of the Hill k-parametrized equation.

Parameter	Description	Value
`method`	Symbolic regression method	`SR-T`
`degree`	Degree of the polynomial library	2
$\otimes$	Tensor product library	`False`
n_{gen}	Maximum number of generations	20
$\mathcal{F}$	Function set	$\{+, -1, \times, \div, \wedge\}$
$\mathcal{C}$	Constant range	$[1, 4]$
λ	Parsimony coefficient	0.01
L	Sparsity threshold in STLSQ	0.08

Table A.15: Hill k-parametrized – Symbolic-SINDy hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
100	1	1	0.036	332.161	11.853
101	0	1	0.291	195.258	7.439
102	1	1	0.006	335.955	13.047
103	1	1	0.193	267.975	8.646
104	1	1	0.020	398.060	14.886
105	1	1	0.074	259.437	7.663
106	1	1	0.187	294.952	9.866
107	1	1	0.039	309.440	13.392
108	1	1	0.037	321.821	16.978
109	1	1	0.167	209.922	9.867
110	1	1	0.037	253.150	8.015
111	0	1	0.067	326.973	28.477
112	1	1	0.012	312.982	8.532
113	1	1	0.140	318.496	12.057
114	1	1	0.017	343.046	33.415
115	0	1	0.263	260.963	8.421
116	1	1	0.046	279.909	12.585
117	1	1	0.035	329.216	17.716
118	0	1	0.338	260.126	3.680
119	1	1	0.024	315.331	11.477

Table A.16: Hill k-parametrized – Symbolic-SINDy results for different seeds.

A.2. Revised van der Pol oscillator: supplementary information

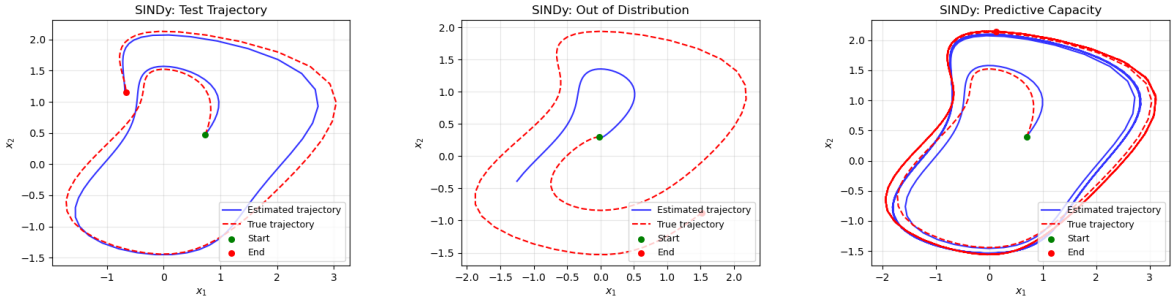
Revised van der Pol – SINDy. Here are reported the hyperparameters used and the result of each experiment using SINDy for the discovery of the revised van der Pol oscillator.

Parameter	Description	Value
`degree`	Degree of the polynomial library	3
L	Sparsity threshold in STLSQ	0.4

Table A.17: Revised van der Pol – SINDy hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
–	0	1	0.534	$\simeq 0$	$\simeq 0$

Table A.18: Revised van der Pol – SINDy results for different seeds.



(a) $(x_0, y_0) = (0.7, 0.4)$ (b) $(x_0, y_0) = (0.0, 0.3)$ (c) $(x_0, y_0) = (0.7, 0.4)$

Figure A.6: SINDy for the revised van der Pol oscillator.

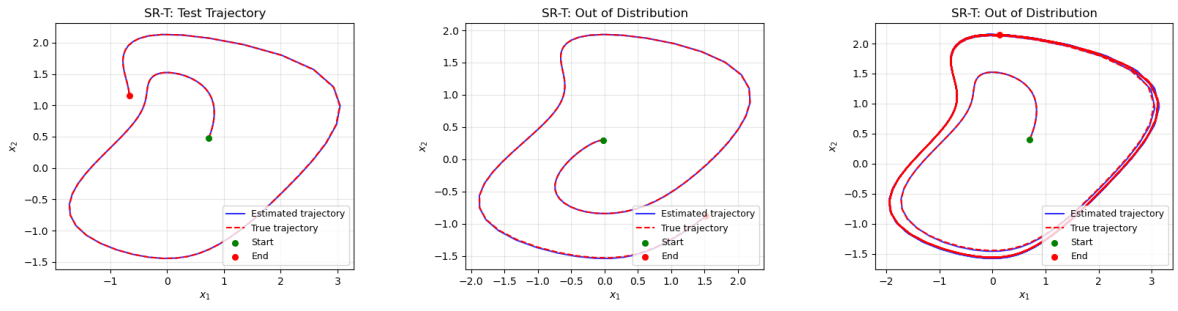
Revised van der Pol – SR-T. Here are reported the hyperparameters used and the result of each experiment using SR-T for the discovery of the revised van der Pol oscillator.

Parameter	Description	Value
n_{gen}	Maximum number of generations	20
$\mathcal{F}$	Function set	$\{+, -, \times, -1, \wedge, \sin, \cos\}$
$\mathcal{C}$	Constant range	$[1, 5]$
λ	Parsimony coefficient	0.01

Table A.19: Revised van der Pol – SR-T hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
100	0	1	0.373	293.963	147.389
101	0	1	0.690	219.810	130.255
102	1	1	0.007	313.613	169.209
103	1	1	0.007	323.977	140.760
104	1	1	0.007	308.161	191.393
105	0	1	0.690	273.034	142.890
106	0	1	0.271	291.958	233.033
107	1	1	0.007	313.121	136.724
108	0	1	0.279	287.381	167.633
109	1	1	0.018	279.969	151.845
110	0	1	0.690	180.599	119.071
111	1	1	0.007	227.806	127.608
112	0	1	0.258	290.477	141.911
113	1	1	0.007	231.750	158.223
114	0	1	1.194	282.228	123.922
115	0	1	0.690	202.733	140.210
116	1	1	0.007	249.080	115.164
117	0	1	0.326	314.648	141.341
118	0	1	0.278	325.256	210.892
119	0	1	0.284	290.597	168.719

Table A.20: Revised van der Pol – SR-T results for different seeds.



(a) $(x_0, y_0) = (0.7, 0.4)$ (b) $(x_0, y_0) = (0.0, 0.3)$ (c) $(x_0, y_0) = (0.7, 0.4)$

Figure A.7: SR-T for the revised van der Pol oscillator, `seed=103`.

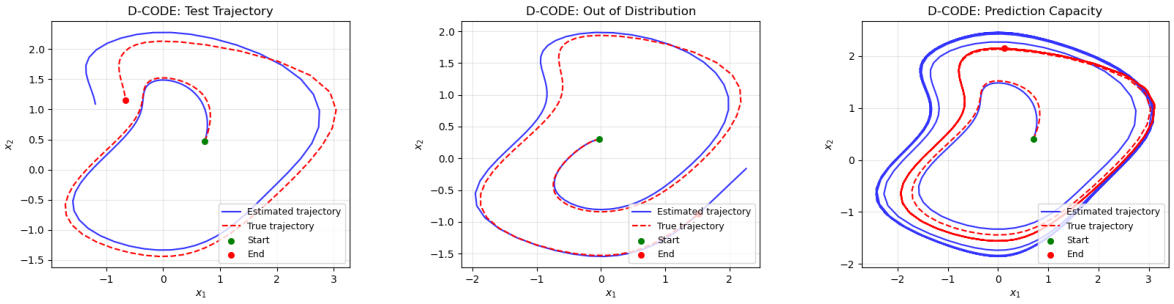
Revised van der Pol – D-CODE. Here are reported the hyperparameters used and the result of each experiment using D-CODE for the discovery of the revised van der Pol oscillator.

Parameter	Description	Value
n_{gen}	Maximum number of generations	20
$\mathcal{F}$	Function set	$\{+, -, \times, -1, \wedge, \sin, \cos\}$
$\mathcal{C}$	Constant range	$[1, 5]$
λ	Parsimony coefficient	2

Table A.21: Revised van der Pol – D-CODE hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
100	0	1	0.694	240.906	148.622
101	0	1	0.689	227.977	101.665
102	1	1	0.007	193.674	141.180
103	0	1	0.277	215.333	192.930
104	0	1	0.656	256.306	150.852
105	0	1	0.689	213.710	109.559
106	1	1	0.007	282.938	95.120
107	1	1	0.007	222.599	77.839
108	0	1	0.470	259.155	124.346
109	0	1	0.689	179.125	75.751
110	0	1	0.559	252.799	103.547
111	1	1	0.007	218.744	72.867
112	0	1	0.391	192.276	109.791
113	1	1	0.007	193.182	50.463
114	1	1	0.014	206.690	105.997
115	0	1	0.337	224.142	119.577
116	1	1	0.007	181.553	92.820
117	0	1	0.689	117.666	60.484
118	0	1	0.317	167.059	62.415
119	0	1	0.689	231.601	119.710

Table A.22: Revised van der Pol – D-CODE results for different seeds.



(a) $(x_0, y_0) = (0.7, 0.4)$ (b) $(x_0, y_0) = (0.0, 0.3)$ (c) $(x_0, y_0) = (0.7, 0.4)$

Figure A.8: D-CODE for the revised van der Pol oscillator, `seed=103`.

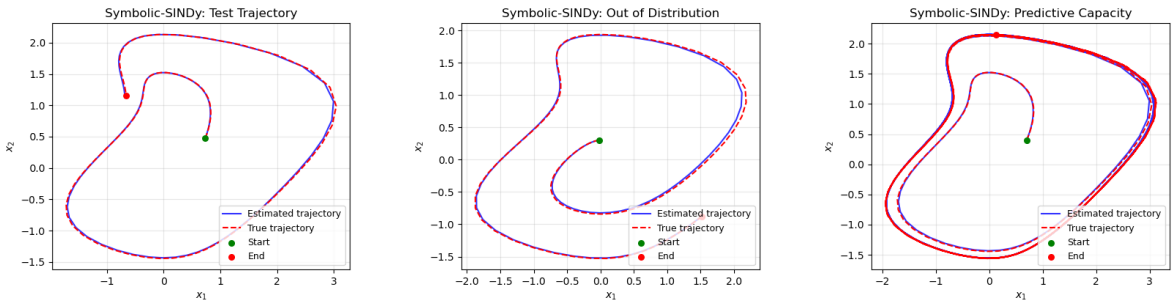
Revised van der Pol – Symbolic-SINDy. Here are reported the hyperparameters used and the result of each experiment using Symbolic-SINDy for the discovery of the revised van der Pol oscillator.

Parameter	Description	Value
`method`	Symbolic regression method	`SR-T`
`degree`	Degree of the polynomial library	3
$\otimes$	Tensor product library	`False`
n_{gen}	Maximum number of generations	6
$\mathcal{F}$	Function set	$\{+, -, \times, -1, \wedge, \sin, \cos, \log\}$
$\mathcal{C}$	Constant range	$[1, 5]$
λ	Parsimony coefficient	0.01
L	Sparsity threshold in STLSQ	0.1

Table A.23: Revised van der Pol – Symbolic-SINDy hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
100	1	1	0.031	106.697	16.232
101	1	1	0.031	110.443	67.400
102	0	1	0.229	110.905	21.496
103	1	1	0.031	111.391	21.243
104	1	1	0.031	106.307	35.637
105	1	1	0.031	103.486	44.566
106	1	1	0.031	109.912	27.090
107	0	1	0.575	108.949	27.711
108	1	1	0.031	107.897	19.625
109	1	1	0.031	105.185	24.952
110	1	1	0.031	111.625	52.799
111	1	1	0.031	105.354	15.502
112	1	1	0.031	102.788	7.965
113	1	1	0.031	109.542	21.497
114	0	1	0.948	104.890	8.406
115	1	1	0.031	102.005	57.644
116	1	1	0.031	107.146	34.151
117	0	1	0.229	106.142	27.189
118	1	1	0.031	99.915	34.946
119	1	1	0.031	106.094	17.257

Table A.24: Revised van der Pol – Symbolic-SINDy results for different seeds.



(a) $(x_0, y_0) = (0.7, 0.4)$ (b) $(x_0, y_0) = (0.0, 0.3)$ (c) $(x_0, y_0) = (0.7, 0.4)$

Figure A.9: Symbolic-SINDy for the revised van der Pol oscillator, `seed=103`.

Revised van der Pol ω-parametrized – SINDy. Here are reported the hyperparameters used and the result of each experiment using SINDy for the discovery of the revised van der Pol ω-parametrized oscillator.

Parameter	Description	Value
`degree`	Degree of the polynomial library	3
L	Sparsity threshold in STLSQ	0.05

Table A.25: Revised van der Pol ω-parametrized – SINDy hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
–	0	1	0.840	$\simeq 0$	$\simeq 0$

Table A.26: Revised van der Pol ω-parametrized – SINDy results for different seeds.

Revised van der Pol ω-parametrized – SR-T. Here are reported the hyperparameters used and the result of each experiment using SR-T for the discovery of the revised van der Pol ω-parametrized oscillator.

Parameter	Description	Value
n_{gen}	Maximum number of generations	20
$\mathcal{F}$	Function set	$\{+, -, \times, -1, \sin\}$
$\mathcal{C}$	Constant range	$[1, 5]$
λ	Parsimony coefficient	0.004

Table A.27: Revised van der Pol ω-parametrized – SR-T hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
100	0	1	0.629	309.437	192.842
101	0	1	0.508	285.636	199.788
102	0	1	0.508	245.533	146.213
103	0	1	0.903	298.994	158.550
104	0	1	0.508	215.404	174.233
105	1	1	0.007	420.170	237.257
106	1	1	0.007	390.550	122.035
107	0	1	0.635	282.275	160.550
108	0	1	0.508	291.363	155.450
109	1	1	0.007	305.836	144.451
110	1	1	0.007	340.891	141.541
111	1	1	0.007	323.047	126.566
112	0	1	0.635	264.371	166.296
113	0	1	0.751	326.716	190.646
114	0	1	0.693	325.428	130.050
115	0	1	0.925	272.713	145.719
116	1	1	0.007	322.387	160.810
117	1	1	0.007	279.158	196.354
118	0	1	0.630	321.650	187.345
119	1	1	0.007	279.626	122.324

Table A.28: Revised van der Pol ω-parametrized – SR-T results for different seeds.

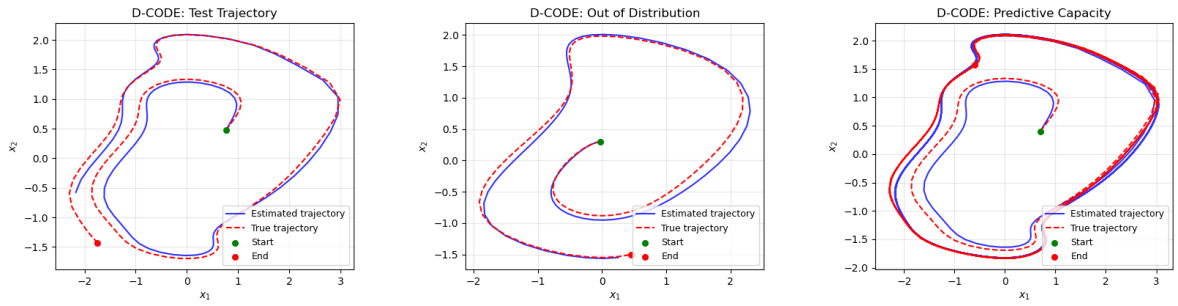
Revised van der Pol ω-parametrized – D-CODE. Here are reported the hyperparameters used and the result of each experiment using D-CODE for the discovery of the revised van der Pol ω-parametrized oscillator.

Parameter	Description	Value
n_{gen}	Maximum number of generations	20
$\mathcal{F}$	Function set	$\{+, -, \times, -1, \sin\}$
$\mathcal{C}$	Constant range	$[1, 5]$
λ	Parsimony coefficient	0.5

Table A.29: Revised van der Pol ω-parametrized – D-CODE hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
100	0	1	0.949	238.072	133.209
101	0	1	0.906	239.992	112.730
102	0	1	0.250	224.815	176.860
103	0	1	0.464	204.684	97.040
104	0	1	0.165	227.957	103.555
105	0	1	0.940	236.295	79.127
106	0	1	0.960	226.077	160.223
107	0	1	0.955	237.093	88.008
108	0	1	0.885	282.435	103.371
109	1	1	0.007	224.728	97.145
110	0	1	1.263	165.925	95.509
111	0	1	1.008	233.617	88.718
112	0	1	0.810	236.018	123.362
113	0	1	0.939	234.623	69.758
114	0	1	0.909	203.277	72.707
115	0	1	0.955	269.044	82.355
116	0	1	0.961	229.137	113.642
117	0	1	0.805	234.093	76.205
118	0	1	0.931	145.164	74.541
119	0	1	0.766	305.553	90.888

Table A.30: Revised van der Pol ω-parametrized – D-CODE results for different seeds.



(a) $(x_0, y_0, \omega) = (0.7, 0.4, 2.5)$ (b) $(x_0, y_0, \omega) = (0.0, 0.3, 0.8)$ (c) $(x_0, y_0, \omega) = (0.7, 0.4, 2.5)$

Figure A.10: D-CODE for the ω-parametrized revised van der Pol oscillator, `seed=103`.

Revised van der Pol ω-parametrized – Symbolic-SINDy. Here are reported the hyperparameters used and the result of each experiment using Symbolic-SINDy for the discovery of the revised van der Pol oscillator.

Parameter	Description	Value
`method`	Symbolic regression method	`SR-T`
`degree`	Degree of the polynomial library	3
$\otimes$	Tensor product library	`False`
n_{gen}	Maximum number of generations	11
$\mathcal{F}$	Function set	$\{+, -, \times, -1, \sin, \log\}$
$\mathcal{C}$	Constant range	$[1, 5]$
λ	Parsimony coefficient	0.004
L	Sparsity threshold in STLSQ	0.1

Table A.31: Revised van der Pol ω-parametrized – Symbolic-SINDy hyperparameters.

Seed	Succ. 1	Succ. 2	RMSE	Time 1	Time 2
100	0	1	0.818	198.726	36.312
101	0	1	0.827	188.648	123.648
102	1	1	0.201	181.445	143.979
103	1	1	0.201	187.779	121.617
104	1	1	0.201	189.906	125.772
105	0	1	0.831	200.425	77.114
106	0	1	0.599	188.830	127.506
107	1	1	0.201	220.731	162.468
108	1	1	0.201	175.015	19.400
109	0	1	0.831	184.649	142.988
110	0	1	0.831	159.765	144.876
111	0	1	0.831	184.804	44.284
112	1	1	0.201	198.848	119.018
113	1	1	0.201	202.411	36.935
114	0	1	0.831	178.395	81.279
115	0	1	0.831	186.543	62.527
116	0	1	0.830	184.690	92.066
117	0	1	0.609	189.606	122.850
118	1	1	0.201	184.914	147.040
119	0	1	0.831	198.792	77.784

Table A.32: Revised van der Pol ω-parametrized – Symbolic-SINDy results for different seeds.

B | Appendix B

B.1. Forced van der Pol: supplementary information

Online Symbolic-SINDy hyperparameters. Here are reported the hyperparameters of Online Symbolic-SINDy for the discovery of the time-varying forced van der Pol oscillator.

Parameter	Description	Value
	Initial SINDy configuration	
`degree`	Degree of the polynomial library	3
L	Sparsity threshold in STLSQ	0.4
	Symbolic-SINDy configuration	
`method`	Symbolic regression method	`SR-T`
`degree`	Degree of the polynomial library	3
$\otimes$	Tensor product library	`True`
n_{gen}	Maximum number of generations	11
$\mathcal{F}$	Function set	$\{+, -, \times, -1, \cos\}$
$\mathcal{C}$	Constant range	$[3, 5]$
λ	Parsimony coefficient	0.01
L	Sparsity threshold in STLSQ	0.3
W_{SR}	Symbolic regression moving window	10
W_{SINDy}	SINDy moving window	15
`patience`	Change point before execute again SR	10
	Change-point configuration	
N	CUSUM calibration interval	8
c_1	CUSUM threshold	1
ϵ	CUSUM tolerance error	0.1
c_2	Maximum model error	1.36

Table B.1: Time-varying forced Van der Pol – Online Symbolic-SINDy hyperparameters.

B.2. Non-autonomous Lotka-Volterra: supplementary information

Online Symbolic-SINDy hyperparameters. Here are reported the hyperparameters of Online Symbolic-SINDy for the discovery of the non-autonomous Lotka-Volterra model.

Parameter	Description	Value
Initial SINDy configuration		
`degree`	Degree of the polynomial library	2
L	Sparsity threshold in STLSQ	0.07
Symbolic-SINDy configuration		
`method`	Symbolic regression method	`D-CODE`
`degree`	Degree of the polynomial library	2
$\otimes$	Tensor product library	`True`
n_{gen}	Maximum number of generations	14
$\mathcal{F}$	Function set	$\{+, -, \times, -1, \sin\}$
$\mathcal{C}$	Constant range	$[0.01, 6]$
λ	Parsimony coefficient	0.5
L	Sparsity threshold in STLSQ	0.07
W_{SR}	Symbolic regression moving window	10
W_{SINDy}	SINDy moving window	10
`patience`	Change point before execute again SR	6
Change-point configuration		
N	CUSUM calibration interval	10
c_1	CUSUM threshold	1
ϵ	CUSUM tolerance error	0.1
c_2	Maximum model error	0.8

Table B.2: Non-autonomous Lotka-Volterra – Online Symbolic-SINDy hyperparameters.

C | Appendix C

C.1. Double gyre flow: supplementary information

CAE+SINDy hyperparameters. Here are reported the architecture and the hyperparameters used for the discovery of the double gyre flow problem, when SINDy is applied at latent level.

Parameter	Description	Value
`input_shape`	Input shape of the CAE	$(2, 64, 32)$
`filter_widths`	Number of convolutional filters per layer	$[16,\ 32]$
`latent_dim`	Dimension of the latent space	1
$\boldsymbol{\Theta}(z)$	SINDy latent library	$[1,\ z,\ z^2,\ z^3,\ \sin(z)]$
λ_1	SINDy penalty in $\dot{\mathbf{x}}$	10^{-3}
λ_2	SINDy penalty in $\dot{\mathbf{z}}$	10^{-3}
λ_3	L_1 penalty on $\boldsymbol{\Xi}$	10^{-6}
`sequential_thresholding`	Sequential thresholding activation	`True`
`threshold_frequency`	Thresholding frequency (epochs)	100
`coefficient_threshold`	Threshold value for SINDy coefficients	10^{-2}
`activation`	Activation function	`tanh`
`learning_rate`	Learning rate	10^{-3}
`batch_size`	Batch size	32
`max_epochs`	Number of training epochs	1001
`refinement_epochs`	Number of refinement epochs	201

Table C.1: CAE+SINDy architecture and hyperparameters.

CAE+Symbolic-SINDy hyperparameters. Here are reported the architecture and the hyperparameters used for the discovery of the double gyre flow problem, when Symbolic-SINDy is applied at latent level.

Parameter	Description	Value
`input_shape`	Input shape of the CAE	$(2, 64, 32)$
`filter_widths`	Number of convolutional filters per layer	$[16,\ 32]$
`latent_dim`	Dimension of the latent space	1
$\boldsymbol{\Theta}(z)$	SINDy latent library	$[1,\ z,\ z^2,\ z^3,\ \sin(z)]$
λ_1	SINDy penalty in $\dot{\mathbf{x}}$	10^{-3}
λ_2	SINDy penalty in $\dot{\mathbf{z}}$	0
λ_3	L_1 penalty on $\boldsymbol{\Xi}$	10^{-6}
`sequential_thresholding`	Sequential thresholding activation	`True`
`threshold_frequency`	Thresholding frequency (epochs)	100
`coefficient_threshold`	Threshold value for SINDy coefficients	10^{-2}
`activation`	Activation function	`tanh`
`learning_rate`	Learning rate	10^{-3}
`batch_size`	Batch size	32
`max_epochs`	Number of training epochs	201
`refinement_epochs`	Number of refinement epochs	201

Table C.2: CAE+Symbolic-SINDy initial architecture and hyperparameters.

Parameter	Description	Value
`method`	Symbolic regression method	`D-CODE`
`degree`	Degree of the polynomial library	3
$\otimes$	Tensor product library	`False`
n_{gen}	Maximum number of generations	11
$\mathcal{F}$	Function set	$\{+, -, \times, -1, \sin, \cos\}$
$\mathcal{C}$	Constant range	$[1, 6]$
λ	Parsimony coefficient	1
L	Sparsity threshold in STLSQ	0.1

Table C.3: Revised van der Pol ω-parametrized – Symbolic-SINDy hyperparameters.

Parameter	Description	Value
`input_shape`	Input shape of the CAE	$(2, 64, 32)$
`filter_widths`	Number of convolutional filters per layer	$[16,\ 32]$
`latent_dim`	Dimension of the latent space	1
$\boldsymbol{\Theta}(z, t)$	SINDy latent library	$[1,\ z,\ z^2,\ z^3,\ \cos(4.998\, t)]$
λ_1	SINDy penalty in $\dot{\mathbf{x}}$	10^{-3}
λ_2	SINDy penalty in $\dot{\mathbf{z}}$	10^{-2}
λ_3	L_1 penalty on $\boldsymbol{\Xi}$	10^{-6}
`sequential_thresholding`	Sequential thresholding activation	`True`
`threshold_frequency`	Thresholding frequency (epochs)	100
`coefficient_threshold`	Threshold value for SINDy coefficients	10^{-2}
`activation`	Activation function	`tanh`
`learning_rate`	Learning rate	10^{-3}
`batch_size`	Batch size	32
`max_epochs`	Number of training epochs	1001
`refinement_epochs`	Number of refinement epochs	201

Table C.4: CAE+Symbolic-SINDy final architecture and hyperparameters.

List of Figures

List of Tables

List of Symbols

Variable	Description	Dimension
$\mathbf{x}(t)$	state vector at time t	$\mathbb{R}^n$
$\dot{\mathbf{x}}(t)$	time derivative of the state vector at time t	$\mathbb{R}^n$
$\mathbf{f}$	equation of motion of a dynamical system	–
$\boldsymbol{\beta}$	vector of parameters of the dynamical system	$\mathbb{R}^p$
$\mathcal{X}$	symbolic state variables set	–
$\mathcal{F}$	functions set	–
$\mathcal{C}$	constants set	–
n_{gen}	maximum number of generations for a symbolic regression run	$\mathbb{N}$
λ	parsimony coefficient for a symbolic regression run	$\mathbb{R}_+$
$\mathbf{X}$	snapshot matrix of the state vectors	$\mathbb{R}^{N_t \times n}$
$\dot{\mathbf{X}}$	snapshot matrix of time derivative of the state vectors	$\mathbb{R}^{N_t \times n}$
$\boldsymbol{\Theta}$	SINDy nonlinear candidate function library	$\mathbb{R}^l$
$\boldsymbol{\Theta}(\mathbf{X})$	SINDy design matrix	$\mathbb{R}^{Nt \times l}$
$\boldsymbol{\Xi}$	SINDy coefficient matrix	$\mathbb{R}^{l \times n}$
$\boldsymbol{\xi}_i$	i-th column of the SINDy coefficient matrix	$\mathbb{R}^d$
L	SINDy threshold for STLSQ optimization	$\mathbb{R}_+$
$\otimes$	boolean tensor product library	–
$\mathbf{z}(t)$	latent state vector at time t	$\mathbb{R}^r$
$\dot{\mathbf{z}}(t)$	time derivative of the latent state vector at time t	$\mathbb{R}^r$
$\phi(\cdot)$	encoder map	–
$\psi(\cdot)$	decoder map	–
λ_1	SINDy penalty in $\dot{\mathbf{x}}$	$\mathbb{R}$
λ_2	SINDy penalty in $\dot{\mathbf{z}}$	$\mathbb{R}$
λ_3	L_1 penalty in $\boldsymbol{\Xi}$	$\mathbb{R}$

Acknowledgements

Desidero esprimere la mia sincera gratitudine al Prof. Andrea Manzoni per avermi dato l'opportunità di lavorare su un argomento che mi ha fin da subito affascinato, permettendomi di mettermi in gioco e di ampliare le mie conoscenze. Lo ringrazio per la supervisione, per la fiducia e per la libertà concessami nello sviluppo di questa tesi. Il suo entusiasmo è stato uno stimolo costante per provare a dare quel qualcosa in più, ed è stato determinante nei momenti di smarrimento.
Vorrei inoltre ringraziare il Dott. Paolo Conti per i preziosi consigli, che si sono rivelati fondamentali per lo sviluppo di questo lavoro.

Ringrazio il Politecnico di Milano, per la formazione le opportuntà offerte.
Ringrazio le persone che ho incrociato in questi anni, con il quale ho condivisio esperienze, conoscienze e conversazioni di valore.

Ringrazio la mia famiglia, che alle mie spalle mi ha sempre supportato.
Sono profondamente grato ai miei genitori, che mi hanno dato la possibilità di studiare ciò che mi ha sempre incuriosito e appassionato, senza mai mettermi pressione o fretta, e senza mai dubitare delle mie scelte. I loro sacrifici e il loro interesse genuino nella mia formazione mi hanno permesso di affrontare questo percorso con la massima serenità.
Ringrazio infine mio fratello Roberto, con il quale ho condiviso questo cammino. La sua presenza è stata per me una sicurezza.

Tutto questo non l'ho mai dato per scontato, e ho sempre cercato di ripagarlo con il massimo impegno.