Algebraic Design of Physical Computing System for Time-Series Generation

Mizuka Komatsu
Graduate School of System Informatics
Kobe University
Kobe, Japan, 657-8501
m-komatsu@bear.kobe-u.ac.jp

Takaharu Yaguchi
Graduate School of Science
Kobe University
Kobe, Japan, 657-8501
yaguchi@pearl.kobe-u.ac.jp

Kohei Nakajima
Graduate School of Information Science and Technology
The University of Tokyo
Tokyo, Japan, 113-0033
k-nakajima@isi.imi.i.u-tokyo.ac.jp

Abstract

Recently, computational techniques that employ physical systems (physical computing systems) have been developed. To utilize physical computing systems, their design strategy is important. Although there are practical learning based methods and theoretical approaches, no general method exists that provides specific design guidelines for given systems with rigorous theoretical support. In this paper, we propose a novel algebraic design framework for a physical computing system for time-series generation, which is capable of extracting specific design guidelines. Our approach describes input-output relationships algebraically and relates them to this task. We present two theorems and the results of experiments. The first theorem offers a basic strategy for algebraic design. The second theorem explores the "replaceability" of such systems.

1 Introduction

Recent developments in sensing and Internet of Things technology require enormous complex time series data to be processed efficiently in real-time. The von Neumann architecture, which is a conventional computational architecture that aligns the processor and memory separately, causes an intrinsic limitation in processing speed, the von Neumann bottleneck. To design a successful real-time information processing method with lower computational and energy costs, it is necessary to reconsider the computational architecture and concept [9]. Therefore, many approaches based on the non-von Neumann type architectures have recently been proposed, capitalizing on the power of material properties [27, 11] and the diverse dynamics of physical systems [18]. Physical systems outsource the computational load to natural physical dynamics and add some properties in addition to computation, such as robustness against a radioactive environment [12, 1], which significantly extends the application domains. In this paper, the term systems refers to physical computing systems, and we propose a design principle for such systems, in particular, for time-series generation based on the theory of algebra.

We focus on mapping input sequences to output sequences, which we consider computing in this paper. Specifically, we examine physical computing, where a physical system characterized by its parameters is employed to map an input sequence (Fig. 1). In the research field of physical computing, designing a system that exhibits desirable computing has been a focus of attention [26, 20, 7, 15]. Common and practical methods include simulation- [26] and learning-based methods [20]. Although the efficacy of such methods has been partially confirmed, they basically rely on an enormous number of trial and error attempts or simulations, and/or it is necessary to learn their input-output relationships,

37th First Workshop on Machine Learning with New Compute Paradigms at NeurIPS 2023(MLNPCP 2023).

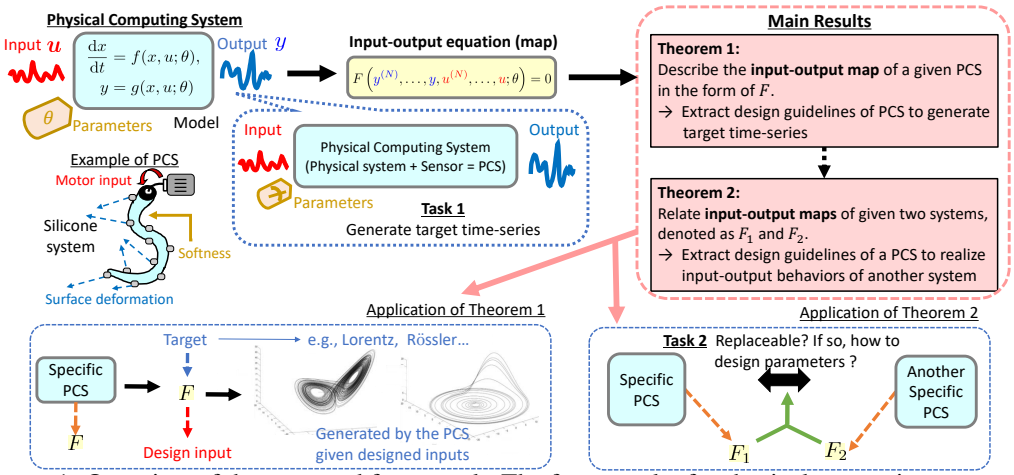

Figure 1: Overview of the proposed framework. The framework of a physical computing system is depicted at the upper left. Theorems on the algebraic design of physical computing systems are the main results. Their applications, which are illustrated at the bottom, are shown in Section 3.

which are not supported by theories. On one hand, researchers [7, 15] have attempted to understand the design principle of physical computing systems, which are based on the theory of nonlinear filter approximation [2]. In these studies, computational tasks are reduced to the approximation of a class of filters by the systems. In other words, each filter in the class has the role of mapping an input stream to an output stream. Then, sufficient conditions for the approximation such as the echo-state property [8, 16], fading memory property and point separation property [7, 2] are provided. The practical use of such theories can be challenging. Specifically, there is a lack of methods for relating the specific physical systems under consideration to the conditions. Hence, no specific design guidelines for physical computing systems are available.

Based on the above, we aim to establish a theoretical framework for designing physical computing systems that can offer specific design guidelines. We assume that a physical computing system is modeled as a state-space model and examine its specific design guidelines based on algebra. An overview of our framework is depicted in Fig. 1. Our approach not only achieves our research aim but also reveals the potential of physical computing systems. In this framework, the specific input-output relationships of a given system are first described ($F = 0$ in Fig. 1.) Then, F is related to the computational tasks (Theorems 1 and 2). To be more precise, the input-output relationships are characterized by the Gröbner basis, which is an algebraic technique. This makes it possible to describe the relationships and obtain design guidelines using computer algebra software without being aware of the underlying algebraic theory. Theorem 1 outlines a fundamental design strategy for computational tasks. In this strategy, a physical system generates the desired time-series after its inputs or parameters are adjusted, which is a process that is often conducted empirically in practical methods. Theorem 2 extends Theorem 1 by addressing situations where one physical computing system might be preferable over another, considering the "replaceability" of systems. As an example, such a situation may arise as a result of the emergence of diverse implementations of physical computing. Section 3demonstrates the effectiveness of Theorem 2, which provides specific guidelines for replacing one system with another. The limitations of our framework are discussed in Section 4.

2 Proposed framework

2.1 Notations and basic assumptions

In this study, we assume that the physical computing system is modeled by a system of differential equations with an observation function as follows:

$$\dot{x} = f(x,u;\theta), \quad y = g(x,u;\theta). \tag{1}$$

where $f : \mathbb{R}^{N+M} \to \mathbb{R}^N$, $g : \mathbb{R}^{N+M} \to \mathbb{R}$. $x \in \mathbb{R}^N$ is the state variable vector of the physical computing system; $u \in \mathbb{R}^M$ is the input vector; $y \in \mathbb{R}^M$ is the output variable; $\theta \in \mathbb{R}^n$ is the parameter vector. Let us denote the initial state variable vector as $x(0) \in \mathbb{R}^N$. The former equation of (1) describes the physical system of interest. We assume the existence and uniqueness of its solution. An observation equation, which is the latter equation of (1), describes how to observe the output of the system. In the field of reservoir computing [19], this is commonly referred to as the readout, where the outputs of (1) correspond to the outputs from readouts equipped with a physical

reservoir. Throughout this paper, we assume that f and g are polynomials of x, u, whose coefficients are rational functions of θ; $f, g \in \mathbb{C}(\theta)[x, u]$. Hereafter, symbols with a hat denote outputs from the physical computing system and those with a bar or tilde denote the given time-series. We denote $\mathrm{d}^N x/\mathrm{d}t^N$ as $x^{(N)}$ and $\mathrm{d}x/\mathrm{d}t$ as $\dot{x}$, depending on the context.

To design physical computing system, we first consider whether the output of a given system generates target time-series $\bar{y}(t)$. This can be regarded as a machine learning task, which is typically conducted using a method such as a recurrent neural network.

Target task 1 (time-series generation) *Let us denote output $y(t)$ of* (1) *as $\hat{y}(t; \theta, u, x(0))$ given $\theta \in \Theta, x(0) \in \mathbb{R}^N$, and u in a set of functions, $\mathbb{R} \to \mathbb{R}^M$, denoted as U. Suppose that we have a target time-series denoted as $\bar{y}(t)$ over a time duration. A time-series generation task is to find $(\theta, u, x(0)) \in \mathbb{R}^n \times U \times \mathbb{R}^N$ such that $\hat{y}(t; \theta, u, x(0)) = \bar{y}(t)$ holds over the specified domain.*

Although this is a common task in the field of machine learning, our approach is different from learning-based design, as explained in Section 1. The following extended task can be successfully handled using our algebraic approaches, as shown in Section 2.4.

Target task 2 (designing a system and exploiting its replaceability with another system) *Suppose that we have two systems described by state-space models such as* (1)*. The first is a system to be designed, whereas the other is the target. With common inputs $u \in U$, their output variables are denoted as $\hat{y}(t; \theta, u, x(0))$ and $\bar{y}(t; \tilde{\theta}, u, \tilde{x}(0))$ given initial states $x(0), \tilde{x}(0) \in \mathbb{R}^N$ and parameter vectors $\theta \in \mathbb{R}^n, \tilde{\theta} \in \mathbb{R}^{\tilde{n}}$, respectively. In this task, $(\theta, u, x(0)) \in \mathbb{R}^n \times U \times \mathbb{R}^N$ such that $\hat{y}(t; \theta, u, x(0)) = \bar{y}(t; \tilde{\theta}, u, \tilde{x}(0))$ holds over the specified duration is to be found if it exists. Note that if such $(\theta, u, x(0))$ exists, the system is said to be replaceable with respect to the target system.*

We make two assumptions regarding target time-series $\bar{y}(t)$. The first relates to its smoothness (Assumption 1), which is assumed throughout this paper (Section 4.) The other assumption (Assumption 2) is introduced in Section 2.4 and is related to the replaceability.

Assumption 1 *Target time-series $\bar{y}(t)$ is a sufficiently smooth function from $\mathbb{R}$ onto $\mathbb{R}$ over the duration under consideration.*

Assumption 2 *Target time-series $\bar{y}(t)$ satisfies $h(\bar{y}^{(N)}, \ldots, \bar{y}; \tilde{\theta}) = 0$, where h is a polynomial differential equations up to the Nth order of y given parameter vector $\tilde{\theta} \in \mathbb{R}^{\tilde{n}}$.*

2.2 Key ingredient in algebraic design: the input-output map of physical computing systems

To extract equations representing the input-output relations in the system modeled by (1), we consider eliminating its state variables. Because (1) contains derivatives of the state variables, such eliminations are realized through algebraic manipulations and derivations of the model described by (1). Here, the question is how many times we need to differentiate the models to extract equations representing the input-output relations, which may not be bounded. Proposition 1 (Appendix B), which is a theorem shown in [6], clarifies the order of the derivations needed for this. It supports the existence of maps describing the input-output relationships of a physical computing system. In terms of algebra, we regard the model as polynomials of the states, inputs, outputs, and their higher-order derivatives. If we consider derivations up to the 2nd order, a set of polynomials obtained by manipulating $\mathrm{d}x/\mathrm{d}t - f, \mathrm{d}^2x/\mathrm{d}t^2 - f^{(2)}, y - x, y^{(1)} - x^{(1)}$ is considered. Proposition 1 shows that polynomials representing the input-output relations are obtained through algebraic manipulations of a set of polynomials up to at least Nth order derivatives, where N is the dimension of the state vector of the system. Let us denote a set of polynomials obtained through algebraic manipulations of the model differentiated up to the Nth order as $I^{(N)}$ (Appendix B). Using this, polynomials in which the state variables and their derivatives are eliminated are described as follows:

$$I^{(N)} \cap \mathbb{C}(\theta)[y^{(N)}, \ldots, y, u^{(N)}, \ldots, u], \tag{2}$$

which is a set of polynomials representing the input-output relationships of the model (Fig. 1d). As (2) is shown to have a non-zero polynomial, the existence of the equations representing the input-output relations of the physical computing systems modeled by (1) is shown.

Next, we explain how to characterize (2), which represents the input-output relations of the model. In terms of algebra, this problem involves describing the ideal where the Gröbner basis is known to be a key (Appendix A.) Intuitively, (2) (an ideal) can be characterized by its Gröbner basis, as a basis characterizes a linear subspace in terms of linear algebra [3]. In particular, any polynomial equation corresponding to a polynomial in (2) holds if all the equations constituting the Gröbner basis for (2) hold. In this sense, the Gröbner basis for (2) explicitly describes the input-output relations for a given physical computing system. Notably, the Gröbner basis can be derived automatically using computer algebra software. The details on the Gröbner basis and its derivation using computer algebra

software are summarized in Appendix D. Furthermore, there is another benefit of using the Gröbner basis. The Gröbner basis can be used to determine whether a given differential equation belongs to (2). This allows us to compare different physical implementations of a physical computing system, which is explained in Section 2.4. The following equation, which is an element of the Gröbner basis, representing the input-output relationships of (1) is called the input-output equation (Fig. 1d):

$$F(y^{(N)}, \dots, y, u^{(N)}, \dots, u; \theta) = 0. \tag{3}$$

This is a polynomial differential equation up to the Nth order of u, y.

2.3 Main Theorem 1: Deriving design guidelines of physical computing systems algebraically

Theorem 1 is a fundamental theorem for deriving design guidelines from (3) for Target task 1. Intuitively, it provides a discriminator to determine whether a given physical computing system generates $\bar{y}(t)$, which reveals design guidelines for the system.

Theorem 1 *Let $\mathcal{F}_y$ and $\mathcal{F}_u$ denote sets of sufficiently smooth functions containing functions from $\mathbb{R}$ onto $\mathbb{R}$ and $\mathbb{R}^M$, respectively. Let us consider a physical computing system described by* (1) *and assume it is generic in terms of Definition 13. Suppose that there exists $\bar{y}(t) \in \mathcal{F}_y$, which is observed over $[0, T]$ (Assumption 1.) Suppose also that for* (1) *given $\bar{y}$, there exists a non-empty set of consistent initial values $X \subset \mathbb{R}^N$. Then, $\bar{y}(t)$ can be generated by* (1) *given $(\theta, u, x(0)) \in \Theta \times \mathcal{F}_u \times X$ if and only if*

$$F(\bar{y}^{(N)}(t), \dots, \bar{y}(t), u^{(N)}(t), \dots, u(t); \theta) = 0, \tag{4}$$

holds over $[0, T]$, where $\Theta \subset \mathbb{R}^n$ is a set of generic parameter vectors.

See Appendix E for the proof and related definitions. This guarantees that (3) plays a role in deriving the design guidelines of the system for the time-series generation task. There are two assumptions besides those mentioned in Section 2.1: the genericity of the physical computing system (Definition 13) and consistent initial conditions of the system corresponding to the given time-series (Definition 12). The first assumptions is weak, whereas the second assumption is of importance especially for non-stationary tasks. Generally, the consistent initial state $x(0)$ is one that satisfies $\hat{y}(t_1; \theta, u, x(0)) = \bar{y}(t_1)$ given arbitrary θ and u. The second assumption is critical for non-stationary tasks, whereas it is not for stationary ones. See Experiment 3 for examples.

Experiment: Designing inputs applied to physical computing systems We consider the physical system modeled using the FitzHugh-Nagumo equation [21],

$$\dot{x}_1 = x_1 (x_1 - 1)(1 - (1/3) x_1) - x_2 + u, \quad \dot{x}_2 = x_1. \tag{5}$$

We assume that x_2 is observed; $y = x_2$. Then, following the procedures in Appendix D, the input-output equation of this system, $y + \frac{1}{3}\dot{y}^3 - \frac{4}{3}\dot{y}^2 + \dot{y} + \ddot{y} - u = 0$ is obtained. According to Theorem 1, the input-output equation in which a target time-series is substituted, serves as a discriminant for generation tasks. Manipulating the discriminant, a design guideline on input

$$u = \bar{y} + \left(\dot{\bar{y}}^3/3\right) - \left(4\dot{\bar{y}}^2/3\right) + \dot{\bar{y}} + \ddot{\bar{y}}, \tag{6}$$

is derived. Using the input defined as (6) and given consistent initial states $(x_1(0), x_2(0))^\top = (\dot{\bar{y}}(0), \bar{y}(0))^\top$, the system generates the given time-series, $\bar{y}$. The guideline derived based on Theorem 1, (6), suggests that the system can generate several target time-series by adjusting its input while maintaining itself and its observation method consistent. See Appendix I for numerical examples. Notably, this property, which is a type of multi-tasking property, is desirable in physical computing, leading to the efficient use of given physical assets to perform various computations. See Appendix J for details on how these initial values are chosen.

2.4 Main Theorem 2: Relating two different physical computing systems

In this section, we provide a theorem that enables the design of physical computing systems that can be replaced with other computing systems. Such a design is useful considering the diverse implementations of physical systems (Target task 2.) The basic idea is to provide a description of the relationship between the input-output equations of a given physical computing system and a target system. Such a relationship provides information about whether those systems can be interchanged, as well as providing design guidelines. To accomplish this, we leverage another benefit of the Gröbner basis. The Gröbner basis (e.g., (3)) can determine whether a given polynomial in a set of polynomials corresponds to itself (e.g., (2)); a polynomial is shown to be or not to be in a certain ideal (Appendix A) through division by its Gröbner basis. Suppose that the input-output equation of a target system, which is denoted as h in Assumption 2, is set as the polynomial to be divided. Then, intuitively, the division by (3) makes it possible to determine whether the target input-output behavior is realized by a given physical computing system.

When performing division between the polynomials of a single variable, in general, we first arrange the terms of the polynomial in descending order of the degree of the variable. We then focus on the monomials whose variable degree is the highest in the polynomial to be divided and for the division. If the monomial for the polynomial to be divided is higher than or equal to that of the polynomial to divide, then the polynomial to be divided is factorized by the highest degree term of the polynomial to divide. Therefore, the order of monomials is critical in dividing polynomials of a single variable. This holds for the multivariate case as well. In general, the monomial ordering fixes the order of the monomials in the set of monomials of interest [3].

In this paper, we consider differential polynomial equations such as (3); therefore, we introduce an order of monomials, whose variables are u, y, along with their derivatives, up to the Nth order. First, we fix an order of $(N+1)+(M(N+1))$ variables as $y^{(N)}, \ldots, y, u^{(N)}, \ldots, u$. Let us consider a monomial, M_1, of these variables, and denote the order of each variable as $\alpha_1, \ldots, \alpha_{(N+1)+(M(N+1))}$. In the same way, we consider another monomial, M_2, and denote the order as $\beta_1, \ldots, \beta_{(N+1)+(M(N+1))}$. We then evaluate $\beta_i - \alpha_i (i = 1, \ldots, (N+1) + (M(N+1)))$ in order and find the non-zero elements with the smallest i. The smallest i is denoted as s. We define $M_1 < M_2$ if $\beta_s - \alpha_s$ is positive. We call this the input-output order, which is precisely defined in Definition 6.

Example 1 (the input-output order) *Let us consider a polynomial, $h := y^{(2)} + y - (1 - y^2)y^{(1)}$ where the order of variables is $y^{(2)}, y^{(1)}, y$, as discussed in Section 3. The monomials in h are $y^{(2)}, y, y^{(1)}$, and $y^{(1)}y^2$. Given the input-output order, they are ordered as $y^{(2)} > y^{(1)}y^2 > y^{(1)} > y$.*

Hereafter, the monomial with the highest order in polynomial p with respect to order $<$ is denoted as $\mathrm{LM}_<(p)$ and the coefficient of $\mathrm{LM}_<(p)$ is denoted as $\mathrm{LC}_<(p)$. For instance, consider h appears in Example 1. $\mathrm{LM}_<(h) = y^{(2)}$ and $\mathrm{LC}_<(h) = 1$ hold with respect to the input-output order. With these notations, a theorem on the replaceability is provided. Let us consider the correspondence between $h = 0$ and the input-output equation of a target system for the practical use of the theorem.

Theorem 2 *Let us consider a physical computing system described by* (1) *and assume it is generic. Suppose that $\bar{y}(t)$ satisfying Assumption 1 and 2 are observed over $[0, T]$; and suppose that there exists $\bar{y}(t) \in \mathcal{F}_y$ satisfying a differential equation $h(\bar{y}^{(N)}, \ldots, \bar{y}; \tilde{\theta}) = 0$ where the initial value of $\bar{y}$ is denoted as $\bar{y}_0 := (\bar{y}(0), \ldots, \bar{y}^{(N)}(0)) \in \mathbb{R}^{N+1}$. Suppose that for* (1) *given $\bar{y}$, there exists a non-empty set of consistent initial values, $X \subset \mathbb{R}^N$. We fix a monomial order of $\mathbb{C}(\theta, \tilde{\theta})[y^{(N)}, \ldots, y, u^{(N)}, \ldots, u]$ as the input-output order $<$. $h(y^{(N)}, \ldots, y; \tilde{\theta})$ is divided by F, defined as* (3)*, and represented as*

$$q(y^{(N)}, \ldots, y, u^{(N)}, \ldots, u; \theta, \tilde{\theta})F + R(y^{(N)}, \ldots, y, u^{(N)}, \ldots, u; \theta, \tilde{\theta}), \tag{7}$$

where we assume that $\mathrm{LM}_<(h) \geq \mathrm{LM}_<(F)$, $\mathrm{LC}_<(F)$ is non-zero, and the coefficient of h with respect to $\mathrm{LM}_<(F)$ is non-zero or a non-zero function over $[0, T]$. Then, $\bar{y}(t)$ can be generated by (1) *given $(\theta, u, x(0)) \in \Theta \times \mathcal{F}_u \times X$ if the following holds:*

$$R(\bar{y}^{(N)}(t), \ldots, \bar{y}(t), u^{(N)}(t), \ldots, u(t); \theta, \tilde{\theta}) = 0. \tag{8}$$

The basic assumptions are the same as those for Theorem 1 together with Assumption 2. Additional assumptions are related to the order of the monomials. In general, when the polynomials are divided, the polynomials to be divided have higher-order monomials than those used for the division. Hence, to consistently perform the division, it must hold that $\mathrm{LM}_<(h) \geq \mathrm{LM}_<(F)$, which is one of the additional assumptions. In addition, we assume that $\mathrm{LC}_<(F)$ and $\mathrm{LC}_<(h)$ are non-zero or non-zero functions over $[0, T]$, which are generic ones. (8) is a differential polynomial equation up to the Nth order, whose coefficients are functions of $\theta, \tilde{\theta}$ and the variables are u. In the same manner as for (4), design guidelines for a physical computing system can be derived from (8). However, in this case, they are related to the replaceability of (1) with a target system generating $\bar{y}$.

3 Experiments: Exploitation of the replaceability and subsequent design

We apply Theorem 2 and investigate specific design guidelines for a physical computing system to be utilized to replace a target system that generates parameterized time-series. Let us consider a physical computing system modeled by a FitzHugh-Nagumo equation with parameters $(\theta_1, \ldots, \theta_4)^\top \in \mathbb{R}^4$,

$$\dot{x}_1 = \theta_1 \left(x_1 - (1/3)\, x_1^3 + x_2 + \theta_2\right), \quad \dot{x}_2 = -\left(1/\theta_1\right)\left(x_1 - \theta_3 + \theta_4 x_2\right) \tag{9}$$

with readout $y = x_1$. We denote this system as Σ_1 and regard this as the system to be designed. The input-output equation of Σ_1 is as follows:

$$\theta_4 y^3 + 3\theta_1 \dot{y} y^2 + 3\left(1 - \theta_4\right) y + 3\ddot{y} - 3\dot{y}\left(\theta_1 - (\theta_4/\theta_1)\right) - 3\left(\theta_3 - \theta_2\theta_4\right) = 0. \tag{10}$$

We denote the left side of (10) as F_1. Then, we consider a target system that consists of van der Pol equation [25], $\dot{\tilde{x}}_1 = \tilde{x}_2, \quad \dot{\tilde{x}}_2 = -\tilde{x}_1 + (1 - \tilde{x}_1^2)\tilde{x}_2$ and observation equation $\tilde{y} = x_1$. We denote

this system as Σ_2. The input-output equation of Σ_2 is $\dddot{\tilde{y}} + \ddot{\tilde{y}} - (1 - \tilde{y}^2)\dot{\tilde{y}} = 0$. By dividing the left-side of this by F_1 following the procedures explained in Example 6 of Appendix H, we obtain $(1-\theta_1)\dot{y}y^2 - (\theta_4/3)\,y^3 + \left(\left(\theta_1^2 - \theta_1 - \theta_4\right)/\theta_1\right)\dot{y} + \theta_4 y + (\theta_2\theta_4 + \theta_3)$ as the remainder, denoted by R_1. Based on R_1 and Theorem 2, it is shown that Σ_1 generates a time-series generated by Σ_2 if, essentially, it holds that $(\theta_1, \theta_3, \theta_4)^\top = (1, 0, 0)^\top$ given consistent initial values (see Appendix J). Note that the value of θ_2 does not affect the replaceability. Given these constraints on the parameters and initial values, Σ_1 is shown to be replaceable with Σ_2. To numerically confirm this, we conduct the following comparative experiments, in which the value of θ_2 is randomly taken from $[0, 1]$.

Replaceability with respect to a single target time-series We fix initial states of Σ_2 as $(\tilde{x}_1(0), \tilde{x}_2(0))^\top = (2, 0)^\top$, and thus, consider the replaceability with respect to a target time-series. In Fig. 2(a), Σ_1 is designed such that it satisfies the constraints on the parameters and initial states. As it follows the proposed design guidelines, the output of Σ_1 with Σ_2 is well-fitted, and the average of the normalized MSE is 0.0056. In contrast, Figs. 2 (b) and (c) show that the results with Σ_1 disobey some of these guidelines without the constraints on the (b) parameters and (c) initial states. As shown in Figs. 2 (b) and (c), when neither $(\theta_1, \theta_3, \theta_4)^\top = (1, 0, 0)^\top$ nor $x(0) = (2, 0.167)^\top$ is unsatisfied, the outputs of Σ_1 and Σ_2 greatly differ. The average values of the normalized MSE were 1.0395 and 0.0983. In addition, we confirmed that a stationary state of Σ_2, where $t \in [100, 200]$, which forms an attractor, is also appropriately generated (Fig. 2(d).) See Section K for the details. Note that both the stationary and non-stationary tasks are accomplished using our algebraic approach.

Replaceability with respect to initial values of a target system For simplicity, we consider $(\theta_1, \theta_2, \theta_3, \theta_4)^\top = (1, 0.5, 0, 0)^\top$, which are the same constraints used in Fig. 2(a). Here, we randomly choose $(\tilde{x}_1(0), \tilde{x}_2(0))^\top$ from $[-2, 2]^2$ 10 times and design the initial states of Σ_1 to be consistent so that it they fit each target time-series. The results are summarized in Fig. 3 (a). As observed, when the initial values of Σ_1 are consistent, Σ_1 approximats the output of Σ_2 well (the normalized MSE is 3.8173×10^{-6}). This indicates that the Σ_1 satisfying the proposed design guidelines can be replaced with Σ_2 for various initial states from $[-2, 2]^2$.

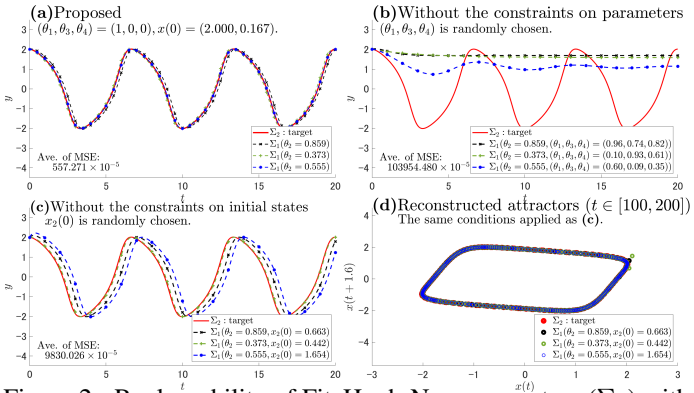

Figure 2: Replaceability of FitzHugh-Nagumo system (Σ_1) with respect to the target system (Σ_2) given constraints on the **(a)** parameters and initial states, **(b)** initial states, and **(c)** parameters. **(d)** The reconstructed attractors under the same conditions as **(c)**.

Replaceability with respect to parameters of a target system We consider a modified target system, denoted by Σ_2'. Differential equations of Σ_2' are $\dot{\tilde{x}}_1 = \tilde{x}_2$, $\dot{\tilde{x}}_2 = -\tilde{x}_1 + \tilde{\theta}(1 - \tilde{x}_1^2)\tilde{x}_2$ where $\tilde{\theta} \in \mathbb{R}$ denotes its parameter and $\tilde{x}_1$ is observed. The input-output equation of Σ_2' is $\ddot{y} + y - \tilde{\theta}(1 - y^2)\dot{y}$. The remainder of this by F_1 with respect to the input-output order, is $(-\theta_1 + \tilde{\theta})\dot{y}y^2 + ((\theta_1{}^2 - \theta_1\tilde{\theta} - \theta_4)/\theta_1)\dot{y} - (\theta_4/3)y^3 + \theta_4 y + (\theta_2\theta_4 + \theta_3)$. Here, we let $\tilde{\theta}$ take $1, 3$, and -3 and set $(\theta_1, \theta_2, \theta_3, \theta_4)^\top$ as $(\tilde{\theta}, 0.5, 0, 0)^\top$. The same initial values of the target system were applied as previous experiments and $(x_1(0), x_2(0))^\top$ are designed consistently. As observed in Fig. 3(b), if Σ_1 satisfies $(\theta_1, \theta_3, \theta_4)^\top = (\tilde{\theta}, 0, 0)^\top$, it can be replaced with and approximate Σ_2' (the average of the normalized MSE was 1.7765×10^{-6}).

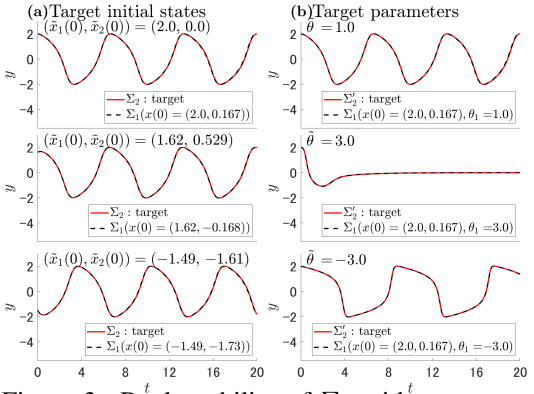

Figure 3: Replaceability of Σ_1 with respect to $(\tilde{x}_1(0), \tilde{x}_2(0))^\top$ or $\tilde{\theta}$ of Σ_2. The outputs of Σ_1 are shown in dashed lines and the target time-series are shown in solid red lines.

4 Conclusion

In this paper, we proposed an algebraic design framework for physical computing systems that can offer specific design guidelines. Two main theorems were provided and their validity was shown through experiments. The proposed framework has the following limitations. The Gröbner basis computation is often computationally expensive for large physical computing systems. Incorporating efficient methods for deriving input-output equations such as [5], which is recently proposed, into our framework may be a promising method to avoid such a scale limitation. Considering the system and observation noise caused by physical systems, the relaxation of Assumption 1 should be investigated. Combining algebraic techniques with the integration of the model instead of its differentiation may be one way to accomplish this. See Appendix L for further discussions.

Acknowledgments and Disclosure of Funding

This work is supported by JST, CREST Grant JPMJCR1914, PRESTO Grants JPMJPR15E7 and JPMJPR16EC, ACT-X Grant JPMJAX22A7, Japan. M. K is supported by JSPS KAKENHI Grand Number 22K21278. K.N. is supported by JST, CREST Grant JPMJCR2014.

References

[1] N. Akashi, Y. Kuniyoshi, S. Tsunegi, T. Taniguchi, M. Nishida, R. Sakurai, Y. Wakao, K. Kawashima, and K. Nakajima. A coupled spintronics neuromorphic approach for high-performance reservoir computing. *Advanced Intelligent Systems*, 4(10):2200123, 2022.

[2] S. Boyd and L. Chua. Fading memory and the problem of approximating nonlinear operators with volterra series. *IEEE Transactions on Circuits and Systems*, 32(11):1150–1161, 1985.

[3] D. A. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, Cham, 2015.

[4] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 4-1-1 — A computer algebra system for polynomial computations. `http://www.singular.uni-kl.de`, 2018.

[5] R. Dong, C. Goodbrake, H. A. Harrington, and G. Pogudin. Differential elimination for dynamical models via projections with applications to structural identifiability. 2023.

[6] K. Forsman. *Constructive Commutative Algebra in Nonlinear Control Theory*. PhD thesis, Linköping UniversityLinköping University, Automatic Control, The Institute of Technology, 1991.

[7] H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass. Towards a theoretical foundation for morphological computation with compliant bodies. *Biological Cybernetics*, 105(5):355–370, 2011.

[8] H. Jaeger. The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148, 01 2001.

[9] H. Jaeger. Towards a generalized theory comprising digital, neuromorphic and unconventional computing. *Neuromorphic Computing and Engineering*, 1(1):012002, jul 2021.

[10] D. Kapur. Comprehensive gröbner basis theory for a parametric polynomial ideal and the associated completion algorithm. *Journal of Systems Science and Complexity*, 30:196–233, 02 2017.

[11] C. Kaspar, B. Ravoo, W. Wiel, S. Wegner, and W. Pernice. The rise of intelligent matter. *Nature*, 594:345–355, 06 2021.

[12] D. Kobayashi, Y. Kakehashi, K. Hirose, S. Onoda, T. Makino, T. Ohshima, S. Ikeda, M. Yamanouchi, H. Sato, E. C. I. Enobio, T. Endoh, and H. Ohno. Influence of heavy ion irradiation on perpendicular-anisotropy cofeb-mgo magnetic tunnel junctions. *IEEE Transactions on Nuclear Science*, 61:1710–1716, 2013.

[13] M. Komatsu, T. Yaguchi, and K. Nakajima. Algebraic approach towards the exploitation of "softness": the input–output equation for morphological computation. *The International Journal of Robotics Research*, 40(1):99–118, 2021.

[14] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2):130 – 141, 1963.

[15] W. Maass, P. Joshi, and E. D. Sontag. Computational aspects of feedback in neural circuits. *PLoS computational biology*, 3(1):e165, Jan 2007.

[16] G. Manjunath and H. Jaeger. Echo State Property Linked to an Input: Exploring a Fundamental Characteristic of Recurrent Neural Networks. *Neural Computation*, 25(3):671–696, 03 2013.

[17] M. Mannattil, A. Pandey, M. K. Verma, and S. Chakraborty. On the applicability of low-dimensional models for convective flow. *The European Physical Journal B*, 90(259), 2017. https://doi.org/10.1140/epjb/e2017-80391-1.

[18] K. Nakajima. Physical reservoir computing—an introductory perspective. *Japanese Journal of Applied Physics*, 59(6):060501, may 2020.

[19] K. Nakajima and I. Fischer. *Reservoir Computing: Theory, Physical Implementations, and Applications*. Springer-Singapore, 2021.

[20] M. Nakajima, K. Inoue, K. Tanaka, Y. Kuniyoshi, T. Hashimoto, and K. Nakajima. Physical deep learning with biologically inspired training method: gradient-free approach for physical hardware. *Nature communications*, 13(1):7847, December 2022.

[21] L. H. Nguyen and K.-S. Hong. Synchronization of coupled chaotic fitzhugh–nagumo neurons via lyapunov functions. *Mathematics and Computers in Simulation*, 82(4):590–603, 2011.

[22] O. Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976.

[23] M. H. Stone. The generalized weierstrass approximation theorem. *Mathematics Magazine*, 21(4):167–184, 1948.

[24] L. N. Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM-Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2019.

[25] B. van der Pol. Relaxation-oscillations. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 7:978–992, 1926.

[26] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon. Deep physical neural networks trained with backpropagation. *Nature*, 601(7894):549—555, January 2022.

[27] H. Yasuda, P. R. Buskohl, A. Gillman, T. D. Murphey, S. Stepney, R. A. Vaia, and J. R. Raney. Mechanical computing. *Nature*, 598(7879):39—48, October 2021.

Appendix

A Algebraic preliminary

Here, we introduce some algebraic terminology related to the study. For basic terminology concerning the field, ring, polynomial ring, and order, please refer to [3, 13]. Throughout this study, $K[x]$ denotes a polynomial ring whose variables are $x = (x_1, \ldots, x_s)$ over a field K.

Definition 1 (Ideal on a polynomial ring) *A non-zero subset of $K[x]$ satisfies*

$$\text{if } p \in I, q \in I, \text{ then } p + q \in I, \tag{11}$$
$$\text{if } p \in I, q \in K[x], \text{ then } qp \in I, \tag{12}$$

where $K[x]$ is a polynomial ring; p, q are polynomials.

Definition 2 (Generator) *For an ideal I, suppose that there exists a non-zero subset $\{p_\lambda \mid \lambda \in \Lambda\}$ of ring $K[x]$ that satisfies*

$$I = \left\{ \sum_{\lambda \in \Lambda} q_\lambda p_\lambda \,\middle|\, \text{for all } q_\lambda \in K[x] \right\}. \tag{13}$$

Then, $\{p_\lambda \mid \lambda \in \Lambda\}$ is defined as a generator of I.

Notably, the set on the right side of (13) is shown to satisfy the definition of the ideal[3]. Representing the ideal I, whose generator is $\{p_\lambda \mid \lambda \in \Lambda\}$, is commonly expressed as follows:

$$I = \langle \{p_\lambda \mid \lambda \in \Lambda\} \rangle. \tag{14}$$

For example, the ideal generated by (1), and their derivatives up to the Nth order is denoted as (23). The underlying algorithm for deriving the input-output equation (3) and remainder such as r appearing in (7) is common - polynomial division. We introduce some concepts related to dividing a multivariate polynomial by a single multivariate polynomial for conciseness. For a general case, i.e., dividing a multivariate polynomial by a set of multivariate polynomials, please refer to [3]. As stated in Section 2.4, we consider the monomial order to handle polynomials, which is a special type of the total order.

Definition 3 (Total order) *$\leq$ is a total order on a set Σ if the followings holds for all $a, b, c \in \Sigma$:*

$$a \leq a, \tag{15}$$
$$a \leq b \,\text{or}\, b \leq a, \tag{16}$$
$$\text{if } a \leq b \,\text{and}\, b \leq a, \text{then } a = b, \tag{17}$$
$$\text{if } a \leq b \,\text{and}\, b \leq c, \text{then } a \leq c. \tag{18}$$

Conventionally, we denote $a < b$ if both $a \leq b$ and $a \neq b$ hold. Among a set Σ equipped with a total order, we can compare every pair of elements. This is important in ordering monomials, as explained in Section 2.4. By adding some properties to the total order, the monomial orderings are defined as follows.

Definition 4 (Monomial order) *Monomial order $<$ is the total order on a set of monomials M_s of a polynomial ring such that*

$$1 < u, \text{for all } u \in M_s, u \neq 1, \tag{19}$$
$$\text{if } u \leq v\, u, v \in M_s \text{ then } uw \leq vw, \text{for all } w \in M_s. \tag{20}$$

The input-output order defined in Definition 6, a monomial order called the lexcographic order.

Definition 5 (Lexcographic order) *Let us consider a ring $K[x]$ and monomial order $<$ on the ring such that for arbitrary $u = x_1^{\alpha_1} x_2^{\alpha_2}, \ldots, x_s^{\alpha_s}, v = x_1^{\beta_1} x_2^{\beta_2}, \ldots,$ and $x_s^{\beta_s} \in K[x]$, $u < v$ holds if the left-most element among the non-zero elements of $(\beta_1 - \alpha_1, \beta_2 - \alpha_2, \ldots, \beta_s - \alpha_s)$ is positive. We define $<$ as the lexcographic order.*

The lexicographic order can be used to eliminate variables from ideals [3]; hence, it is used in this study. We eliminate state variables and their derivatives from (23) to derive (3). The type of lexcographic order shown in the following examples is what we used to compute the Gröbner basis and derive input-output equations.

Example 2 *For a physical computing system modeled as* (1)*, we consider the polynomial ring*

$$\mathbb{C}(\theta)[x^{(N)}, \dots, x, y^{(N)}, \dots, y, u^{(N)}, \dots, u] \tag{21}$$

as in Proposition 1. To eliminate $x^{(N)}, \dots, x$ from the ideal (23) *corresponding to the model, we fix the monomial order of the ring to the lexcographic order such that $x^{(N)} > \dots > x > y^{(N)} > \dots > y > u^{(N)} > \dots > u$. That is, monomials in the ring are considered as $\left(x^{(N)}\right)^{\alpha_1}\left(x^{(N-1)}\right)^{\alpha_2}\cdots\left(u^{(1)}\right)^{\alpha_{s-1}}(u)^{\alpha_s}$, where the derivatives (e.g., $x^{(N)}$) are considered distinct from the ordinary variables (e.g., x).*

Definition 6 (input-output order) *Let us denote the ring $\mathbb{C}(\theta)[y^{(N)}, \dots, y, u^{(N)}, \dots, u]$ as $\mathbb{C}(\theta)[z]$, in which each element of $z = (z_1, z_2, \dots, z_s)$ corresponds to $y^{(N)}, \dots, y, u^{(N)}, \dots, u$ in order. Let us consider a monomial order $<$ such that for arbitrary $u = z_1^{\alpha_1}z_2^{\alpha_2}, \dots, z_s^{\alpha_s}, v = z_1^{\beta_1}z_2^{\beta_2}, \dots, z_s^{\beta_s} \in \mathbb{C}(\theta)[z]$, $u < v$ holds if the left-most element among the non-zero elements of $(\beta_1 - \alpha_1, \beta_2 - \alpha_2, \dots, \beta_s - \alpha_s)$ is positive. We define $<$ as an input-output order.*

Example 3 *In Theorem 2, we consider the polynomial ring $\mathbb{C}(\theta)[y^{(N)}, \dots, y, u^{(N)}, \dots, u]$, which is the subset of the ring considered in Example 2. The input-output order defined in Definition 6 is the order induced by the lexcographic order explained in Example 2. That is, we fix the monomial order of the ring to the lexcographic order such that $y^{(N)} > \dots > y > u^{(N)} > \dots > u$. This corresponds to considering monomials in the ring as $\left(y^{(N)}\right)^{\beta_1}\left(y^{(N-1)}\right)^{\beta_2}\cdots\left(u^{(1)}\right)^{\beta_{s-1}}(u)^{\beta_s}$.*

As stated in Section 2.3, the input-output equations are derived by computing the Gröbner basis, whose underlying algorithm is polynomial division. Before explaining this, we introduce related terminology.

Definition 7 *The largest monomial in the set of monomials, whose coefficients are non-zero, appearing in a polynomial $p \in K[x]$ with respect to a given monomial order $<$ is defined as the leading monomial of p. We denote this by $\mathrm{LM}_<(p)$.*

Definition 8 *The coefficient of the leading monomial of $p \in K[x]$ with respect to a given monomial order $<$ is defined as the leading coefficient of p. We denote this as $\mathrm{LC}_<(p)$.*

Definition 9 *Let us consider a ring $K[x]$ with a given a monomial order $<$. Suppose we have non-zero polynomials p and h in $K[x]$. We represent h as*

$$h = qp + R, \tag{22}$$

where the followings holds:

- *if $R \neq 0$, then R is not represented by $R = q'p$, where $q \in K[x]$;*
- *if $q \neq 0$, then $\mathrm{LM}_<(h) \geq \mathrm{LM}_<(qp)$.*

R is defined as the remainder of h divided by p with respect to $<$ over $K[x]$.

The existence of q and R in Definition 9 is shown [3, p.64]. Although this is the case of a polynomial being divided by a single polynomial, it can be extended to polynomial division using a set of polynomials[3]. Because $\mathrm{LM}_{(h)}\mathrm{LM}_{(qp)}$ holds if $q \neq 0$, $\mathrm{LM}_<(h) < \mathrm{LM}_<(p)$ implies $q = 0$, which leads to an identical equation $h = r$. That is, if the leading monomial of a polynomial to be used to divide is greater than that of the polynomial to be divided, the remainder is trivial. Therefore, we make an assumption to avoid this (See the assumption $\mathrm{LM}_<(h) \geq \mathrm{LM}_<(p)$ in Theorem 2.) In general, because the division algorithm is implemented on computer algebra software, r can be specifically computed using a given h, p, and monomial order. See [3] for the details and algorithms on the division of polynomials of multiple variables.

B Proposition 1 and its proof

Proposition 1 (Theorem shown in [6]) *For a model* (1) *with* $f, g \in \mathbb{C}(\theta)[x, u]$*, suppose that the ideal* $I^{(N)}$ *is defined as*

$$\begin{aligned}&\langle \frac{\mathrm{d}x}{\mathrm{d}t} - f, \ldots, \frac{\mathrm{d}^N x}{\mathrm{d}t^N} - f^{(N-1)}, \ldots y - g, \ldots, y^{(N)} - g^{(N)} \rangle \\ &\subset \mathbb{C}(\theta)[x^{(N)}, \ldots, x, y^{(N)}, \ldots, y, u^{(N)}, \ldots, u]. \end{aligned} \tag{23}$$

Then, (2) *is not zero ideal.*

The following Lemma shown in [6] is used to prove Proposition 1.

Lemma 1 (Theorem 1.6 of [6]) *Let* K *be the field. Polynomials* $p_1, \ldots, p_S \in K[x_1, \ldots, x_s]$ *are algebraically dependent over* K *if* $S > s$*, that is, there exists a non-zero polynomial* $P \in k[z_0, \ldots, z_S]$ *such that* $P(p_1, \ldots, p_S) = 0$*.*

Proof of Proposition 1 Regarding $y^{(N)}, \ldots, y$ satisfying (1) and their higher order derivatives up to the Nth order, the following holds:

$$y^{(N)}, \ldots, y \in \mathbb{C}(u^{(N)}, \ldots, u, \theta)[x_1, \ldots, x_N], \tag{24}$$

because the higher order derivatives of $x_1, \ldots, x_N$ of (1) can be represented by those with the lower order derivatives. According to Lemma 1, $y^{(i)} (i = 0, \ldots, N)$ being algebraically dependent over $\mathbb{C}(u^{(N)}, \ldots, u, \theta)$ holds. Thus, a non-zero polynomial $P \in \mathbb{C}(u^{(N)}, \ldots, u, \theta)[z_0, \ldots, z_N]$ exists such that $P(y^{(N)}, \ldots, y) = 0$. By clearing the denominators of the coefficients of $P(y^{(N)}, \ldots, y)$, we obtain a polynomial in $I^{(N)} \cap \mathbb{C}(\theta)[y^{(N)}, \ldots, y, u^{(N)}, \ldots, u]$. This concludes the proof of Proposition 1.

C Gröbner basis

Definition 10 (Gröbner basis) *For an ideal* I *on a polynomial ring* $K[x]$ *with a given monomial order, if its generator* $G = \{g_1, \ldots, g_s\}$ *satisfies*

$$p \in I \Leftrightarrow p \text{ is divisible by } G, \tag{25}$$

then G *is called the Gröbner basis of* I*.*

For an arbitrary non-zero ideal on polynomial rings with a fixed monomial order, there exists a Gröbner basis [3]. According to (25), any polynomial in an ideal can be represented as a linear combination of its Gröbner basis. Furthermore, according to (25), the Gröbner basis is known to solve a membership problem, i.e., discriminate whether a given polynomial is in the ideal [3], and thus characterize the ideal. In this study, we applied this property in Section 2.4.

Although the Gröbner basis of an ideal with respect to a fixed monomial order is not unique, the following is unique.

Definition 11 (Reduced Gröbner basis) *For an ideal* I *on a polynomial ring* $K[x]$ *with a given monomial order, if* $G = \{g_1, \ldots, g_s\}$ *is the Gröbner basis of* I *and satisfies the followings:*

- $\mathrm{LC}(g_i)$ *is 1 for all* $i = 1, \ldots, s$*; and*
- *if* $i \neq j$*, any monomial in the set of monomials, whose coefficients are non-zero, appear in* g_i *is not divisible by* $\mathrm{LM}(g_i)$*,*

then G *is the reduced Gröbner basis of* I*.*

To conclude this section, we provide a proposition known as the elimination theorem [3]. This is critical for obtaining the input-output equations.

Proposition 2 (Elimination Theorem) *Suppose that the lexicographic order is used as the monomial order of a ring of polynomials* $K[x_1, \ldots, x_n]$ *such that* $x_1 > x_2 > \ldots > x_n$*. If* G *is a reduced Gröbner basis of an ideal* $I \subset K[x_1, \ldots, x_n]$*, then* $G \cap K[x_1, \ldots, x_j]$ *is a Gröbner basis of* $I \cap K[x_1, \ldots, x_j]$*.*

This ensures that variables can be eliminated from an ideal using the reduced Gröbner basis with respect to the lexicographic order. This is used to obtain the input-output equations, which is detailed in Appendix D.

D Derivation of the input-output equation

The Gröbner basis can be computed using the computer algebra software. We provide commands required to compute the Gröbner basis using Singular [4], which is free. We first consider Example 4 of Appendix H, in which the derivation of the variables does not appear.

Next, we provide Singular commands required to compute the input-output equation of (1). In the following examples and commands, θ in (1) is replaced with a for brevity. In this case, the variables to be eliminated are state variables and their derivatives. As stated in Theorem 1, to eliminate these, considering (1), and their higher order derivatives up to the Nth order suffices, each of which becomes an element of the generators of (23). Derivatives are considered distinct variables in the computations; hence, the algebraic computations explained are applicable. See Example 4 of Appendix H for a specific example.

E Proof of Theorem 1

Definition 12 (Consistent initial values) *We define the consistent initial values of* (1) *and their derivatives up to Nth order as $\left(x(0), \dots, x^{(N)}(0)\right) \in \mathbb{R}^{N\times(N+1)}$ such that*

$$\begin{aligned} &\dot{x}(0) - f(x(0), u(0); \theta), \dots, x^{(N)}(0) - f^{(N-1)}(x(0), u(0); \theta), \\ &\bar{y}(0) - g(x(0), u(0); \theta), \dots, \bar{y}^{(N)}(0) - g^{(N)}(x(0), u(0); \theta) \end{aligned} \tag{26}$$

hold, where u is in a set of sufficiently smooth functions $\mathcal{F}_u$, which are functions from $\mathbb{R} \to \mathbb{R}^M$, and $(\bar{y}(0), \dots, \bar{y}^{(N)}(0)) \in \mathbb{R}^{(N+1)}$ is an initial value of a given time-series $\bar{y} \in \mathcal{F}_y$, which is a set of sufficiently smooth functions $\mathbb{R} \to \mathbb{R}$.

In Theorem 1, the set of consistent initial values are denoted as X, where only a subset of variables $x(0) \in \mathbb{R}^N$ are considered, unlike in Definition 12. Strictly, the consistency of initial variables concerns $\left(x(0), \dots, x^{(N)}(0)\right)$, as in Definition 12.

Lemma 2 *Suppose that $u \in \mathcal{F}_u$ and that the system of differential equations in* (1) *admits a unique solution given initial state vector $x(0)$. Then, consider*

$$\begin{pmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(N+1)} \end{pmatrix} = \begin{pmatrix} f(x, u; \theta) \\ \frac{\mathrm{d}}{\mathrm{d}t} f(x, u; \theta) \\ \vdots \\ \frac{\mathrm{d}^{(N)}}{\mathrm{d}t^{(N)}} f(x, u; \theta) \end{pmatrix}, \tag{27}$$

which is the system of differential equations that consists of the system of differential equations in (1) *and its higher order derivatives up to the Nth order. Then,* (27) *admits a unique solution given initial values such that*

$$\begin{pmatrix} x^{(1)}(0) \\ \vdots \\ x^{(N)}(0) \end{pmatrix} = \begin{pmatrix} f(x(0), u(0); \theta) \\ \vdots \\ \frac{\mathrm{d}^{(N-1)}}{\mathrm{d}t^{(N-1)}} f(x(0), u(0); \theta) \end{pmatrix}. \tag{28}$$

Furthermore, the output satisfying the observation equation in (1) *and their derivatives up to the Nth order given* (28) *are also unique.*

Proof of Lemma 2 Let us substitute the unique solution of the system of differential equations in (1) into their right side, $f(x, u; \theta)$. We then obtain a unique $x^{(1)}(t)$, which is well defined because $f(x, u; \theta)$ is a polynomial vector with respect to x and u. Then, $x^{(1)}(0) = f(x(0), u(0); \theta)$ holds. By substituting $x(t), x^{(1)}(t), u(t), u^{(1)}(t)$, and θ into the right side of the derivative of the system of differential equations, $\frac{\mathrm{d}}{\mathrm{d}t} f(x, u; \theta)$, we obtain a unique $x^{(2)}(t)$. $x^{(2)}(t)$ that is well defined because

$\frac{\mathrm{d}}{\mathrm{d}t}f(x,u;\theta)$ is a polynomial vector with respect to $x, x^{(1)}, u$ and $u^{(1)}$. $x^{(2)}(0) = f^{(1)}(x(0),u(0);\theta)$ holds as well as $x^{(1)}(0)$. By repeating these procedures, we obtain a unique $x(t), x^{(1)}(t), \ldots, x^{(N)}(t)$, that is, the solution of (27) given (28). Furthermore, these $x(t), x^{(1)}(t), \ldots, x^{(N)}(t)$ are shown to be unique by substituting them into the observation equation in (1) and their derivatives up to the Nth order.

Definition 13 (Generic physical computing system) *Let us consider* (1)*, whose input-output equation is* (3)*, given a time-series $\bar{y} \in \mathcal{F}_y$. Considering* (3)*, as a differential equation with respect to y, given initial values $(\bar{y}(0), \ldots, \bar{y}^{(N)}(0))$, and admitting a unique solution given arbitrary $u \in \mathcal{F}_u$,* (1) *are defined as generic.*

Definition 14 (Generic parameter vectors) *Let us consider* (1)*, the input-output equation of which is* (3)*. A specific parameter vector $\theta_* \in \mathbb{R}^n$ is defined as generic if the input-output equation of* (1) *with $\theta = \theta_*$ is equivalent to* (3) *with θ_* substituted for θ.*

In general, generic parameters can be configured using a comprehensive Gröbner system [10]. Throughout this work, we consider generic parameter vectors and a set of generic parameter vectors $\Theta \subset \mathbb{R}^n$.

Proof of Theorem 1 Let us take an arbitrary $(\theta, u, x(0)) \in \Theta \times \mathcal{F}_u \times X$. According to Lemma 2, a unique y and their derivatives up to the Nth order exist satisfying (1) given $(\theta, u, x(0))$ and their derivatives up to the Nth order. We denote these as $\hat{y}(t;\theta,u,x(0)), \ldots, \hat{y}^{(N)}(t;\theta,u,x(0))$, introducing the notation shown in Section 2.2.

First, we show that for any $(\theta, u, x(0)) \in \Theta \times \mathcal{F}_u \times X$, $\hat{y}(t;\theta,u,x(0)) = \bar{y}(t)$ holds over $[0,T]$ if

$$F\left(\bar{y}^{(N)}(t), \ldots, \bar{y}(t), u^{(N)}(t), \ldots, u(t); \theta\right) = 0. \tag{29}$$

Because $(x(0), \ldots, x^{(N)}(0))$ is consistent based on the assumption, it holds that

$$(\hat{y}(0;\theta,u,x(0)), \ldots, \hat{y}^{(N)}(0;\theta,u,x(0))) = (\bar{y}(0), \ldots, \bar{y}^{(N)}(0)). \tag{30}$$

Therefore, $\hat{y}(t;\theta,u,x(0))$ should satisfy (3) considering that F is in (23) and $\theta \in \Theta$. Hence, considering that both $\hat{y}(t;\theta,u,x(0))$ and $\bar{y}(t)$ satisfy (3) given $(\bar{y}(0), \ldots, \bar{y}^{(N)}(0))$, and (1) is consistent, they must coincide over $[0,T]$. Thus, we finish by showing sufficiency.

Next, we demonstrate that for any $(\theta, u, x(0)) \in \Theta \times \mathcal{F}_u \times X$, (4) holds if $\bar{y}(t)$ can be generated by (1). Let us again take an arbitrary $(\theta, u, x(0)) \in \Theta \times \mathcal{F}_u \times X$. Because $\bar{y}(t) \in \mathcal{F}_y$ can be generated by (1), this is equivalent to $\hat{y}(t;\theta,u,x(0))$ up to the Nth order, where $\hat{y}(t;\theta,u,x(0))$ is a unique solution of (1), as shown in Lemma 2. Furthermore, $\left(\hat{y}(t;\theta,u,x(0)), \ldots, \hat{y}^{(N)}(t;\theta,u,x(0))\right)$ satisfies (3) because F is in (23) and $\theta \in \Theta$. Therefore, $\left(\bar{y}, \ldots, \bar{y}^{(N)}\right)$ satisfies F, that is, (4) holds. This concludes the proof.

F Proof of Theorem 2

Proof of Theorem 2 Because $\mathrm{LM}_<(h) \geq \mathrm{LM}_<(F)$ holds, q in (7) is represented by $q'/\mathrm{LC}(F)$, that is, $q = q'/\mathrm{LC}(F)$. According to the assumptions, $\mathrm{LC}_<(F)$ is non-zero and q' is non-zero or a non-zero function; thus, q is non-zero over $t \in [0,T]$. By substituting $(\bar{y}, \ldots, \bar{y}^{(N)})$ into $(y, \ldots, y^{(N)})$ of (7), the left side of (7) and r take zeros over $t \in [0,T]$ because $\bar{y}$ is a solution of $h(y^{(N)}, \ldots, y; \tilde{\theta}) = 0$ and (8). Thus, it must hold that

$$F\left(\bar{y}^{(N)}(t), \ldots, \bar{y}(t), u^{(N)}(t), \ldots, u(t); \theta\right) = 0 \tag{31}$$

over $t \in [0,T]$. This is equivalent to (4). Hence, by connecting this to the sufficiency of Theorem 1, we finish the proof.

G Division by the input-output equation

A multivariate polynomial can be divided by another multivariate polynomial using computer algebra software. Singular commands for such divisions are shown in Example 6 of Appendix H.

H Examples

Example 4 *Suppose we have simultaneous equations as follows:*

$$x^2 + y^2 + z^2 = 4, \quad x^2 + 2y^2 = 5, \quad xz = 1. \tag{32}$$

Let us eliminate x and y to solve (32) *for z based on the computation of the Gröbner basis of the ideal generated by* (32)*. To do this, we consider the following procedures:*

1. *We consider ring $\mathbb{C}[x, y, z]$. We fix the monomial ordering over this as the lexcographic ordering $<$ such that $x < y < z$.*
2. *We define the ideal I generated by* (32)*, $\langle x^2 + y^2 + z^2 - 4, \quad x^2 + 2y^2 - 5, \quad xz - 1\rangle$.*
3. *Compute the Gröbner basis of I with respect to $<$.*

An example of Singular commands for conducting these procedures follows.

```
ring r = 0,(x,y,z),lp;
ideal I = x^2+y^2+z^2-4, x^2+2*y^2-5,x*z-1;
option(redSB);
groebner(I);
```

The first and second steps correspond to the first and second rows of the series of commands. The 0 in the first command specifies the field that we consider: the rational number field. The following (x,y,z) and lp specify the variables and ordering of the considered ring, respectively, defined as r. The second command defines the ideal I by specifying the generators, which are described on the right side of the equal sign. The third step corresponds to the fourth row of the commands. The third row of the commands is an option to compute the reduced Gröbner basis. By executing these commands, we obtain the following results:

```
_[1]=2z4-3z2+1
_[2]=y2-z2-1
_[3]=x+2z3-3z
```

This indicates that we obtain the Gröbner basis as follows:

$$\left\{2z^4 - 3z^2 + 1, y^2 - z^2 - 1, x + 2z^3 - 3z\right\}. \tag{33}$$

Notably, the first element does not contain x or y. Thus, this shows that the elimination of x, y is completed using the Gröbner basis.

Example 5 *Suppose we have a physical computing system modeled as follows:*

$$\dot{x}_1 = a_1x_1 + a_2x_2 + u, \dot{x}_2 = a_3x_1 + a_4x_2, y = a_5x_1 + a_6x_2. \tag{34}$$

We compute the Gröbner basis of the ideal generated by (34) *and their derivatives up to the second order and thus obtain an input-output equation of* (34)*. Note that $a_i (i = 1, \ldots, 6)$ in* (34) *denote the parameters. The following are commands that can be used to compute it:*

```
ring r =(0,a1,a2,a3,a4,a5,a6),(X6,X5,X4,X3,X2,X1,Y3,Y2,Y1,U3,U2,U1),lp;
ideal I =X3-U1-X1*a1-X2*a2,X5-U2-X3*a1-X4*a2,X4-X1*a3-X2*a4,
X6-X3*a3-X4*a4,Y1-X1*a5-X2*a6,Y2-X3*a5-X4*a6,Y3-X5*a5-X6*a6;
option(redSB);
groebner(I);
```

Here, a1,a2,a3,a4,a5,a6 correspond to the $a_1, a_2, a_3, a_4, a_5, a_6$ of (34). The symbols beginning with U, X, Y denote the input variables, state variables, output variable, and their higher order derivatives. Us, where $1 \leq s \leq N$ and $s \in \mathbb{N}$ denotes the $(s-1)$th order derivative of u. The same holds for X and Y. By executing these commands, we obtain the following result, the Gröbner basis of the second order ideal of (34):

```
_[1]=Y3+(-a1-a4)*Y2+(a1*a4-a2*a3)*Y1+(-a5)*U2+(-a3*a6+a4*a5)*U1
_[2]=(a1*a5*a6-a2*a5^2+a3*a6^2-a4*a5*a6)*X1+(-a6)*Y2+(a2*a5+a4*a6)*Y1
      +(a5*a6)*U1
```

```
_[3]=(a1*a5*a6-a2*a5^2+a3*a6^2-a4*a5*a6)*X2+(a5)*Y2+(-a1*a5-a3*a6)*Y1
     +(-a5^2)*U1
_[4]=(a1*a5*a6-a2*a5^2+a3*a6^2-a4*a5*a6)*X3+(-a1*a6+a2*a5)*Y2
     +(a1*a4*a6-a2*a3*a6)*Y1+(-a3*a6^2+a4*a5*a6)*U1
_[5]=(a1*a5*a6-a2*a5^2+a3*a6^2-a4*a5*a6)*X4+(-a3*a6+a4*a5)*Y2
     +(-a1*a4*a5+a2*a3*a5)*Y1+(a3*a5*a6-a4*a5^2)*U1
_[6]=(a1*a5*a6-a2*a5^2+a3*a6^2-a4*a5*a6)*X5
     +(-a1^2*a6+a1*a2*a5-a2*a3*a6+a2*a4*a5)*Y2
     +(a1^2*a4*a6-a1*a2*a3*a6-a1*a2*a4*a5+a2^2*a3*a5)*Y1
     +(-a1*a5*a6+a2*a5^2-a3*a6^2+a4*a5*a6)*U2
     +(-a1*a3*a6^2+a1*a4*a5*a6+a2*a3*a5*a6-a2*a4*a5^2)*U1
_[7]=(a1*a5*a6-a2*a5^2+a3*a6^2-a4*a5*a6)*X6
     +(-a1*a3*a6+a2*a3*a5-a3*a4*a6+a4^2*a5)*Y2
     +(a1*a3*a4*a6-a1*a4^2*a5-a2*a3^2*a6+a2*a3*a4*a5)*Y1
     +(-a3^2*a6^2+2*a3*a4*a5*a6-a4^2*a5^2)*U1
```

The first element does not contain symbols beginning with X and thus describes the input-output equation of (34) as follows:

$$\ddot{y} - (a_1 + a_4)\dot{y} + (a_1 a_4 - a_2 a_3)y - w_1\dot{u} + (a_4 a_5 - a_3 a_6)u = 0. \tag{35}$$

Example 6 *Suppose that we have polynomials in $\mathbb{C}(\theta)[y_3, y_2, y_1]$ to be divided h and to divide F as follows:*

$$\begin{aligned} h =& y_3 + y_1 - (1 - {y_1}^2)y_2, \\ F =& a_4 y_1^3 + 3y_2 a_1 {y_1}^2 + (3 - 3a_4)y + 3y_3 - 3y_2(a_1 - a_4/a_1) - 3a_3 - 3a_2 a_4. \end{aligned} \tag{36}$$

We fix a monomial order of the ring to the lexcographic order such that $y_3 > y_2 > y_1$. Note that $\mathrm{LM}_<(h) \geq \mathrm{LM}_<(F)$ holds because $\mathrm{LM}_<(h) = y_3$ and $\mathrm{LM}_<(F) = y_3$. h is divided by F using singular as follows:

```
ring r=(0, a1, a2, a3, a4),(y3, y2, y1),lp;
poly F = a4*y1 ^3 + 3* y2*a1*y1 ^2 + (3 -3* a4)* y1 + 3* y3
- 3* y2 *(a1 - a4/a1) -3*a3 -3* a2*a4;
poly h =y3 + y1 - (1- y1 ^2)* y2;
NF(h, F);
```

The second command defines the polynomial to divide F. The third command corresponds to the polynomial to be divided h. The last command compute division of h by F. Because of the commands, we get the following reminder:

```
(-a1+1)*y2*y1^2+(a1^2-a1-a4)/(a1)*y2+(-a4)/3*y1^3+(a4)*y1+(a2*a4+a3)
```

$$(1 - a_1)y_2{y_1}^2 + \frac{{a_1}^2 - a_1 - a_4}{a_1 y_2} - \frac{a_4}{3{y_1}^3} + a_4 y_1 + (a_2 a_4 + a_3) \tag{37}$$

Although this example concerns non-differential polynomials, it is applicable to differential polynomials such as in Section 3. Indeed, if (y_1, y_2, y_3) is replaced by $(y, \dot{y}, \ddot{y})$, this example corresponds to the division in Section 3.

I Numerical examples of the multi-tasking property

In this section, we provide numerical examples for demonstrating the multi-tasking property of (5) discussed in Section 2.3. We consider chaotic dynamical systems called the Lorentz system [14],

$$\dot{\bar{x}}_1 = 10(\bar{x}_2 - \bar{x}_1),\ \dot{\bar{x}}_2 = \bar{x}_1(28 - \bar{x}_3) - \bar{x}_2,\ \dot{\bar{x}}_3 = \bar{x}_1\bar{x}_2 - (8\bar{x}_3/3)\,, \tag{38}$$

and the Rössler system [22],

$$\dot{\bar{x}}_1 = -\bar{x}_2 - \bar{x}_3,\ \dot{\bar{x}}_2 = \bar{x}_1 + (\bar{x}_2/10)\,,\ \dot{\bar{x}}_3 = (1/10) + \bar{x}_3(\bar{x}_1 - 14). \tag{39}$$

where data points are sampled from the first variables. We substituted these target time-series into the design guideline (6) to obtain the design inputs. Then, they are applied to (5). Fig. 4 shows

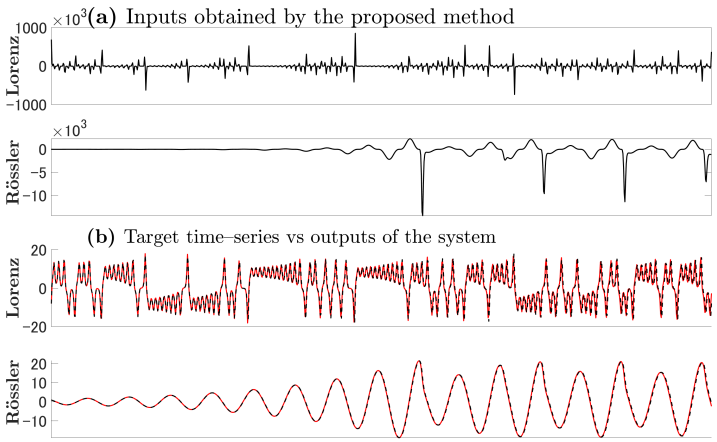

Figure 4: Generation of chaotic time-series (the Lorentz and the Rössler equations) by the physical computing system (5).

the designed inputs and the outputs of the physical computing system for generating the two types of time-series. Thus, the different time-series, both of which are obtained from chaotic dynamics, are numerically confirmed to be generated by the physical computing system based on the design guidelines derived by our approach. See Appendix J for the details on how to design initial states of (5).

J Details on estimations of consistent initial parameters

Theorems 1 and 2 require the existence of consistent initial values as defined in Definition 12. To satisfy the consistency requirement, (26) must be satisfied given a time-series and input functions. Noting that $f^{(i)}(x(0), u(0); \theta)$ and $g^{(i+1)}(x(0), u(0); \theta)$ $(i = 0, \ldots, N-1)$ can be represented as functions of $x(0), u(0), \ldots, u^{(N)}(0)$, consistent initial values $x(0), \ldots, x^{(N)}(0)$ can be computed by using this value; (26) given specific values of $\bar{y}(0), \ldots, \bar{y}^{(N)}(0), u(0), \ldots, u^{(N)}(0)$ can be regarded as a system of equations of $x(0), \ldots, x^{(N)}(0)$; thus, its solution must correspond to the consistent initial vectors.

In Experiment of Section 2.3, the initial values of (5) are fixed consistently. Since x_2 is observed, it holds that $x_2(0) = \bar{y}(0)$. Based on Definition 12, the second equation of (5) must holds, which leads to $x_1(0) = \dot{\bar{y}}(0)$. In this experiment, $\dot{\bar{y}}(0)$ are estimated via the approximation of the numerical solution of $\bar{y}$ of (38) and (39) using polynomial interpolants in the Chebyshev points [24].

In Section 3, the initial values of (9) must be fixed consistently depending on initial values of Σ_2. Here, we consider the case that initial values of van der Pol equation, which is a part of Σ_2, are known. In this case, we first estimate the values of $\bar{y}^{(1)}(0)$ and $\bar{y}^{(2)}(0)$ via Chebyshev approximation similar as in Experiment in Section 2.3. The approximated values of $\bar{y}^{(1)}(0)$ and $\bar{y}^{(2)}(0)$ together with known $\bar{y}(0)$ are then substituted into the system of equation corresponding to (26). In this procedure, $\bar{y}$ is numerically computed by *the chebop function* of *chebfun*[1]. To perform polynomial interpolation, we apply *the interp1 function* of *chebfun*. A system of equations corresponding to (26) is solved as a nonlinear least-squares problem by the Levenberg-Marquardt method, which estimates consistent initial values of Σ_1. To conclude this section, we present two additional numerical experiments. In practice, one may not be satisfied that replaceability holds only for a single time-series, as shown in Section 3. Considering this, we now confirm the replaceability with respect to the parameters of Σ_2, which allows us to consider replaceability for a set of time-series.

K Details on the experiments on the stationary task

This section provides the details of numerical example corresponding to Fig. 2(d). We consider a stationary state of Σ_2, which is defined in Section 3. In particular, we consider the time duration

[1] `https://www.chebfun.org/download/` (the 3-clause BSD license)

$t \in [100, 200]$. At this duration, states of Σ_2 forms an attractor, as shown in the red line in Fig. 2(d). Considering that Σ_2 is originally a two-dimensional system, the attractor is reconstructed in two-dimensional delayed coordinate. The time delay was set so that the autocorrelation function of the target time-series data points takes the first local minima [17][2]. Owing to the replaceability, the only condition left for the Σ_1 to generate the target time-series in stationary time could be $(\theta_1, \theta_3, \theta_4)^\top = (1, 0, 0)^\top$. Σ_1 under the similar condition as that in Fig. 2(c) can reconstruct similar attractors, as shown in Fig. 2(d). Thus, we confirmed that Σ_2 can be replaced by Σ_1 under the design guideline $(\theta_1, \theta_3, \theta_4)^\top = (1, 0, 0)^\top$ with consistent initial conditions $x(0) = (2, 0.167)^\top$ for the stationary task, and $(\theta_1, \theta_3, \theta_4)^\top = (1, 0, 0)^\top$ for the non-stationary task.

L Additional Discussion

In this study, we assumed that physical computing system is modeled by polynomial differential equations and polynomial observation equations. However, some systems are not described in these types of equations. The existence of system noise and observation noise is also of concern as mentioned in Section 4. Thus, from a technical viewpoint, the expansion of the scope of our framework in terms of models is needed. In conventional approaches [7, 15, 2] based on approximation theory of filters [2], the Stone-Weierstrass theorem [23] is one of the keys. In the theorem, errors between a continuous function that corresponds to a considered filter and polynomial functions that approximate the filter are evaluated. Thereafter, it is shown that there exists a polynomial function that approximates the function representing the filter sufficiently under the appropriate assumptions. This implies that algebraic frameworks may be applicable to models described by continuous functions as well. This may broaden the scope of our frameworks. Therefore, if algebraic approaches are combined with these conventional approaches, designing methods of physical computing system that are practically applicable may be constructed, which we leave for future work.

Another additional limitation is the application in more realistic problems. For example, problem settings of feedback controls are common for physical computing [7, 15], and this study did not cover. Typically, for feedback control of physical computing system state variables are considered. This may be integrated with the input-output maps based approaches. Furthermore, considering that algebraic design allows physical computing systems to perform non-stationary tasks, novel applications may be considered. For instance, sequence-to-sequence tasks that appear in the field of natural language processing are in our scope in principle. It is important to apply algebraic design to these applications while considering the above technical considerations, which we leave for future work.

[2] `https://github.com/manu-mannattil/nolitsa` (the 3-clause BSD license)