Infinite-dimensional next-generation reservoir computing

Lyudmila Grigoryeva [*]
Mathematics and Statistics Division, Universität Sankt Gallen, CH-9000, Switzerland

Hannah Lim Jing Ting [†] and Juan-Pablo Ortega [‡]
School of Physical and Mathematical Sciences, Nanyang Technological University, 637371, Singapore

(Received 16 December 2024; accepted 17 February 2025; published 14 March 2025)

Next-generation reservoir computing (NG-RC) has attracted much attention due to its excellent performance in spatiotemporal forecasting of complex systems and its ease of implementation. This paper shows that NG-RC can be encoded as a kernel ridge regression that makes training efficient and feasible even when the space of chosen polynomial features is very large. Additionally, an extension to an infinite number of covariates is possible, which makes the methodology agnostic with respect to the lags into the past that are considered as explanatory factors, as well as with respect to the number of polynomial covariates, an important hyperparameter in traditional NG-RC. We show that this approach has solid theoretical backing and good behavior based on kernel universality properties previously established in the literature. Various numerical illustrations show that these generalizations of NG-RC outperform the traditional approach in several forecasting applications.

DOI: 10.1103/PhysRevE.111.035305

I. INTRODUCTION

Reservoir computing (RC) [1–4] has established itself as an important tool for learning and forecasting dynamical systems [3,5–10]. It is a methodology in which a recurrent neural network with a randomly generated state equation and a functionally simple readout layer (usually linear) is trained to proxy the data-generating process of a time series. One example is the *echo state networks (ESNs)* [3,11,12], which have demonstrated excellent empirical performance and have been shown to have universal approximation properties in several contexts [13–16]. Another family with similar theoretical properties is the *state-affine systems (SAS)* introduced in [17,18] and that is prevalent in engineering applications [19] and in quantum reservoir computing (QRC) [20,21].

Despite the ease of training associated with the RC methodology, its implementation depends heavily on hyperparameters that are sometimes difficult to estimate robustly. This difficulty has motivated various authors to replace the standard RC approach with nonlinear vector autoregression in which the covariates are monomials constructed using the previous inputs. In these methods, the only hyperparameters to be chosen are the maximum order of the monomials and the number of lags of past signals. This approach has been called *next-generation reservoir computing (NG-RC)* [22–27] and has displayed excellent performance in control and spatiotemporal forecasting tasks.

One downside of the NG-RC approach is that its performance and complexity depend strongly on the above-mentioned hyperparameters. Yet, as these hyperparameters grow, the computational effort associated with NG-RC increases exponentially. One goal of this paper is to place NG-RC in the context of kernel methods. Kernels are classical tools employed in static classification, regression, and pattern recognition tasks [28–30]. Due to the representer theorem, kernels provide a way of passing inputs into higher-dimensional feature spaces where learning takes place linearly without scaling with the dimension of the feature space. By kernelizing NG-RC, we show a more computationally tractable approach to carrying out the NG-RC methodology that does not increase complexity with the hyperparameter values. In particular, we shall see that *NG-RC is a particular case of polynomial kernel regression*.

The idea that we just explained can be pushed all the way to considering all past lags and polynomial orders of arbitrarily high order in the polynomial kernel regression. Even though this leads to an infinite-dimensional covariate space (hence the title of the paper), the kernel regression can be explicitly and efficiently implemented using the recurrence properties of the *Volterra kernel* introduced in [31]. Since the Volterra kernel regression method has as covariates the left-infinite sequence of inputs and all the monomials of all degrees constructed from these inputs, this implies that, unlike NG-RC or the polynomial kernel regression, this approach is agnostic with respect to the number of lags and the order of monomials. Moreover, the Volterra kernel has been shown in [31] to be universal in the space of fading memory input/output systems with uniformly bounded inputs and, moreover, has a computational complexity that outperforms the NG-RC whenever higher-order monomials and lags are required for modeling more functionally complex systems.

The last part of the paper contains various numerical simulations that illustrate that (i) using the more computationally

[*]Contact author: lyudmila.grigoryeva@unisg.ch
[†]Contact author: hannahji001@e.ntu.edu.sg
[‡]Contact author: juan-pablo.ortega@ntu.edu.sg

tractable polynomial kernel regression, one has access to a broader feature space, which allows learning more complex systems than those typically presented when using the NG-RC methodology and (ii) the Volterra kernel is a useful tool that can produce more accurate forecasts because it allows one to take infinite lags and monomial orders into account, which is of relevance in the modeling of long-memory phenomena that exhibit functionally complex dependencies. All the codes and data necessary to reproduce the results in the paper are available at [32].

II. PRELIMINARY DISCUSSION

A. Notation

Let $\mathbb{Z}$ denote the set of integers and $\mathbb{Z}_-$ be the set of nonpositive integers. Denote by $\mathbb{R}$ the set of real numbers and the d-dimensional Euclidean space by $\mathbb{R}^d$. We often work with sequence spaces. Let the space of sequences of d-dimensional real vectors indexed by $\mathbb{Z}_-$ be denoted $(\mathbb{R}^d)^{\mathbb{Z}_-}$. Given $\tau \in \mathbb{N}$, we shall use $(\mathbb{R}^d)^\tau$ for the space of τ-long sequences of d-dimensional real inputs indexed by $\{-\tau+1, \ldots, -1, 0\}$. In the description of sequence spaces, we may also replace $\mathbb{R}^d$ with any subset of finite-dimensional Euclidean space $\mathcal{W}$. The components of sequences $\underline{\mathbf{z}} \in \mathcal{W}^{\mathbb{Z}_-}$ are given by $\mathbf{z}_i \in \mathcal{W}$, $i \in \mathbb{Z}_-$, that is, $\underline{\mathbf{z}} = (\mathbf{z}_i)_{i\in\mathcal{W}}$. Now, given $i \in \mathbb{Z}_-$, $\tau \in \mathbb{N}$, and $\mathbf{z}_i \in \mathcal{W}$, we denote by $\underline{\mathbf{z}}_i := (\ldots, \mathbf{z}_{i-1}, \mathbf{z}_i) \in \mathcal{W}^{\mathbb{Z}_-}$ and by $\underline{\mathbf{z}}_i^\tau := (\mathbf{z}_{i-\tau+1}, \ldots, \mathbf{z}_i) \in \mathcal{W}^\tau$. We use $\mathcal{Z}$ and $\mathcal{Y}$ to refer to the input space and to the output space, respectively. The examples of $\mathcal{Z}$ can be $\mathbb{R}^d$, $(\mathbb{R}^d)^\tau$, $(\mathbb{R}^d)^{\mathbb{Z}_-}$, $\mathcal{W}^{\mathbb{Z}_-}$, etc. The output space $\mathcal{Y}$ is typically finite-dimensional subsets of Euclidean space, or even subsets of $\mathbb{R}$.

B. Data generating processes

Throughout this paper, we are interested in the learning and forecasting of discrete-time, time-invariant, and causal dynamic processes that are generated by functionals of the form $H : (\mathbb{R}^d)^{\mathbb{Z}_-} \longrightarrow \mathcal{Y}$, that is, given $\underline{\mathbf{z}} \in (\mathbb{R}^d)^{\mathbb{Z}}$, we have that $\mathbf{y}_t = H(\underline{\mathbf{z}}_t)$, $t \in \mathbb{Z}$. In practice, the inputs $\underline{\mathbf{z}} \in (\mathbb{R}^d)^{\mathbb{Z}}$ can be just deterministic sequences or realizations of a stochastic process. A related data generating process that we shall be using is the *causal chains with infinite memory (CCIM)* [33–35]. These are infinite sequences $(\mathbf{y}_i)_{i\in\mathbb{Z}}$, where $\mathbf{y}_i \in \mathcal{Y}$ for all $i \in \mathbb{Z}$, are such that $\mathbf{y}_i = H(\underline{\mathbf{y}}_{i-1})$ for all $i \in \mathbb{Z}$ and some functional $H : \mathcal{Y}^{\mathbb{Z}_-} \to \mathcal{Y}$. Takens' Theorem [36] and its generalizations [37–40] guarantee that the low-dimensional observations of dynamical systems follow, under certain conditions, a dynamical prescription of this type.

C. Kernel methods

Kernels and induced RKHS. A *kernel* on $\mathcal{Z}$ is a function $K : \mathcal{Z} \times \mathcal{Z} \longrightarrow \mathbb{R}$ that is symmetric and positive semidefinite. In this context, positive semidefiniteness means that for any $\alpha_i, \alpha_j \in \mathbb{R}$, $\mathbf{z}_i, \mathbf{z}_j \in \mathcal{Z}$, $i, j \in \{1, \ldots, n\}$,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j K(\mathbf{z}_i, \mathbf{z}_j) \geqslant 0.$$

Given $\mathbf{z}_1, \ldots \mathbf{z}_n \in \mathcal{Z}$, define the *Gram matrix* or *Gramian* to be the matrix $\mathsf{K} := [K(\mathbf{z}_i, \mathbf{z}_j)]_{i,j=1}^n \in \mathbb{R}^{n\times n}$. The kernel function being symmetric and positive semidefinite is equivalent to the Gramian being positive semidefinite in a matrix sense. The next example is a family of kernel functions of importance in our discussion.

Example 1 (Polynomial kernels). Let $\mathcal{Z} \subset \mathbb{R}^d$ for some $d \in \mathbb{N}$. For any constant $c > 0$, a polynomial kernel of degree $p \in \mathbb{N}$ is the function $K^{\text{poly}} : \mathcal{Z} \times \mathcal{Z} \longrightarrow \mathbb{R}$ given by

$$K^{\text{poly}}(\mathbf{z}, \mathbf{z}') := (\mathbf{z}^\top \mathbf{z}' + c)^p, \ \mathbf{z}, \mathbf{z}' \in \mathcal{Z}.$$

Let $\mathbb{H}$ be a Hilbert space of real-valued functions on $\mathcal{Z}$ endowed with pointwise sum and scalar multiplication. Denote the inner product on $\mathbb{H}$ by $\langle\cdot,\cdot\rangle_{\mathbb{H}}$. We say that $\mathbb{H}$ is a *reproducing kernel Hilbert space (RKHS)* associated to the kernel K if the following two conditions hold. First, for all $\mathbf{z} \in \mathcal{Z}$, we have that the functions $K(\cdot, \mathbf{z}) \in \mathbb{H}$, and for all $\mathbf{z} \in \mathcal{Z}$ and all $f \in \mathbb{H}$, the *reproducing property*, $f(\mathbf{z}) = \langle f, K(\cdot, \mathbf{z})\rangle_{\mathbb{H}}$, $z \in \mathcal{Z}$, is satisfied. The maps of the form $K_{\mathbf{z}}(\cdot) := K(\cdot, \mathbf{z}) : \mathcal{Z} \longrightarrow \mathbb{R}$ are called *kernel sections*. Second, the Dirac functionals $\delta_{\mathbf{z}} : \mathbb{H} \to \mathbb{R}$ defined by $\delta_{\mathbf{z}}(f) := f(\mathbf{z})$ are continuous, for all $\mathbf{z} \in \mathcal{Z}$, with respect to the metric on $\mathbb{H}$ induced by the inner product $\langle\cdot,\cdot\rangle_{\mathbb{H}}$.

Canonical feature maps and spaces. In this setup, one may define the map $\Phi_{\mathbb{H}} : \mathcal{Z} \longrightarrow \mathbb{H}$ given by $\Phi_{\mathbb{H}}(\mathbf{z}) := K_{\mathbf{z}}$, $\mathbf{z} \in \mathcal{Z}$. Then, from the reproducing property of the RKHS, the kernel function can be written as the inner product of kernel sections. Indeed, given $\mathbf{z}, \mathbf{z}' \in \mathcal{Z}$, we can write

$$K(\mathbf{z}, \mathbf{z}') = \langle K_{\mathbf{z}'}, K_{\mathbf{z}}\rangle_{\mathbb{H}} = \langle \Phi_{\mathbb{H}}(\mathbf{z}'), \Phi_{\mathbb{H}}(\mathbf{z})\rangle_{\mathbb{H}}.$$

We call the map $\Phi_{\mathbb{H}}$ the *canonical feature map* and the RKHS $\mathbb{H}$ is referred to as its *canonical feature space*.

By the Moore-Aronszajn Theorem [30,41], any kernel has a unique RKHS, and this RKHS can be written as

$$\mathbb{H} := \overline{\text{span}\{K_{\mathbf{z}} \mid \mathbf{z} \in \mathcal{Z}\}},$$

where the bar denotes the completion with respect to the metric induced by the inner product $\langle\cdot,\cdot\rangle_{\mathbb{H}}$. This inner product obviously satisfies that for $f = \sum_{i=1}^m \alpha_i K_{\mathbf{z}_i'}$, $g = \sum_{j=1}^n \beta_j K_{\mathbf{z}_j} \in \mathbb{H}$, $\langle f, g\rangle_{\mathbb{H}} = \sum_{i=1}^m \sum_{i,j=1}^n \alpha_i \beta_j K(\mathbf{z}_i', \mathbf{z}_j)$. We use the symbol $\|\cdot\|_{\mathbb{H}}$ for the norm induced by $\langle\cdot,\cdot\rangle_{\mathbb{H}}$.

Feature maps and their associated kernels. The construction we just presented produces an RKHS and a feature map out of a kernel map. Conversely, given any Hilbert space $(H, \langle\cdot,\cdot\rangle_H)$ and a map $\Phi : \mathcal{Z} \to H$ from the set $\mathcal{Z}$ into H, we can construct a kernel map that has Φ as a feature map and H as a feature space, that is, define

$$K(\mathbf{z}, \mathbf{z}') := \langle \Phi(\mathbf{z}'), \Phi(\mathbf{z})\rangle_H, \quad \mathbf{z}, \mathbf{z}' \in \mathcal{Z}. \tag{1}$$

Such a function K is clearly symmetric and positive semidefinite; thus, it satisfies the conditions needed to be a kernel function. It can actually be proved (see [[30], Theorem 4.16]) that a map $K : \mathcal{Z} \times \mathcal{Z} \longrightarrow \mathbb{R}$ is a kernel if and only if there exists a feature map that allows K to be represented as in (1). Note that given a kernel function, the feature representation (1) is not unique in the sense that neither the feature space H nor the feature map $\Phi : \mathcal{Z} \to H$ are unique. In particular, given K, the canonical feature map and the RKHS are a choice of feature map and feature space for K, respectively.

When a kernel function K is defined as in (1) using a feature map $\Phi : \mathcal{Z} \to H$, Theorem 4.21 in [30] proves that the corresponding RKHS $\mathbb{H}$ is given by

$$\mathbb{H} = \{f : \mathcal{Z} \to \mathbb{R} \mid f(\cdot) = \langle w, \Phi(\cdot)\rangle_H, w \in H\}, \quad (2)$$

equipped with the Hilbert norm

$$\|f\|_{\mathbb{H}} := \inf\{\|w\|_H \mid w \in H \text{ such that } f(\cdot) = \langle w, \Phi(\cdot)\rangle_H\}. \quad (3)$$

Using this fact, we prove the following lemma, which shows the equality of the RKHS spaces associated with kernels with feature maps related by a linear isomorphism.

Lemma 2. Suppose K_1 and K_2 are kernels with feature spaces H_1 and H_2 and feature maps $\Phi_1 : \mathcal{Z} \to H_1$ and $\Phi_2 : \mathcal{Z} \to H_2$, respectively. Denote the corresponding RKHSs by $\mathbb{H}_1$ and $\mathbb{H}_2$. Suppose that there exists a bounded linear isomorphism $L : H_1 \to H_2$ such that $\Phi_2 = L \circ \Phi_1$, then $\mathbb{H}_1 = \mathbb{H}_2$.

Proof. By the bounded inverse theorem, L^{-1}, which is linear, is bounded. By the Riesz representation theorem, the adjoints of L and L^{-1}, denoted by L^* and $(L^{-1})^*$ are well defined. By (2), for $f \in \mathbb{H}_1$, there exists $w \in H_1$ such that $f(\mathbf{z}) = \langle w, \Phi_1(\mathbf{z})\rangle_{H_1}$ for any $\mathbf{z} \in \mathcal{Z}$. Then, it is easy to see that for any $\mathbf{z} \in \mathcal{Z}$, $f(\mathbf{z}) = \langle (L^{-1})^* w, \Phi_2(\mathbf{z})\rangle_{H_2}$ so that $\mathbb{H}_1 \subset \mathbb{H}_2$. The reverse inclusion is similar.

Kernel ridge regression. Suppose that $\mathcal{Z} \subset \mathbb{R}^d$, $\mathcal{Y} \subset \mathbb{R}$, and that a function $f : \mathcal{Z} \to \mathcal{Y}$ needs to be estimated using a finite sample of n pairs of input and outputs $\{(\mathbf{z}_i, y_i := f(\mathbf{z}_i))\}_{i=1}^n$ where $\mathbf{z}_i \in \mathcal{Z}$ and $y_i \in \mathcal{Y}$, $i = 1, \ldots, n$. If the estimation is carried out using an *empirical risk with squared loss*

$$\widehat{R}_n(g) := \frac{1}{n}\sum_{i=1}^n (g(\mathbf{z}_i) - y_i)^2,$$

and the hypothesis set is the RKHS $\mathbb{H}$ corresponding to a kernel K, the representer theorem [29] states that the minimizer of the ridge regularized empirical risk on $\mathbb{H}$, that is,

$$f^* = \arg\min_{g\in\mathbb{H}}\left\{\widehat{R}_n(g) + \Omega\left(\|g\|_{\mathbb{H}}^2\right)\right\}, \quad (4)$$

where $\Omega : (0, \infty) \longrightarrow \mathbb{R}$ is any increasing function, lies on the span of the kernel sections generated by the sample. More explicitly, the solution of (4) has the form

$$f^*(\cdot) = \sum_{i=1}^n \alpha_i^* K(\cdot, \mathbf{z}_i), \quad (5)$$

where the vector $\boldsymbol{\alpha}^* = (\alpha_1^*, \ldots, \alpha_n^*)^\top \in \mathbb{R}^n$ can be determined by solving the regularized linear (Gramian) regression problem (nonlinear in inputs, linear in covariates) associated to the Gram matrix K of the sample, that is,

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n}\left\{\frac{1}{n}\|\mathsf{K}\boldsymbol{\alpha} - \mathsf{Y}_n\|_2^2 + \Omega(\boldsymbol{\alpha}^\top\mathsf{K}\boldsymbol{\alpha})\right\}, \quad (6)$$

where $\mathsf{Y}_n := (y_1, \ldots, y_n)^\top$ and $\|\cdot\|_2$ denotes the Euclidean norm. Let the regularization function $\Omega : (0, \infty) \longrightarrow \mathbb{R}$ be defined as $\Omega(u) = \lambda_{\text{reg}} u$ for some regularization strength $\lambda_{\text{reg}} > 0$. Then, the optimization problem (6) can be rewritten as

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n}\left\{\frac{1}{n}\|\mathsf{K}\boldsymbol{\alpha} - \mathsf{Y}_n\|_2^2 + \lambda_{\text{reg}}\boldsymbol{\alpha}^\top\mathsf{K}\boldsymbol{\alpha}\right\}, \quad (7)$$

which admits the closed-form solution

$$\boldsymbol{\alpha}^* = (\mathsf{K}^2 + \lambda_{\text{reg}}\mathsf{K})^{-1}\mathsf{K}\mathsf{Y}_n. \quad (8)$$

The optimization problem formulated in (7) is usually referred to as a *kernel ridge regression*. When the kernel K that is used to implement it has a feature map $\Phi : \mathcal{Z} \to (H, \langle\cdot,\cdot\rangle_H)$ associated, the kernel ridge regression can be reformulated as a standard linear regression in which the covariates are the components of the feature map. More explicitly, due to (2) and (3), the following equality holds true:

$$\min_{g\in\mathbb{H}}\left\{\widehat{R}_n(g) + \lambda_{\text{reg}}\|g\|_{\mathbb{H}}^2\right\} = \min_{w\in H}\left\{\widehat{R}_n(\langle w, \Phi(\cdot)\rangle_H) + \lambda_{\text{reg}}\|w\|_H^2\right\}. \quad (9)$$

Then, if we set

$$w^* = \arg\min_{w\in H}\left\{\widehat{R}_n(\langle w, \Phi(\cdot)\rangle_H) + \lambda_{\text{reg}}\|w\|_H^2\right\},$$

we can conclude that

$$f^*(\cdot) = \sum_{i=1}^n \alpha_i^* K(\cdot, \mathbf{z}_i) = \langle w^*, \Phi(\cdot)\rangle, \quad (10)$$

which proves our claim in relation to interpreting the kernel ridge regression as a standard linear regression with covariates given by the feature map. In particular, for new inputs $\mathbf{z} \in \mathcal{Z}$, the estimator f^* generates the out-of-sample outputs

$$\widehat{y} = f^*(\mathbf{z}) = \sum_{i=1}^n \alpha_i^* K(\mathbf{z}, \mathbf{z}_i) = \langle w^*, \Phi(\mathbf{z})\rangle. \quad (11)$$

Kernel universality. An important feature of some kernels is their *universal* approximating properties [30,42,43] which we now define. Suppose we have a continuous kernel map K. For any compact subset $\mathcal{K} \subset \mathcal{Z}$, define the *space of kernel sections* $K(\mathcal{K})$ to be the subset of $C^0(\mathcal{K})$ given by

$$K(\mathcal{K}) := \overline{\text{span}\{K_{\mathbf{z}} \mid \mathbf{z} \in \mathcal{K}\}}. \quad (12)$$

This time around, the bar denotes the uniform closure. Note that the continuity of K, the reproducing property, and the compactness of $\mathcal{K}$ imply that the uniform closure that defines $K(\mathcal{K})$ contains the completion of the vector space $\text{span}\{K_{\mathbf{z}} \mid \mathbf{z} \in \mathcal{K}\}$. The continuous kernel K is called *kernel universal* when for any compact subset $\mathcal{K} \subset \mathcal{Z}$,

$$K(\mathcal{K}) = C^0(\mathcal{K}).$$

Many kernels used in practice are universal. When a kernel is universal, the RKHS associated to the kernel approximates arbitrarily well all continuous functions defined on compacta $\mathcal{K}$. This implies that having kernel universality ensures that the corresponding RKHS used in the kernel ridge regression problems discussed in the previous section is rich enough to approximate arbitrarily well any target (continuous) function. Examples of universal kernels include the Gaussian kernel and the Volterra kernel [31], which we discuss in detail later on in Sec. IV. On the other hand, the polynomial kernel, although popular, is not universal [44].

Note that in the framework discussed in Sec. II 2, where the behavior of outputs is determined by causal functionals or CCIMs, whenever the corresponding functional is continuous and the inputs are defined on a compact input space, kernel

universality implies that the elements of the corresponding RKHS can be used to uniformly approximate the data generating functional.

D. The NG-RC methodology

Next-generation reservoir computing, as introduced in [22], proposes to estimate a causal polynomial functional link between an explained variable y_t at time $t \in \mathbb{Z}$ and the values of certain explanatory variables encoded in the components of $\underline{\mathbf{z}_t}^\tau$ at time t and in all the τ-instants preceding it. Mathematically speaking, its estimation amounts to solving a linear regression problem that has as covariates polynomial functions of the explanatory variables in the past.

More explicitly, assume that we have a collection of n inputs and outputs $\{(\mathbf{z}_1, y_1), \ldots, (\mathbf{z}_n, y_n) \mid \mathbf{z}_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}$. For each $t \in \{\tau, \ldots, n\}$, construct the t-th τ-delay vector $\underline{\mathbf{z}_t}^\tau$, and the vector $[\underline{\mathbf{z}_t}^\tau]^k$ containing all k-degree monomials of elements of $\underline{\mathbf{z}_t}^\tau$ as follows:

$$\underline{\mathbf{z}_t}^\tau := \left(\begin{pmatrix} z_{t-\tau+1,1} \\ \vdots \\ z_{t-\tau+1,d} \end{pmatrix}, \ldots, \begin{pmatrix} z_{t,1} \\ \vdots \\ z_{t,d} \end{pmatrix} \right) \in (\mathbb{R}^d)^\tau,$$

$$\left[\underline{\mathbf{z}_t}^\tau\right]^k := \left(\prod_{s=t}^{t-\tau+1} \prod_{u=1}^{d} z_{s,u}^{k_{s,u}} \right)_{\sum_{s,u} k_{s,u}=k,\, k_{s,u}\geqslant 0}.$$

Re-indexing $\underline{\mathbf{z}_t}^\tau$ so that

$$\underline{\mathbf{z}_t}^\tau = \left(\begin{pmatrix} z_{t,1} \\ \vdots \\ z_{t,d} \end{pmatrix}, \ldots, \begin{pmatrix} z_{t,\tau d-d+1} \\ \vdots \\ z_{t,\tau d} \end{pmatrix} \right), \tag{13}$$

we can then rewrite $[\underline{\mathbf{z}_t}^\tau]^k$

$$\left[\underline{\mathbf{z}_t}^\tau\right]^k = \left(\prod_{s=1}^{\tau d} z_{t,s}^{k_{t,s}} \right)_{\substack{\sum_{s=1}^{\tau d} k_{t,s}=k, \\ k_{t,1},\ldots k_{t,\tau d}\geqslant 0}},$$

which we will use in the proof of Proposition 3.

For a chosen maximum monomial order $p \in \mathbb{N}$, define the feature vector $\Phi : (\mathbb{R}^d)^\tau \longrightarrow \mathbb{R}^N$ as

$$\Phi\left(\underline{\mathbf{z}_i}^\tau\right) = \left(1, \left[\underline{\mathbf{z}_i}^\tau\right], \left[\underline{\mathbf{z}_i}^\tau\right]^2, \ldots, \left[\underline{\mathbf{z}_i}^\tau\right]^p\right)^\top, \tag{14}$$

where N denotes the dimension of the feature space, namely,

$$N = 1 + \sum_{k=1}^{p} \binom{\tau d + k - 1}{k} = \binom{\tau d + p}{p}. \tag{15}$$

NG-RC ridge regression. NG-RC proposes as a link between inputs and outputs the solution of the ridge regression (nonlinear in inputs, linear in covariates) that uses the components of Φ as covariates, that is, it requires solving the optimization problem:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^N} \left\{ \frac{1}{n-\tau+1} \sum_{i=\tau}^{n} \left(\Phi\left(\underline{\mathbf{z}_i}^\tau\right)^\top \mathbf{w} - y_i\right)^2 + \lambda_{\text{reg}} \|\mathbf{w}\|_2^2 \right\}. \tag{16}$$

This ridge-regularized problem obviously admits a unique solution that we now explicitly write. We first collect the feature vectors and outputs into a design matrix X and an output vector Y, respectively:

$$\mathsf{X} := \begin{bmatrix} \Phi\left(\underline{\mathbf{z}_\tau}^\tau\right)^\top \\ \vdots \\ \Phi\left(\underline{\mathbf{z}_n}^\tau\right)^\top \end{bmatrix} \quad \text{and} \quad \mathsf{Y} := \begin{bmatrix} y_\tau \\ \vdots \\ y_n \end{bmatrix}.$$

The closed-form solution for the optimization problem (16) is

$$\mathbf{w}^* = \left(\mathsf{X}^\top \mathsf{X} + \lambda_{\text{reg}} \mathbb{I}_N\right)^{-1} \mathsf{X}^\top \mathsf{Y}, \tag{17}$$

where $\mathbb{I}_N$ denotes the identity matrix of dimension N. Consequently, the output $y_t \in \mathbb{R}$ of an NG-RC system at each time step $t \in \mathbb{Z}$ corresponding to an input $\underline{\mathbf{z}} \in (\mathbb{R}^d)^{\mathbb{Z}}$ is given by

$$\widehat{y_t} = (\mathbf{w}^*)^\top \Phi\left(\underline{\mathbf{z}_t}^\tau\right). \tag{18}$$

III. KERNELIZATION OF NG-RC

We now show that NG-RC can be kernelized using the polynomial kernel function introduced in Example 1. The term kernelization in this context means that, along the lines of what we saw in (10), the solution (18) of the NG-RC nonlinear regression problem can be written as the solution of the kernel ridge regression problem (4) associated to the polynomial kernel. More explicitly, we shall see that the solution function $f^*(\underline{\mathbf{z}}^\tau) = (\mathbf{w}^*)^\top \Phi(\underline{\mathbf{z}}^\tau)$ in (18) can be written as a linear combination of the kernel sections of K^{poly} generated by the data, with the coefficients obtained in (7) coming from the representer theorem. A similar result can be obtained for kernels based on Taylor polynomials as in [30].

Proposition 3. Consider a sample of n input/output observations $\{(\mathbf{z}_t, y_t)\}_{t\in\{1,\ldots,n\}}$ where $\mathbf{z}_t \in \mathbb{R}^d$ and $y_t \in \mathbb{R}$. Let $\tau \in \mathbb{N}$ be a chosen delay and let $p \in \mathbb{N}$ be a maximum polynomial order, let $K^{\text{poly}} : (\mathbb{R}^d)^\tau \times (\mathbb{R}^d)^\tau \longrightarrow \mathbb{R}$

$$K^{\text{poly}}(\underline{\mathbf{z}}^\tau, \underline{\mathbf{z}'}^\tau) = (1 + (\underline{\mathbf{z}}^\tau)^\top \underline{\mathbf{z}'}^\tau)^p, \tag{19}$$

be the τ-lagged polynomial kernel on $\mathbb{R}^{\tau d}$. Then, the kernel regression problem on the left-hand side of (9), corresponding to the input/output set $\{(\underline{\mathbf{z}_\tau}^\tau), y_\tau), \ldots, (\underline{\mathbf{z}_n}^\tau), y_n)\}$ and the kernel K^{poly} has the same solution as the NG-RC optimization problem given in (16), corresponding to $\{(\mathbf{z}_1, y_1), \ldots, (\mathbf{z}_n, y_n)\}$. In particular, the corresponding solution functions coincide, that is,

$$f^*(\underline{\mathbf{z}}^\tau) = (\mathbf{w}^*)^\top \Phi(\underline{\mathbf{z}}^\tau) = \sum_{i=0}^{n-\tau} \alpha_i^* K^{\text{poly}}_{\underline{\mathbf{z}}^\tau_{\tau+i}}(\underline{\mathbf{z}}^\tau), \tag{20}$$

for any $\underline{\mathbf{z}}^\tau \in (\mathbb{R}^d)^\tau$ and where $\mathbf{w}^* \in \mathbb{R}^N$ is the NG-RC solution (17), N as in (15), and $\boldsymbol{\alpha}^* := (\alpha_1^*, \ldots, \alpha_{n-\tau+1}^*)$ is the solution of the Gramian regression in (8) for K^{poly}.

Proof. For $i \in \{\tau, \ldots, n\}$, recall the reindexing of the τ-delay vector $\underline{\mathbf{z}_i}^\tau$ in (13) and additionally let $z_{i,0} = 1$. By the multinomial theorem, for any $i, j \in \{\tau, \ldots, n\}$,

$$K^{\text{poly}}\left(\underline{\mathbf{z}_i}^\tau, \underline{\mathbf{z}_j}^\tau\right) = \left(1 + \left(\underline{\mathbf{z}_i}^\tau\right)^\top \underline{\mathbf{z}_j}^\tau\right)^p = \left(\sum_{k=0}^{\tau d} z_{i,k} z_{j,k}\right)^p$$

$$= \sum_{\substack{k_0+k_1,\ldots+k_{\tau d}=p \\ k_0,k_1,\ldots,k_{\tau d}\geqslant 0}} \binom{p}{k_0, k_1, \ldots, k_{\tau d}} \prod_{t=0}^{\tau d} z_{i,t}^{k_t} z_{j,t}^{k_t}$$

$$= 1 + \sum_{k_0=0}^{p-1} \sum_{\substack{k_1+\ldots+k_{\tau d}=p-k_0 \\ k_1,\ldots,k_{\tau d}\geqslant 0}} \binom{p}{k_0, k_1, \ldots, k_{\tau d}} \times \prod_{t=1}^{\tau d} z_{i,t}^{k_t} \prod_{t=1}^{\tau d} z_{j,t}^{k_t}$$

$$= \left(c \odot \Phi\left(\underline{\mathbf{z}_i}^{\tau}\right)\right)^{\top} \left(c \odot \Phi\left(\underline{\mathbf{z}_j}^{\tau}\right)\right), \tag{21}$$

where $\Phi : (\mathbb{R}^d)^{\tau} \longrightarrow \mathbb{R}^N$ is the NG-RC feature map introduced in (14), $\odot$ denotes component-wise (Hadamard) multiplication, and the constant vector c is given by

$$c = \left(\sqrt{\binom{p}{k_o, k_1, \ldots, k_{\tau d}}}\right)_{\substack{k_0+k_1+\ldots+k_{\tau d}=p \\ k_0,k_1,\ldots,k_{\tau d}\geqslant 0}}^{\top} \in \mathbb{R}^N.$$

Notice that the relation (21) implies that the map $c \odot \Phi : (\mathbb{R}^d)^{\tau} \longrightarrow \mathbb{R}^N$ is a feature map for the kernel K^{poly}. Additionally, whenever $N < \infty$, the component-wise product with the vector $c \in \mathbb{R}^N$ can be written as a bounded linear isomorphism, which can be represented by the diagonal N by N matrix with the elements of c on the diagonal.

Define the kernel function $K^{\text{NG-RC}} : (\mathbb{R}^d)^{\tau} \times (\mathbb{R}^d)^{\tau} \longrightarrow \mathbb{R}$ obtained out of the dot product of the NG-RC feature vector (14), that is,

$$K^{\text{NG-RC}}(\underline{\mathbf{z}}^{\tau}, \underline{\mathbf{z}'}^{\tau}) := \Phi(\underline{\mathbf{z}}^{\tau})^{\top} \Phi(\underline{\mathbf{z}'}^{\tau}),$$

which is obviously a kernel function because the dot product is symmetric and positive semidefinite.

By the Moore-Aronszajn Theorem, K^{poly} and $K^{\text{NG-RC}}$ each have unique RKHSs associated $\mathbb{H}_{\text{poly}}$ and $\mathbb{H}_{\text{NGRC}}$, respectively. Since each of their feature maps $c \odot \Phi$ and Φ, respectively, are related by a bounded linear isomorphism, by Lemma 2 we have that,

$$\mathbb{H}_{\text{poly}} = \mathbb{H}_{\text{NG-RC}}.$$

This implies that the kernel regression problems (4) associated with K^{poly} and $K^{\text{NG-RC}}$ are identical and hence have the same solution. This observation, combined with the identity (9), proves the statement.

Example 4. In the setup of the previous proposition, consider the case $d = 1$, $\tau = 2$, and $p = 2$. In that situation, the two kernels in the previous discussion are defined as

$$K^{\text{poly}}\left(\underline{\mathbf{z}_i}^{\tau}, \underline{\mathbf{z}_j}^{\tau}\right) = \left(1 + \left(\underline{\mathbf{z}_i}^{\tau}\right)^{\top} \underline{\mathbf{z}_j}^{\tau}\right)^2$$
$$= (1 + (z_i, z_{i-1})(z_j, z_{j-1})^{\top})^2$$
$$= 1 + 2z_i z_j + 2z_{i-1} z_{j-1} + z_i^2 z_j^2 + z_{i-1}^2 z_{j-1}^2 + 2z_i z_j z_{i-1} z_{j-1},$$

and provided that the NG-RC map is given by

$$\Phi(\underline{\mathbf{z}_i}^{\tau}) = \left(1, z_j, z_{j-1}, z_j^2, z_j z_{j-1}, z_{j-1}^2\right)^{\top},$$

we have that

$$K^{\text{NG-RC}}\left(\underline{\mathbf{z}_i}^{\tau}, \underline{\mathbf{z}_j}^{\tau}\right) = \Phi\left(\underline{\mathbf{z}_i}^{\tau}\right)^{\top} \Phi\left(\underline{\mathbf{z}_j}^{\tau}\right)$$
$$= 1 + z_i z_j + z_{i-1} z_{j-1} + z_i^2 z_j^2 + z_{i-1} z_{j-1} z_i z_j + z_{i-1}^2 z_{j-1}^2.$$

As we already pointed out in the proof of Proposition 3, the same monomials appear in K^{poly} and $K^{\text{NG-RC}}$, which only differ by constants.

Another important observation that is visible in these expressions is the difference in computation complexity between NG-RC and its kernelized version introduced in Proposition 3. Recall that NG-RC produces the vector $\mathbf{w}^* \in \mathbb{R}^N$ in (20) while the polynomial kernel regression yields $\boldsymbol{\alpha}^* \in \mathbb{R}^{n-\tau+1}$. NG-RC is a nonlinear (on inputs) regression on the components of the (six-dimensional in this case) feature map Φ, and to carry it out, all those components have to be evaluated at all the data points; on the contrary, the kernelized version only requires the evaluation of K^{poly} at the data points, which is computationally simpler. The kernelized version of NG-RC is, hence, computationally more efficient. This difference is even more visible as τ, d, and p increase since the dependence (15) of the number of covariates in the NG-RC regression on those parameters makes them grow rapidly, while the behavior of the computational complexity of the polynomial kernel K^{poly} is much more favorable. This difference in computational performance between the two approaches will be more rigorously analyzed later in Sec. V 2.

IV. INFINITE-DIMENSIONAL NG-RC AND VOLTERRA KERNELS

Apart from the computational efficiency associated with the kernelized version of NG-RC, this approach allows for an extension of this methodology that would be impossible in its original feature map-based version. More explicitly, in this section, we will see that by pursuing the kernel approach, NG-RC can be extended to the limiting cases $\tau \to \infty$, $p \to \infty$, hence taking into account infinite lags into the past and infinite polynomial degrees in relation with the input series. This is a valuable feature in modeling situations in which one is obliged to remain agnostic with respect to τ and p. The natural tool to carry this out is the *Volterra kernel* introduced in [31], which is, roughly speaking, an infinite lag and infinite monomial degree counterpart of the polynomial kernel and that we recall in the following paragraphs.

Let $\pi_t : (\mathbb{R}^d)^{\mathbb{Z}_-} \longrightarrow (\mathbb{R}^d)$, $t \in \mathbb{Z}_+$, such that $\pi_t(\underline{\mathbf{z}_i}) = \mathbf{z}_{i-t}$ be the projection operator onto the $(-t)$-th term of a semi-infinite sequence. Given $\tau \in \mathbb{N}$, define the τ-time-delay operator $T_\tau : (\mathbb{R}^d)^{\mathbb{Z}_-} \longrightarrow (\mathbb{R}^d)^{\mathbb{Z}_-}$ by $\pi_t(T_\tau(\underline{\mathbf{z}})) := \pi_{t-\tau}(\underline{\mathbf{z}})$ for all $t \in \mathbb{Z}_-$. Given $M > 0$, define the space of M-bounded inputs by $K_M := \{\underline{\mathbf{z}} \in (\mathbb{R}^d)^{\mathbb{Z}_-} \mid \|\pi_t(\underline{\mathbf{z}})\|_2^2 \leqslant M, \text{ for all } t \in \mathbb{Z}_-\}$. Choose $\theta > 0$ such that $\theta^2 M^2 < 1$ and choose some λ such that $0 < \lambda < \sqrt{1 - \theta^2 M^2}$. Define the *Volterra kernel* $K^{\text{Volt}} : K_M \times K_M \longrightarrow \mathbb{R}$ by the recursion

$$K^{\text{Volt}}(\underline{\mathbf{z}}, \underline{\mathbf{z}'}) = 1 + \lambda^2 \frac{K^{\text{Volt}}(T_1(\underline{\mathbf{z}}), T_1(\underline{\mathbf{z}'}))}{1 - \theta^2 \langle \pi_0(\underline{\mathbf{z}}), \pi_0(\underline{\mathbf{z}'}) \rangle}. \tag{22}$$

The rationale behind this recursion is the definition of the Volterra kernel proposed in [31] as the kernel associated with a feature map obtained as the unique solution of a certain state space equation in an infinite-dimensional tensor space. The recursion in that state space equation implies the defining recursion in (22). Alternatively, the Volterra kernel can be introduced by writing the unique solution of (22), namely

$$K^{\mathrm{Volt}}(\underline{\mathbf{z}}, \underline{\mathbf{z}}') = 1 + \sum_{\tau=1}^{\infty} \lambda^{2\tau} \frac{1}{1-\theta^2\langle \pi_0(\underline{\mathbf{z}}), \pi_0(\underline{\mathbf{z}}')\rangle} \times \frac{1}{1-\theta^2\langle \pi_1(\underline{\mathbf{z}}), \pi_1(\underline{\mathbf{z}}')\rangle} \cdots \frac{1}{1-\theta^2\langle \pi_{\tau-1}(\underline{\mathbf{z}}), \pi_{\tau-1}(\underline{\mathbf{z}}')\rangle}. \tag{23}$$

It can be verified that the Volterra kernel is a kernel map on the space of semi-infinite sequences with real entries in the sense discussed in Sec. II 3.

The Volterra kernel as an infinite order and lag polynomial kernel. Observe that the τ-lagged polynomial kernel map (19) can be rewritten as

$$\begin{aligned} K^{\mathrm{poly}}&\left(\underline{\mathbf{z}_i}^{\tau}, \underline{\mathbf{z}_j}^{\tau}\right) \\ &= \left(1 + \left(\underline{\mathbf{z}_i}^{\tau}\right)^{\top} \underline{\mathbf{z}_j}^{\tau}\right)^p \\ &= \left(1 + \mathbf{z}_i^{\top}\mathbf{z}_j + \ldots + \mathbf{z}_{i-\tau+1}^{\top}\mathbf{z}_{j-\tau+1}\right)^p \\ &= \sum_{k=0}^{p} \underbrace{\sum_{\substack{k_0+\ldots+k_{\tau-1}=k \\ k_0,\ldots,k_{\tau-1}\geqslant 0}} \binom{p}{p-k, k_0, \ldots, k_{\tau-1}} \prod_{t=0}^{\tau-1} \left(\mathbf{z}_{i-t}^{\top}\mathbf{z}_{j-t}\right)^{k_t}}_{(*)}, \end{aligned}$$

where we notice that the term marked with $(*)$ is the sum of all monomials of order k on the variables that appear in the inner products $\mathbf{z}_i^{\top}\mathbf{z}_j, \ldots, \mathbf{z}_{i-\tau+1}^{\top}\mathbf{z}_{j-\tau+1}$. This expression yields the polynomial kernel as a polynomial of some finite degree p on the components of the input terms up to some finite lag τ.

Rewriting (23) using the geometric series, we have for $\underline{\mathbf{z}_i}, \underline{\mathbf{z}_j} \in K_M \subset (\mathbb{R}^d)^{\mathbb{Z}_-}$,

$$\begin{aligned} K^{\mathrm{Volt}}(\underline{\mathbf{z}_i}, \underline{\mathbf{z}_j}) &= 1 + \sum_{\tau=1}^{\infty} \lambda^{2\tau} \prod_{t=0}^{\tau-1} \sum_{k=0}^{\infty} (\theta^2 \mathbf{z}_{i-t}^{\top}\mathbf{z}_{j-t})^k \\ &= 1 + \sum_{\tau=1}^{\infty}\sum_{k=0}^{\infty} \lambda^{2\tau}\theta^{2k} \underbrace{\sum_{\substack{k_0+\ldots+k_{\tau-1}=k \\ k_0,\ldots,k_{\tau-1}\geqslant 0}} \prod_{t=0}^{\tau-1} (\mathbf{z}_{i-t}^{\top}\mathbf{z}_{j-t})^{k_t}}_{(*)}, \end{aligned}$$

where $(*)$ is again a sum of all monomials of order k on variables similar to the expression for K^{poly}. However, in contrast to the polynomial kernel, note that in this case, we are taking monomial combinations of arbitrarily high degree and lags with respect to $\underline{\mathbf{z}_i}$ and $\underline{\mathbf{z}_j}$. This implies that the Volterra kernel considers additional functional and temporal information about the input, which allows us to use it in situations where we have to remain agnostic about the number of lags and the degree of monomials that need to be used.

Infinite-dimensionality and universality. The discussion above hints that the Volterra kernel can be understood as the kernel induced by the feature map (21) associated with the polynomial kernel, but extended to an infinite-dimensional codomain capable of accommodating all powers and lags of the input variables. This statement has been made rigorous in [31], where the Volterra kernel was constructed out of an infinite-dimensional tensor feature space, which, in particular, makes it universal in the space of continuous functions defined on uniformly bounded semi-infinite sequences. This implies that *any continuous data-generating functional with uniformly bounded inputs can be uniformly approximated by elements in the RKHS generated by the Volterra kernel.* This is detailed in the following theorem proved in [31]. The statement uses the notation introduced in (12).

Theorem 5. Let $K^{\mathrm{Volt}} : K_M \times K_M \longrightarrow \mathbb{R}$ be the Volterra kernel given by (23) and let $K^{\mathrm{Volt}}(K_M)$ be the associated space of kernel sections. Then,

$$K^{\mathrm{Volt}}(K_M) = C^0(K_M).$$

In contrast, the polynomial kernel (equivalently NG-RC) is not universal (see [44]). These arguments suggest that the Volterra kernel should outperform polynomial kernel regressions and the NG-RC in its ability to, for example, learn complex systems. This will indeed be illustrated in numerical simulations in Sec. V.

Computation of Volterra Gramians. Even though the Volterra kernel is defined in the space of semi-infinite sequences, in applications, only finite samples of size n of the form $\{(\mathbf{z}_i, y_i)\}_{i\in\{1,\ldots,n\}}$ are available. In that situation, it is customary to construct semi-infinite inputs of the form $\underline{\mathbf{z}_i} = (\ldots, 0, 0, \mathbf{z}_1, \ldots, \mathbf{z}_{i-1}, \mathbf{z}_i) \in (\mathbb{R}^d)^{\mathbb{Z}_-}$, for each $i \in \{1, \ldots, n\}$, and we then define for that sample the Volterra Gram matrix $\mathsf{K}^{\mathrm{Volt}} \in \mathbb{R}^{n\times n}$ as

$$\mathsf{K}^{\mathrm{Volt}}_{i,j} = K^{\mathrm{Volt}}(\underline{\mathbf{z}_i}, \underline{\mathbf{z}_j}), \quad i, j \in \{1, \ldots, n\}.$$

Due to the recursive nature of the kernel map introduced in (22), the entries of the Gram matrix can be computed also recursively by

$$\mathsf{K}^{\mathrm{Volt}}_{i,j} = 1 + \frac{\lambda^2 \mathsf{K}^{\mathrm{Volt}}_{i-1,j-1}}{1-\theta^2\langle \mathbf{z}_i, \mathbf{z}_j\rangle}, \tag{24}$$

where $\mathsf{K}^{\mathrm{Volt}}_{0,0} = \mathsf{K}^{\mathrm{Volt}}_{i,0} = 1/(1-\theta^2)$ for all $i \in \{1, \ldots, n\}$.

We recall now that, due to the representer theorem, the learning problem (4) associated with the squared loss can be solved using the Gramian that we just constructed by computing (7). Moreover, the solution f^* has the form $f^*(\cdot) = \sum_{i=1}^{n} \alpha_i^* K^{\mathrm{Volt}}_{\underline{\mathbf{z}_i}}(\cdot)$, with $\alpha_1^*, \ldots, \alpha_n^*$ given by (8).

For a newly available set of inputs $\mathbf{z}_{n+1}, \ldots, \mathbf{z}_{n+h}$, the estimator can be used to forecast outputs $\widehat{y}_{n+1}, \ldots, \widehat{y}_{n+h}$. Extend the Gram matrix to a rectangular matrix $\mathsf{K}^{\mathrm{Volt}} \in \mathbb{R}^{n,\times n+h}$ by using the recursion

$$\mathsf{K}^{\mathrm{Volt}}_{i,n+j} = 1 + \frac{\lambda^2 \mathsf{K}^{\mathrm{Volt}}_{i-1,n+j-1}}{1-\theta^2\langle \mathbf{z}_i, \mathbf{z}_{n+j}\rangle},$$

that can be initialized by $\mathsf{K}^{\mathrm{Volt}}_{0,n+j} = 1/(1-\theta^2)$ for all $j \in \{1, \ldots, h\}$. Then, the forecasted outputs are

$$\widehat{y}_{n+j} = \sum_{i=1}^{n} \alpha_i^* K^{\mathrm{Volt}}_{\underline{\mathbf{z}_i}}(\mathbf{z}_{n+j}) = \sum_{i=1}^{n} \alpha_i^* \mathsf{K}^{\mathrm{Volt}}_{i,n+j}, \; j \in \{1, \ldots, h\}.$$

V. NUMERICS

A. Data generating processes and experimental setup

Simulations were performed, using each of the three estimators discussed in the paper, on three dynamic processes: the Lorenz autonomous dynamical system, the Mackey-Glass delay differential equation, and the Baba-Engle-Kraft-Kroner (BEKK) input/output system.

For the Lorenz and Mackey-Glass dynamical systems, the task consisted of performing the usual path continuation. During training, inputs are the spatial coordinates at time t and outputs are the $(t+1)$-th spatial coordinate, and estimators are trained on a collection of n input/outputs $\{(\mathbf{z}_t, \mathbf{z}_{t+1})\}_{t=1}^{n}$. To test their performance, the estimators are run autonomously. That is, after seeing initial input $\mathbf{z}_{n+1}$,the outputs $\widehat{\mathbf{z}}_{n+2}, \ldots, \widehat{\mathbf{z}}_{n+h}$, for some forecasting horizon h, are fed back into the estimator as inputs. These outputs $\widehat{\mathbf{z}}_{n+j}$ for $j = 2, \ldots, h+1$ are compared against the reserved set of testing values $\mathbf{z}_{n+j}$ for $j = 2, \ldots, h+1$, unseen by the estimators.

For the BEKK input/output system, the goal is to perform input/output forecasting. That is, during training, each estimator is given a set of inputs $\{\mathbf{z}_t\}_{t=1}^{n}$ and fitted against a set of outputs $\{\mathbf{y}_t\}_{t=1}^{n}$. Then, during testing, given a new set of unseen inputs $\{\mathbf{z}_t\}_{t=n+1}^{n+h}$, the outputs of the estimator $\{\widehat{\mathbf{y}}_t\}_{t=n+1}^{n+h}$ are compared against the actual outputs $\{\mathbf{y}_t\}_{t=n+1}^{n+h}$, unseen by the estimator.

The Lorenz system is a three-dimensional system of ordinary differential equations used to model atmospheric convection [45]. The following Lorenz system

$$\dot{x} = -10(x-y), \quad \dot{y} = 28x - y - xz, \quad \dot{z} = -\tfrac{8}{3}z + xy,$$

with the initial conditions

$$x_0 = 0, \quad y_0 = 1, \quad z_0 = 1.05,$$

was chosen. A discrete-time dynamical system was derived using Runge-Kutta45 (RK45) numerical integration with time-step 0.005 to simulate a trajectory with 15 001 points. The first 5000 points were reserved for training. The remaining points were reserved for testing. Although we consider all coordinates, one could potentially consider reconstructing the Lorenz dynamical system out of partial observations. In this case, the τ-delay would be especially important due to the celebrated Takens' embedding theorem [36]. Regardless, even in the fully observable case used in this paper, the Lorenz dynamical system lies within the premise discussed in Sec. II 2.

The Mackey-Glass equation is a first-order nonlinear delay differential equation (DDE) describing physiological control systems given by [46]. We chose the following instance of the Mackey-Glass equation

$$\dot{z} = \frac{0.2z(t-17)}{1+z(t-17)^{10}} - 0.1z(t),$$

with the initial condition function being the constant function $z(t) = 1.2$. To numerically solve this DDE, the delay interval was discretized with time step of 0.02, then the usual RK45 procedure was performed on the discretized version of the system. The resulting dataset was flattened back into a one-dimensional dataset, and to reduce the size of the dataset, the dataset was further spliced to take every 50th data point. The final dataset consisted of 7650 points, and the first 3000 points were reserved for training. The remaining points were reserved to compare against the path-continued outputs of each estimator. Due to the discretization process, the differential equation becomes a system of equations where each z_t is a function of past observations, as is assumed by our premise in Sec. II 2.

The BEKK model is an input/output parametric time series model that is used in financial econometrics in the forecasting of the conditional covariances of returns of stocks traded in the financial markets [47]. We consider d assets and the BEKK(1, 0, 1) model for their log-returns $\mathbf{r}_t$ and associated conditional covariances $\Sigma_t = \mathrm{Cov}(\mathbf{r}_t \mid \mathbf{r}_{t-1}, \mathbf{r}_{t-2}, \ldots)$ given by

$$\mathbf{r}_t = \Sigma_t^{1/2}\mathbf{z}_t, \quad \mathbf{z}_t \sim \mathrm{IIDN}(0_d, \mathbb{I}_d)$$
$$\Sigma_t = CC^\top + A\mathbf{r}_{t-1}\mathbf{r}_{t-1}^\top A^\top + B\Sigma_{t-1}B^\top,$$

where the input innovations $\mathbf{z}_t$ are Gaussian IID, and the output observations are the conditional covariances Σ_t. The diagonal BEKK specification is chosen where C is an upper-triangular matrix, and A and B are diagonal matrices of dimension d. It is known that there exists a unique stationary and ergodic solution whenever $A_{ii} > 0, |B_{ii}| < 1$ for $i = 1, \ldots, d$, which expresses Σ_t as a highly nonlinear function of its past inputs [48] (as per our premise in Sec. II 2). Since the covariance matrices are symmetric, we only need to learn the outputs $\mathbf{h}_t = \mathrm{vech}\Sigma_t) \in \mathbb{R}^q$, $q := d(d+1)/2$, where the vech operator stacks the columns of a given square matrix from the principal diagonal downward. An existing dataset from [31] was used with input dimensions of 15 and output dimensions of 120. The dataset consisted of 3760 input and output points, the first 3007 were reserved for training, and the remaining 753 were reserved for testing. Since the output points were very small, to minimize loss of accuracy due to computational truncation errors, the output values were scaled by 1000. The training output data was further normalized so that each dimension would have 0 mean and variance of 1.

Since NG-RC methodology does not typically require normalization, the NG-RC datasets were not normalized. For the polynomial kernel, kernel values could become too large and result in truncation inaccuracies, so the training inputs were normalized to have a maximum of 1 and a minimum of 0. Due to the construction of the Volterra kernel, the input sequence space into the kernel for a finite sample is truncated with zeros. We thus demean the training input data. Moreover, to avoid incurring truncation errors for θ and λ values, the maximum Euclidean norm of the inputs M is set to 1, by scaling the training input values. Note that for all estimators, normalization is always performed based only on information from the training values, then the testing data is shifted and scaled based on what was used for the training data. This prevents leakage of information such as the mean, standard deviation, maximum, minimum, etc., to the testing datasets.

B. Time complexities

Following [[29], page 280], we compute the time complexities for the NG-RC, polynomial kernel regression, and Volterra kernel regression. We assume for both kernel

TABLE I. Time complexities for all forecasting schemes discussed in this paper. Denote $\kappa = \min(p, \tau d)$.

	Training	Prediction
NG-RC	$O(n(p+\tau d)^{2\kappa} + (p+\tau d)^{3\kappa})$	$O((p+\tau d)^{\kappa})$
Polynomial	$O(n^2\tau d + n^3)$	$O(n\tau d)$
Volterra	$O(n^2 d + n^3)$	$O(nd)$

regressions, as per the procedure used in the numerical simulations, that the usual Euclidean dot product was used.

In each forecasting scheme, training involves computing the closed-form solutions (17) for the NG-RC and (8) for the polynomial and Volterra kernel regressions. For the NG-RC, to compute $\mathsf{X}^\top\mathsf{X}$ takes $O(nN^2)$ steps, recalling the definition of N given in (15). To compute matrix inversion in (17), is $O(N^3)$. The remaining matrix multiplications have complexities dominated by $O(nN^2)$. Thus, the final complexity for training weights in NG-RC is $O(nN^2 + N^3)$. On the other hand, for polynomial or Volterra kernel regressions, one needs to compute the kernel map for each entry of the Gram matrix. For the polynomial kernel, in view of (19), this is $O(\tau d)$, and for the Volterra kernel map, in view of (24), this is $O(d)$. Then, to compute the Gram matrix is $O(n^2 d\tau)$ and $O(n^2 d)$ for polynomial and Volterra kernel regression, respectively. Forecasting involves computing (18) for the NG-RC and (11) for the polynomial and Volterra kernel regressions. For each time step, the complexity is $O(N)$ for the NG-RC, $O(n\tau d)$ for the polynomial kernel regression, and $O(nd)$ for the Volterra kernel regression.

The combinatorial N term for the NG-RC can be bounded above by $(p+\tau d)^\kappa$ where $\kappa = \min(p, \tau d)$. Thus, in big-O notation, the N term can be replaced by $(p+\tau d)^\kappa$. It can then be seen that when the sample size is small, and when τ and p need not be large, the NG-RC will be faster than the polynomial and Volterra kernels. However, as τ and p grow, as is needed to learn more complex dynamical systems, the complexity for NG-RC grows exponentially, and the polynomial and Volterra kernel regressions will outperform the NG-RC significantly. For each of the forecasting schemes, the complexities associated with the training and generation of a single prediction are given in Table I.

C. Cross-validation

For each estimator, hyperparameters have to be selected. The hyperparameters that were cross-validated and the chosen values are given in Table II. Note that the washout was not cross-validated for. For both NG-RC and polynomial kernel regression, washout is the number of delays taken. For the Volterra kernel, a longer washout is needed to wash the effect of truncating input samples with zeros. A washout of 100 was sufficient to generate meaningful results both for the full training set and when the training sets were restricted during cross-validation.

For the path-continuation tasks (Lorenz and Mackey-Glass), to select hyperparameters, cross-validation was performed by splitting each training set into training-validation folds that overlapped. During validation, path continuation was performed and compared with the validation set. That is, the outputs of each estimator were fed back as inputs, and these autonomously generated outputs were compared with the validation set. With overlapping datasets, a smaller training set would be sufficient to create multiple training folds starting from different initial points, such that in each training fold, the estimator has sufficient time to capture dynamics during training. Then, during the validation phase for each fold, for a good estimator, there would be dynamical evolution in the outputs generated by the estimator. For example, the estimator did not just fit the average. This leads to meaningful validation set errors which improves the selection process for optimal hyperparameters.

For the BEKK input/output forecasting task, the usual time-series training-validation folds were used. That is, the training dataset was split into equally sized sets where the i-th training fold was the concatenation of the first i sets and the i-th validation fold was $(i+1)$-set. For input/output forecasting, where estimator sees, during forecasting, a new set of inputs, this method of cross-validating turned out to be sufficient for estimators to capture the dynamics of input and output variables. Note that cross-validation training and testing were made to mimic as closely as possible the actual task to be performed on the full training set, so normalization in each fold was also performed as would have been done on the full training dataset.

The range of parameters cross-validated was chosen so that the regularization was performed over the same set of values. As detailed in the previous section, the time complexity for NG-RC grows exponentially for larger lag and degree hyperparameters. It was thus impractical to cross-validate over a large range of parameters as with each increase in a number of lag or maximum degree of monomials, the computational time would grow exponentially. Thus, only a smaller range

TABLE II. Chosen parameters for each estimator and dataset.

System	Estimator	Washout	Hyperparameters
Lorenz	NG-RC	3	$(\tau = 3, p = 2, \lambda_{\text{reg}} = 1 \times 10^{-7})$
	Polynomial	6	$(\tau = 6, p = 2, \lambda_{\text{reg}} = 1 \times 10^{-6})$
	Volterra	100	$(\lambda \approx 0.286, \theta = 0.3, \lambda_{\text{reg}} = 1 \times 10^{-10})$
Mackey-Glass	NG-RC	4	$(\tau = 4, p = 5, \lambda_{\text{reg}} = 1 \times 10^{-7})$
	Polynomial	17	$(\tau = 17, p = 4, \lambda_{\text{reg}} = 1 \times 10^{-5})$
	Volterra	100	$(\lambda \approx 0.859, \theta = 0.3, \lambda_{\text{reg}} = 1 \times 10^{-9})$
BEKK	NG-RC	1	$(\tau = 1, p = 2, \lambda_{\text{reg}} = 0.1)$
	Polynomial	1	$(\tau = 1, p = 2, \lambda_{\text{reg}} = 0.1)$
	Volterra	100	$(\lambda = 0.72, \theta = 0.6, \lambda_{\text{reg}} = 1 \times 10^{-3})$

TABLE III. Error values for each estimator and dataset. A 20% deviation is allowed for T_{valid}. NMSE, MAE, MdAE, and MAPE are computed up to $\lceil T_{\text{valid}} \rceil$ for Lorenz and Mackey-Glass. PSDE and W_1 are computed for the full dataset (Lorenz was sampled). PSDE for BEKK is scaled by 10^{-4}.

System	Estimator	T_{valid}	NMSE	MAE	MdAE	MAPE	PSDE	W_1
Lorenz	NG-RC	7.178	0.379	1.827	0.0897	1.598	7.606	2.114
	Polynomial	9.126	0.188	1.155	0.0377	1.230	**7.169**	**1.683**
	Volterra	**10.566**	**0.0934**	**0.428**	**0.00385**	**0.222**	8.325	1.982
Mackey-Glass	NG-RC	0.3	0.980	0.188	0.174	0.235	28.815	0.155
	Polynomial	7.035	0.0699	0.0274	0.00751	0.0329	6.831	0.00147
	Volterra	**8.305**	**0.0333**	**0.0162**	**0.00202**	**0.0190**	**5.059**	**0.00138**
BEKK	NG-RC		0.875	0.0204	0.0166	0.644	1.565	0.332
	Polynomial		0.877	0.0204	0.0166	0.642	1.140	0.337
	Volterra		**0.619**	**0.0170**	**0.0139**	**0.634**	**0.963**	**0.319**

of parameters could be cross-validated over. The polynomial kernel was cross-validated over a larger space of the same delay and degree hyperparameters. When cross-validating for the Lorenz system, with fully observable coordinates, only one delay suffices to reconstruct the dynamical system. However, in practice, a higher number of lags may offer superior predictive performance. Hence we allowed for up to 10 lags to cross-validate over in the case of Lorenz. For the rest of the datasets, up to 101 lags were cross-validated over. Such a large range of hyperparameters was possible because by Proposition 3 and Example 4 the polynomial kernel regression uses the same covariates but is faster when more covariates are considered. Finally, mean squared error was chosen to be the metric over which the best hyperparameters were chosen.

D. Results

Pointwise and climate metrics were used to evaluate the performance of the estimators. Pointwise metrics are distance functions that evaluate the error committed by estimators from time step to time step. Climate metrics, see also [49], are performance metrics that evaluate whether the statistical or physical properties are similar to the true system. We also consider the valid prediction time for each estimator in Lyapunov time for the two chaotic attractors (Lorenz and Mackey-Glass).

Denoting the true value $\mathbf{y}$, the predicted value $\widehat{\mathbf{y}}$, and the testing set size h, the following pointwise error metrics were chosen: the normalized mean squared error (NMSE) given by

$$\text{NMSE}(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{1}{d} \sum_{u=1}^{d} \frac{\sum_{i=1}^{h} (y_{u,i} - \widehat{y}_{u,i})^2}{\sum_{i=1}^{h} (y_{u,i} - \bar{y}_u)^2},$$

where $\bar{y}_u$ denotes the average of the u-th dimension of the vector $\mathbf{y}$ over time steps $i = 1, \ldots, h$, the mean absolute error (MAE) given by

$$\text{MAE}(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{1}{h} \sum_{i=1}^{h} \|\mathbf{y}_i - \widehat{\mathbf{y}}_i\|_1,$$

where $\|\cdot\|_1$ is the 1-norm, the median absolute error (MdAE) defined as

$$\text{MdAE}(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{1}{d} \sum_{u=1}^{d} \text{median}\big(\{|y_{u,i} - \widehat{y}_{u,i}|\}_{i=1}^{h}\big),$$

and the mean absolute percentage error (MAPE)

$$\text{MAPE}(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{1}{hd} \sum_{u=1}^{d} \sum_{i=1}^{h} \frac{|y_{u,i} - \widehat{y}_{u,i}|}{\max(\varepsilon, |y_i|)},$$

for some very small $\varepsilon > 0$.

Whether the estimator replicated the climate of the true time series was measured by considering the difference in the true and estimated power spectral density (PSD) and the difference in distributions using the Wasserstein-1 distance.

The PSD is the Fourier transform of the autocovariance function and represents the time series in its frequency domain [50]. The PSD of each time series was estimated by periodograms computed using Welch's method, provided by `scipy.signal.welch`, with the Hann window. The number of points per segment was chosen by visual inspection for a balance between frequency resolution and error variance. When the PSD tapers off to zero after some frequency F, the remaining frequencies are not considered in the final difference. Finally, the PSD error (PSDE) is computed by taking

$$\text{PSDE}(\text{PSD}, \widehat{\text{PSD}}) = \sum_{u=1}^{d} \sum_{f=1}^{F} \frac{|\text{PSD}_{u,f} - \widehat{\text{PSD}}_{u,f}|}{\text{PSD}_{u,f}},$$

where PSD is the periodogram of the actual data and $\widehat{\text{PSD}}$ is the periodogram of the estimated data. The subscript u, f denotes the f-th term in the PSD sequence in the u-th dimension.

The Wasserstein-1 distance or earth mover distance, arising out of optimal transport [51], is a measure of the distance between probability measures and is given by

$$W_1(\mu, \widehat{\mu}) = \inf_{\pi \in \Gamma(\mu, \widehat{\mu})} \int \|\mathbf{y} - \widehat{\mathbf{y}}\|_2 d\pi(\mathbf{y}, \widehat{\mathbf{y}}), \tag{25}$$

where $\Gamma(\mu, \widehat{\mu})$ is the set of probability distributions whose marginals are μ and $\widehat{\mu}$ on the first and second factors, respectively. μ and $\widehat{\mu}$ are the joint distributions of $\mathbf{y}$ and $\widehat{\mathbf{y}}$, respectively.

The Wasserstein-1 distance was computed to compare the distributions of the true and estimated systems in our paper. For one-dimensional systems, the Wasserstein-1 distance can be computed using `scipy.stats.wasserstein_distance` which uses the equivalent Cramer-1 distance, that is,

$$W_1(\mu, \widehat{\mu}) = \int_{\mathbb{R}} |\text{CDF}(y) - \widehat{\text{CDF}}(y)| dy,$$

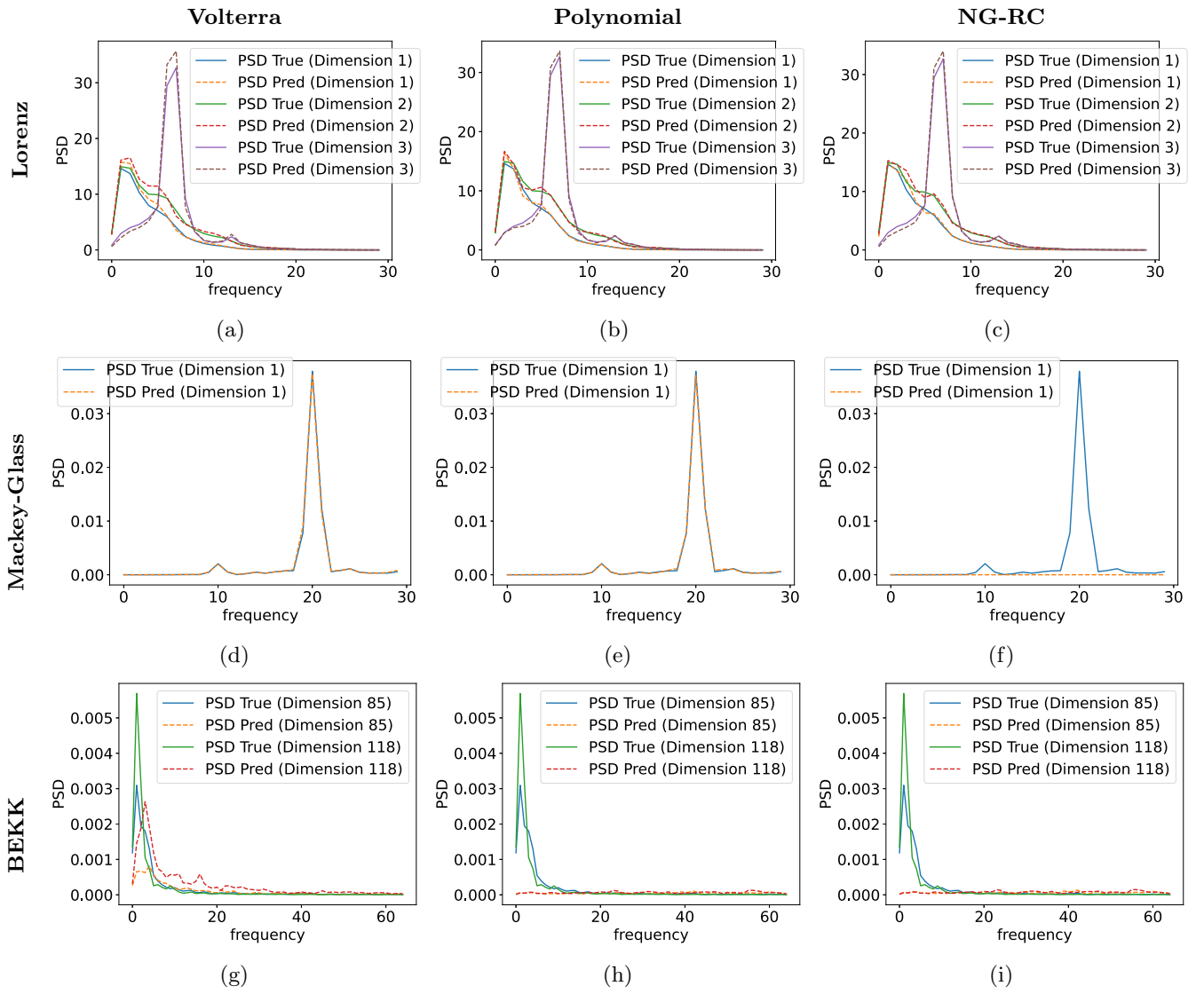

FIG. 1. The PSD for: [1(a)] the Lorenz system for the Volterra kernel, [1(b)] the polynomial kernel, and [1(c)] NG-RC. Dimension 1 corresponds to x, dimension 2 to y, and dimension 3 to z. PSD for Mackey-Glass system for Volterra kernel is [1(d)], polynomial kernel is [1(e)], and NG-RC is [1(f)]. Dimension 1 refers to the z value. PSD for BEKK for Volterra kernel is [1(g)], polynomial kernel is [1(h)], and NG-RC is [1(i)]. Only the two dimensions with the most visually prominent PSD are displayed (dimensions 85 and 118).

where $\mu, \widehat{\mu}$ are the probability distributions of y and $\widehat{y}$, respectively while CDF, $\widehat{\text{CDF}}$ denote the cumulative distributive functions. For time series in d-dimensions, using `scipy.stats.wasserstein_distance_nd`, corresponds to solving the linear programming problem in (25). It is noted that computing the Wasserstein distance for the multidimensional case is significantly more computationally expensive than in the one-dimensional case. In the case of the Lorenz dynamical system, sampling had to be performed to make using `scipy.stats.wasserstein_distance_nd` tractable.

Finally, we note that for the dynamical systems Lorenz and Mackey-Glass, the Lyapunov time, defined to be the inverse of the top Lyapunov exponent, is a timescale for which a chaotic dynamical system is predictable. In deterministic path-continuing tasks, which were carried out for the Lorenz and Mackey-Glass dynamical systems, we measure the valid prediction time percent, T_{valid}, which is the Lyapunov time taken for the predicted dynamics to differ from the true dynamics by 20%. A similar metric was used in [49].

The performance of the estimators was measured in the following manner for the dynamical systems. First, the valid prediction time was computed. Then, the ceiling of the best-performing valid prediction time is taken. The point-to-point error metrics NMSE, MAE, MdAE, and MAPE are measured only up to $\lceil T_{\text{valid}} \rceil$. Beyond the characteristic predictable timescale given by the Lyapunov time, it is not meaningful to measure the step-to-step error as the trajectories diverge exponentially according to the Lyapunov exponent. To determine if the climate is well replicated over the full testing dataset, the climate metrics PSDE and W_1 are computed for the full testing dataset, with the Lorenz needing to be sampled to compute W_1. The errors are reported in Table III.

In more complex systems such as Mackey-Glass and BEKK, the Volterra reservoir and polynomial kernel regressions easily outperform the NG-RC because they have access

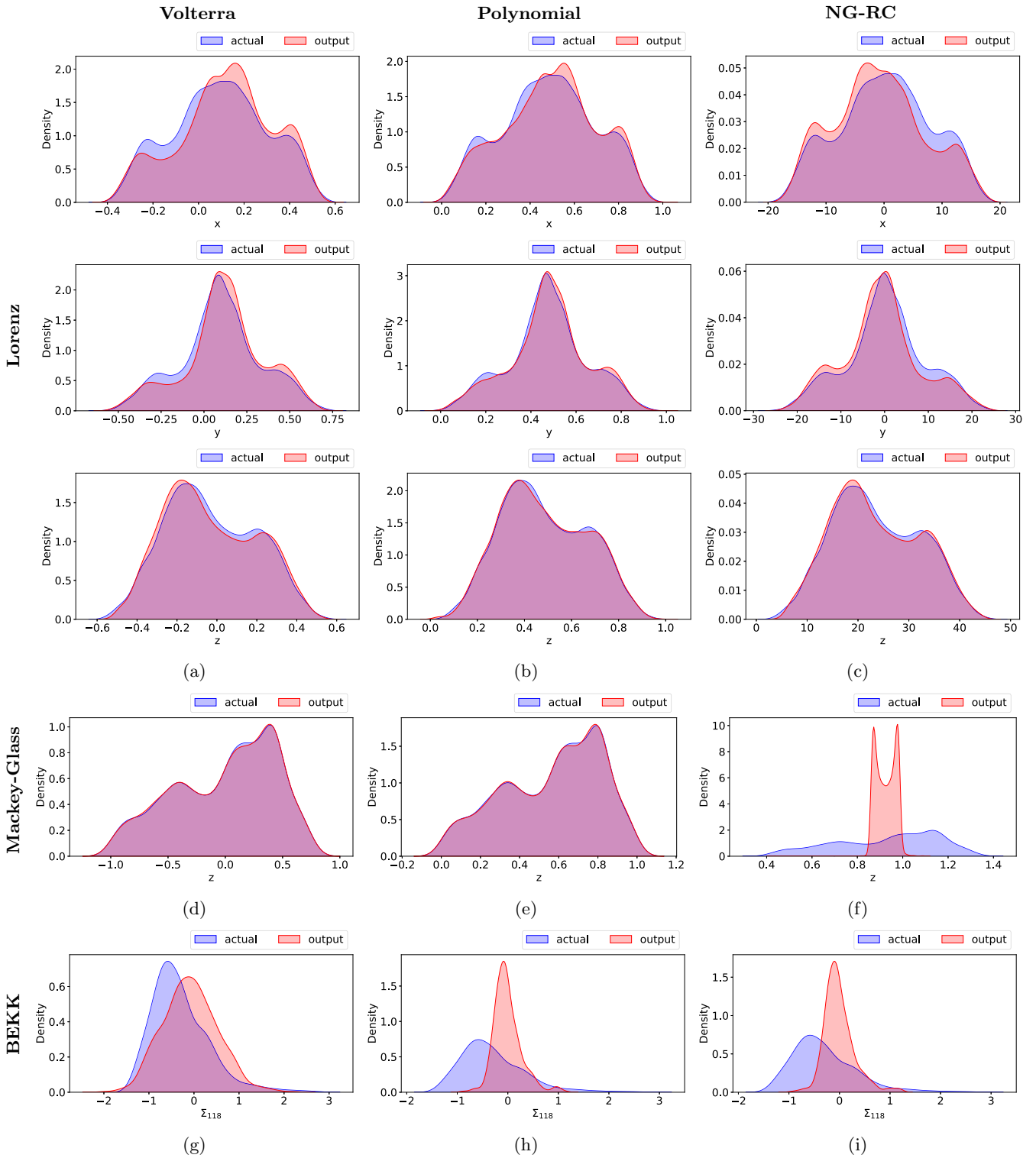

FIG. 2. The distributions for Lorenz system for the Volterra kernel is in [1(a)], the polynomial kernel is in [1(b)], and NG-RC is in [1(c)]. The distribution for the Mackey-Glass system for Volterra kernel is [1(d)], the polynomial kernel is [1(e)], and NG-RC is [1(f)]. The distributions for BEKK for Volterra kernel is [1(g)], the polynomial kernel is [1(h)], and NG-RC is [1(i)]. Only one dimension is displayed (dimension 118).

to much richer feature spaces. In second-order systems such as the Lorenz system, even though pointwise errors perform better than in the polynomial and Volterra kernel regressions, the climate metrics indicate that the NG-RC better captured the climate of the true Lorenz dynamical system. It could be that a lower-order system offered by the NG-RC acts as a better proxy for lower-order true dynamical systems, which accounts for the difference in climate performance. This difference in

climate replication performance, however, is only slight. Observe that in Fig. 1, all estimators capture the power spectral density of the original system well, even if the Volterra kernel is slightly outperformed by the polynomial kernel. For even more complex systems, both the kernel regression methods can capture the climate of the true dynamical system but the same cannot be said for NG-RC. A similar story holds when one considers the Wasserstein-1 distance. The distributions are in Fig. 2. Even though the Wasserstein-1 distance for Volterra performs the poorest, the difference in distribution performance is still small, and the bulk of the distribution is still replicated. On the other hand, for complex systems such as the BEKK, the polynomial kernel and the NG-RC fail to replicate the climate of the original system completely.

We also observe that, especially in the case of BEKK, when significantly complex dynamical systems are being learned, considering large but finite lags or monomial degrees may be insufficient. Even if finite lags are sufficient, the Gram or feature matrix values may, anyway, be too large to be handled with finite precision. In such cases, the Volterra kernel regression significantly outperforms the other two methods because it is agnostic to the lags and monomial powers, and so its Gram values do not grow with the feature choice. Moreover, as we saw in Sec. IV, taking infinite lag and monomial powers into consideration offers a rich feature space and makes the associated RKHS universal, meaning that it can approximate complex systems to any desired accuracy.

Overall, we find that the numerical simulations illustrate the points made in the theory discussed above. First, since the optimization problem in the NG-RC methodology is equivalent to that solved in polynomial kernel regression (as per Proposition 3), in more complex systems, it is better to kernelize to access computationally a richer feature space. This is illustrated especially by the superior performance of the polynomial kernel regression against the NG-RC in the Mackey-Glass system. Second, using the universal Volterra kernel, one can consider infinite lag and monomial powers, which is especially advantageous when learning significantly more complex systems, such as the BEKK input/output systems. In such cases, limited to finite lags and monomial orders, which need to be chosen (the Volterra kernel is agnostic to them), the polynomial kernel regression and NG-RC methodologies fail.

ACKNOWLEDGMENTS

The authors thank Daniel Gauthier for insightful discussions about the relation between NG-RC and the results in this paper. L.G. and J.-P.O. thank the hospitality of the Nanyang Technological University and the University of St. Gallen, respectively; it is during respective visits to these two institutions that some of the results in this paper were obtained. H.L.J.T. is funded by a Nanyang President's Graduate Scholarship of Nanyang Technological University. J.-P.O. acknowledges partial financial support from the School of Physical and Mathematical Sciences of the Nanyang Technological University.

DATA AVAILABILITY

The data that support the findings of this article are openly available [32].

[1] H. Jaeger, The 'echo state' approach to analysing and training recurrent neural networks with an erratum note, *Tech. Rep.* (German National Research Center for Information Technology, 2010).

[2] W. Maass, T. Natschläger, and H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations, Neural Comput. **14**, 2531 (2002).

[3] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science **304**, 78 (2004).

[4] W. Maass, in *Computability In Context: Computation and Logic in the Real World*, edited by S. S. Barry Cooper and A. Sorbi (Imperial College Press, London, 2011), Chap. 8, pp. 275–296.

[5] L. Grigoryeva, J. Henriques, L. Larger, and J.-P. Ortega, Stochastic time series forecasting using time-delay reservoir computers: performance and universality, Neural Networks **55**, 59 (2014).

[6] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data, Chaos **27**, 121102 (2017).

[7] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach, Phys. Rev. Lett. **120**, 024102 (2018).

[8] Z. Lu, B. R. Hunt, and E. Ott, Attractor reconstruction by machine learning, Chaos **28**, 061104 (2018).

[9] A. Wikner, J. Pathak, B. R. Hunt, I. Szunyogh, M. Girvan, and E. Ott, Using data assimilation to train a hybrid forecast system that combines machine-learning and knowledge-based components, Chaos **31**, 053114 (2021).

[10] T. Arcomano, I. Szunyogh, A. Wikner, J. Pathak, B. R. Hunt, and E. Ott, A hybrid approach to atmospheric modeling that combines machine learning with a physics-based numerical model, J. Adv. Model. Earth Syst. **14**, e2021MS002712 (2022).

[11] M. B. Matthews, On the uniform approximation of nonlinear discrete-time fading-memory systems using neural network models, Ph.D. thesis, ETH Zürich, 1992.

[12] M. B. Matthews, Approximating nonlinear fading-memory operators using neural network models, Circuits Syst. Signal Process. **12**, 279 (1993).

[13] L. Grigoryeva and J.-P. Ortega, Echo state networks are universal, Neural Networks **108**, 495 (2018).

[14] L. Gonon and J.-P. Ortega, Reservoir computing universality with stochastic inputs, IEEE Trans. Neural Netw. Learning Syst. **31**, 100 (2020).

[15] L. Gonon and J.-P. Ortega, Fading memory echo state networks are universal, Neural Networks **138**, 10 (2021).
[16] L. Gonon, L. Grigoryeva, and J.-P. Ortega, Approximation error estimates for random neural networks and reservoir systems, Ann. Appl. Probab. **33**, 28 (2023).
[17] E. D. Sontag, Realization theory of discrete-time nonlinear systems: Part I-The bounded case, IEEE Trans. Circuit Syst. **26**, 342 (1979).
[18] L. Grigoryeva and J.-P. Ortega, Universal discrete-time reservoir computers with stochastic inputs and linear readouts using non-homogeneous state-affine systems, J. Mach. Learn. Res. **19**, 24 (2018).
[19] H. D. Van Mien and D. Normand-Cyrot, Nonlinear state affine identification methods: applications to electrical power plants, Automatica **20**, 175 (1984).
[20] R. Martínez-Peña and J.-P. Ortega, Quantum reservoir computing in finite dimensions, Phys. Rev. E **107**, 035306 (2023).
[21] R. Martínez-Peña and J.-P. Ortega, Input-dependence in quantum reservoir computing, arXiv:2412.08322.
[22] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. S. Barbosa, Next generation reservoir computing, Nat. Commun. **12**, 5564 (2021).
[23] E. Bollt, On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD, Chaos **31**, 013108 (2021).
[24] W. A. S. Barbosa and D. J. Gauthier, Learning spatiotemporal chaos using next-generation reservoir computing, Chaos **32**, 093137 (2022).
[25] R. M. Kent, W. A. S. Barbosa, and D. J. Gauthier, Controlling chaotic maps using next-generation reservoir computing, Chaos **34**, 023102 (2024).
[26] R. Kent, W. Barbosa, and D. Gauthier, Controlling chaos using edge computing hardware, Nat. Commun. **15**, 3886 (2024).
[27] I. Ratas and K. Pyragas, Application of next-generation reservoir computing for predicting chaotic systems from partial observations, Phys. Rev. E **109**, 064215 (2024).
[28] B. Schölkopf and A. J. Smola, *Learning with Kernels* (MIT Press, London, 2002).
[29] M. Mohri, A. Rostamizadeh, and A. Tawalkar, *Foundations of Machine Learning*, 2nd ed. (The MIT Press, London, 2018).
[30] A. Christmann and I. Steinwart, *Support Vector Machines* (Springer, New York, 2008).
[31] L. Gonon, L. Grigoryeva, and J.-P. Ortega, Reservoir kernels and Volterra series, arXiv:2212.14641.
[32] L. Grigoryeva, H. J. T. Lim, and J.-P. Ortega, kernelngrcvolterra, https://github.com/Learning-of-Dynamic-Processes/kernelngrcvolterra.
[33] J. Dedecker, P. Doukhan, G. Lang, J. R. León, S. Louhichi, and C. Prieur, *Weak Dependence: With Examples and Applications* (Springer Science + Business Media, New York, 2007).
[34] P. Doukhan and O. Wintenberger, Weakly dependent chains with infinite memory, Stochastic Processes Appl. **118**, 1997 (2008).
[35] P. Alquier and O. Wintenberger, Model selection for weakly dependent time series forecasting, Bernoulli **18**, 883 (2012).
[36] F. Takens, Detecting strange attractors in turbulence, in *Dynamical Systems and Turbulence, Warwick 1980*, edited by D. Rand and L.-S. Young (Springer Berlin Heidelberg, Berlin, Heidelberg, 1981), pp. 366–381.
[37] L. Kocarev and U. Parlitz, General approach for chaotic synchronization with applications to communication, Phys. Rev. Lett. **74**, 5028 (1995).
[38] L. Kocarev and U. Parlitz, Generalized synchronization, predictability, and equivalence of unidirectionally coupled dynamical systems, Phys. Rev. Lett. **76**, 1816 (1996).
[39] L. Grigoryeva, A. G. Hart, and J.-P. Ortega, Chaos on compact manifolds: Differentiable synchronizations beyond the Takens theorem, Phys. Rev. E **103**, 062204 (2021).
[40] L. Grigoryeva, A. G. Hart, and J.-P. Ortega, Learning strange attractors with reservoir systems, Nonlinearity **36**, 4674 (2023).
[41] N. Aronszajn, Theory of reproducing kernels, Trans. Amer. Math. Soc. **68**, 337 (1950).
[42] C. A. Micchelli, Y. Xu, and H. Zhang, Universal Kernels, J. Mach. Learn. Res. **7**, 2651 (2006).
[43] A. Christmann and I. Steinwart, Universal Kernels on Non-Standard Input Spaces, in *Advances in Neural Information Processing Systems*, edited by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (Curran Associates, Inc., New York, 2010), Vol. 23.
[44] B. W. Wenjian Chen and H. Zhang, Universalities of reproducing kernels revisited, Appl. Anal. **95**, 1776 (2016).
[45] E. N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. **20**, 130 (1963).
[46] M. C. Mackey and L. Glass, Oscillation and chaos in physiological control systems, Science **197**, 287 (1977).
[47] R. F. Engle and F. K. Kroner, Multivariate simultaneous generalized ARCH, Econometric Theory **11**, 122 (1995).
[48] F. Boussama, F. Fuchs, and R. Stelzer, Stationarity and geometric ergodicity of BEKK multivariate GARCH models, Stochastic Processes Appl. **121**, 2331 (2011).
[49] A. Wikner, J. Harvey, M. Girvan, B. R. Hunt, A. Pomerance, T. Antonsen, and E. Ott, Stabilizing machine learning prediction of dynamics: Novel noise-inspired regularization tested with reservoir computing, Neural Networks **170**, 94 (2024).
[50] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting* (Springer, New York, 2002), p. 434.
[51] C. Villani, *Optimal Transport: Old and New* (Springer, Heidelberg, 2009).