Kernel-based methods for Volterra series identification [⋆,⋆⋆]

Alberto Dalla Libera [a,b], Ruggero Carli [a], Gianluigi Pillonetto [a]

[a] *Department of Information Engineering, University of Padova, Via Gradenigo 6/B, 35131 Padova, Italy*

[b] *Women's and Children's health Department, University of Padova, Via Giustiniani 3, 35131 Padova, Italy*

Abstract

Volterra series approximate a broad range of nonlinear systems. Their identification is challenging due to the curse of dimensionality: the number of model parameters grows exponentially with the complexity of the input-output response. This fact limits the applicability of such models and has stimulated recently much research on regularized solutions. Along this line, we propose two new strategies that use kernel-based methods. First, we introduce the *multiplicative polynomial kernel* (MPK). Compared to the standard polynomial kernel, the MPK is equipped with a richer set of hyperparameters, increasing flexibility in selecting the monomials that really influence the system output. Second, we introduce the *smooth exponentially decaying multiplicative polynomial kernel* (SED-MPK), that is a regularized version of MPK which requires less hyperparameters, allowing to handle also high-order Volterra series. Numerical results show the effectiveness of the two approaches.

Key words: Nonlinear system identification; Nonparametric methods; Time series modelling

1 Introduction

In many real applications, linear models cannot adequately describe dynamic systems. This can be due to the presence of saturations, quantizers or static nonlinearities at the input and/or the output [23][Section 5]. Even if some insight on the nonlinearities can be available, the formulation of parametric models from finite data records is a difficult task [14,22,35]. In particular, nonlinear system identification is often seen as an extended parametric regression where the choice of regressors and basis functions plays a crucial role. In this context, Volterra series are especially useful since they can represent a broad range of nonlinear systems [30,5,7].
In discrete-time, Volterra series are connected with Taylor expansion of the input-output map. Considering systems with finite memory (current output depends on a finite number m of past inputs and outputs), the r-th order Volterra series is represented as the convolution between the Volterra maps [1] and all the possible monomials up to order r (function of past inputs and outputs). Identification is difficult due to the curse of dimensionality: the number of monomials grows quickly w.r.t. the adopted polynomial degree r and system memory m. Hence, a careful selection of the relevant components to be included in the model is crucial to control complexity, a problem known as regressor selection.

Suboptimal solutions are often adopted, e.g. greedy approaches like forward orthogonal least squares [6,1] and its many variants [17][Section 3]. Another approach uses variance analysis (ANOVA) [21]. These regressor selection methods have however difficulties in handling high-dimensional spaces, as e.g. illustrated in [22] where the divide-and-conquer method TILIA is introduced to mitigate this problem. An interesting option is joint estimation and variable selection whose aim is to automatically set to zero groups of variables in the regression vector. This can be performed exploiting the ℓ_1-norm that leads to LASSO [39], also implementable using LARS [9], a less greedy version of classical forward selection.

[⋆] This paper was not presented at any IFAC meeting. Corresponding author A. Dalla Libera Tel. +39-049-827-7600 Fax +39-049-827-7699.

[⋆⋆] This work has been in part supported by the H2020 FETPROACT-01-2019 project A BIONIC INVISIBLE PANCREAS TO FORGET DIABETES - FORGETDIABETES: 951933

Email addresses: `dallaliber@dei.unipd.it` (Alberto Dalla Libera), `carlirug@dei.unipd.it` (Ruggero Carli), `giapi@dei.unipd.it` (Gianluigi Pillonetto).

[1] These are typically called Volterra kernels in the literature but we adopt this terminology to avoid confusion with the concept of kernels taken from machine learning and introduced later on.

Alternative solutions rely on kernel-based regularization [32]. Positive definite kernels are especially important since they implicitly include a large (possibly infinite) number of basis functions. They also define particular spaces, the so called Reproducing Kernel Hilbert spaces (RKHS), and the related norms can be exploited to control complexity. In particular, regularized least squares, also called regularization networks in [28], determine the unknown system by minimizing an objective sum of two terms. The first is a quadratic loss, accounting for data fit, and the second is a regularization term given by the RKHS squared norm. The balance between these two contributions is given by the regularization parameter, typically estimated from data through marginal likelihood maximization or cross validation [16,29,25]. The crucial aspect in the design of a regularization network is the kernel choice. In system identification the Gaussian kernel is often used to embed just expected smoothness of the input-output map, see e.g. [10,19,40] and also [12,13,15] for state-space approaches. This kernel has however some limitations in system identification. In those regions of the regressor space where few data are collected, output prediction just decays to zero. In this paper, we instead focus on more structured kernels connected with Volterra series. They are discussed in [11] using the polynomial kernel which encodes all the monomials up to the desired degree r. Such implicit representation makes computationally feasible the handling of high-order Volterra series. However, the associated regularization networks have some drawbacks and can overfit the data as stated in [29] (chapter 4.2.2). The reason is that the norm associated with the polynomial kernel is sometimes not able to well control the high number of monomials (implicitly) introduced in the estimation process.

More recent work on regularized Volterra series can be found in [2]. Here, authors do not use kernels to encode monomials but focus on how to control the variance of the monomial coefficients via regularization. This is performed extending some ideas developed for linear system identification in [26], and embedding in a Volterra map smooth-exponential decay concepts, i.e., the fact that as the lag increases inputs should have less effect on the output. In this way, the system memory m is replaced by continuous hyperparameters that are connected with memory fading concepts and can be estimated from data. Despite the remarkable results on both simulated and real data [2–4], one limitation is that computational and memory requirements increase quickly with the polynomial degree r and the memory m. A possible solution to this issue has been proposed in [37,38], where, instead of using monomials, orthonormal basis functions, such as Laguerre or Kautz, are considered. As we will see, the strategy adopted in this work is different since it relies on basis functions which are encoded in kernels.

The approaches proposed in [11] and [2] are in some sense complimentary. The first one uses the kernel to handle efficiently a large number of monomials but regularization is not always so effective.The second one refines this aspect but the price to pay is that implicit encoding of monomials is absent. In this work, we propose new techniques that overcome such dichotomy. To obtain this, two new kernels are introduced.

The first new kernel, which we call *Multiplicative Polinomial Kernel* (MPK), builds upon the polynomial kernel. For Volterra series of order r, it consists of the product of r basic building blocks. Each block consists of a linear kernel containing hyperparameters that permit monomial selection. This is a fundamental novelty w.r.t. the polynomial kernel that does not promote any sparsity, possibly returning solutions too rich of monomials. Hence, we will see that MPK allows a finer regularization, discovering those system parts that really influence the output and improving prediction capability.

Then, inspired by [2], we will show that MPK features can be further improved. Similarly to MPK, the second new kernel is the product of r linear kernels but with different hyperparameters able to model smooth exponential decay of Volterra coefficients. For this reason, it is called *Smooth Exponentially Decaying MPK* (SED-MPK). Hence, SED-MPK combines the nice features of [2] with implicit kernel encoding: handling of high-order Volterra series is so made possible.

Preliminary results about the MPK are reported in [20], where we showed the advantages of the MPK w.r.t. the standard inhomogeneous polynomial kernel through numerical examples. In this work, compared to [20], we analyze rigorously the MPK regularization properties, comparing its performance also with the homogeneous polynomial kernel, as well as introducing the SED-MPK.

The paper is organized as follows. In Section 2 we provide background notions about discrete-time Volterra series, and we describe some solutions proposed to identify such models. We start Section 3 analyzing the regularization properties of standard polynomial kernel, highlighting possible limitations. Then, we introduce the MPK, discussing the advantages due to the MPK parametrization. Finally, we discuss the recovering of the coefficient of Volterra maps from the kernel-based estimate of the model. In Section 4, we describe the SED-MPK, providing insights about the regularization induced by the SED-MPK parameters w.r.t. the coefficients of the Volterra map. Finally, in Section 5 we report experimental results.

2 Background

2.1 Volterra series

Consider a discrete time system, and let z_t be its output at time t. Assume that the system has finite memory m, and let $\boldsymbol{u}_t = [u_t, \dots, u_{t-m}]^T$ be the column vector containing the lagged inputs influencing the system response at time k. When modeling the system response with a truncated Volterra series of order r, the noisy output y_t is defined by the sum of $r+1$ Volterra functions and of the measurement noise $e_t \sim N\left(0, \sigma_n^2\right)$. Specifically, one has

$$y_t = z_t + e_t = h_0 + \sum_{i=1}^{r} H_i(\boldsymbol{u}_t) + e_t \ , \tag{1}$$

where h_0 represents the zero-order Volterra function, constant and independent of the inputs, while $H_i(\boldsymbol{u}_t)$ are the higher-order contributions. Each $H_i(\boldsymbol{u}_t)$ is the convolution between a Volterra map h_i and all the possible monomials of degree i built with the components of $\boldsymbol{u}_t$. The Volterra map h_i is function of i variables $\{\tau_j\}_{j=1}^{i}$ that may assume values $\{0, 1, \dots, m\}$ and represent input lags. In other words, they select (possibly repeated) components from $\boldsymbol{u}_t$ to define a monomial. Then, the expression of $H_i(\boldsymbol{u}_t)$ is

$$H_i\left(\boldsymbol{u}_t\right) = \sum_{\tau_1=0}^{m} \cdots \sum_{\tau_i=0}^{m} h_i\left(\tau_1, \dots, \tau_i\right) \prod_{\tau=\tau_1}^{\tau_i} u_{k-\tau} \ . \tag{2}$$

For example, let $m=2$ so that $\boldsymbol{u}_t = [u_t \ u_{t-1} \ u_{t-2}]$, with $i=4$ and $\tau_1 = 0, \tau_2 = 0, \tau_3 = 1, \tau_4 = 2$. Note that τ_1 and τ_2 both select u_t while τ_3 and τ_4 choose, respectively, u_{k-1} and u_{k-2}. Then $h_4\left(0,0,1,2\right)$ is a coefficient that multiplies the monomial $u_t^2 u_{t-1} u_{t-2}$.

Commonly, Volterra maps are assumed to be symmetric with respect to the input lags. Given a set of i input lags, the h_i value is equal for all the possible permutation of the lags. For instance, coming back to the previous example, under symmetry assumptions one has $h_4\left(0,0,1,2\right) = h_4\left(0,1,0,2\right) = h_4\left(1,0,2,0\right) = \dots$.

Alternatively, (2) can be rewritten more compactly with an inner product. Let $\boldsymbol{\phi}_i(\boldsymbol{u}_t) \in \mathbb{R}^{n_i}$ be the vector collecting all the distinct monomials in $\boldsymbol{u}_t$ with degree i, with $n_i = \binom{m+i}{i}$. Moreover, let $\boldsymbol{w}_i$ be the vector function of the coefficients h_i, multiplied by opportune constants to account for the repetitions due to symmetry. Specifically, consider the generic monomial $\prod_{j=0}^{m} u_{k-j}^{d_j}$, where $\boldsymbol{d} = [d_0, \dots, d_m]$ are the relative degrees of each variable, and $\sum_{j=0}^{m} d_j = i$. Then, the multiplicative constant cited above is equal to the multinomial coefficient $\binom{i}{d_0, \dots, d_m}$, hereafter denoted just by $\binom{i}{\boldsymbol{d}}$. For an opportune permutation of the h_i and $\boldsymbol{w}_i$ elements, one then has

$$H_i\left(\boldsymbol{u}_t\right) = \boldsymbol{\phi}_i^T(\boldsymbol{u}_t)\boldsymbol{w}_i. \tag{3}$$

2.2 Volterra maps identification via regularized least squares

Combining (1) and (3), we have

$$y_t = \boldsymbol{\phi}^T(\boldsymbol{u}_t)\boldsymbol{w} + e_t,$$

with

$$\boldsymbol{\phi}^T(\boldsymbol{u}_t) = \left[1, \boldsymbol{\phi}_1^T(\boldsymbol{u}_t), \dots \boldsymbol{\phi}_r^T(\boldsymbol{u}_t)\right] \in \mathbb{R}^n, \tag{4a}$$

$$\boldsymbol{w}^T = \left[h_0, \boldsymbol{w}_1^T, \dots, \boldsymbol{w}_r^T\right] \in \mathbb{R}^n, \tag{4b}$$

where $n = 1 + \sum_{i=1}^{r} n_i$ with n_i equal to the dimension of $\boldsymbol{w}_i$. System identification then reduces to obtaining an estimate $\hat{\boldsymbol{w}}$ of $\boldsymbol{w}$ from the data set

$$D = \{(\boldsymbol{u}_t, y_t), \, t = 1, \dots, N\} \, . \tag{5}$$

For instance, one can use least squares:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \left\|\boldsymbol{y} - \Phi^T \boldsymbol{w}\right\|^2,$$

where $\boldsymbol{y} = [y_1, \dots y_N]^T$ and the regression matrix Φ^T is

$$\Phi^T = \left[\boldsymbol{\phi}(\boldsymbol{u}_1) \ \dots \ \boldsymbol{\phi}(\boldsymbol{u}_N).\right]^T.$$

This approach suffers of the curse of dimensionality. The number of parameters n grows quickly with r and m, entailing high variance in the estimate. In [2] such problem has been addressed for Volterra series of order two exploiting regularized least squares. In particular, the smooth exponential decay of the Volterra coefficients is enforced by a suitable positive definite matrix (also called kernel) denoted by P. The estimator becomes

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \left\|\boldsymbol{y} - \Phi^T \boldsymbol{w}\right\|^2 + \boldsymbol{w}^T P^{-1} \boldsymbol{w}, \tag{6}$$

with $\hat{\boldsymbol{w}}$ available in closed form as

$$\hat{\boldsymbol{w}} = \left(\Phi\Phi^T + P^{-1}\right)^{-1} \Phi \boldsymbol{y}. \tag{7}$$

The (i,j)-entry of the matrix P can be seen as a similarity measure between the i-th and the j-th component of the vector $\boldsymbol{w}$ in (4), i.e., between two coefficients associated to two monomials. In [2], such matrix is block diagonal, i.e., $P = \text{block diag}([P_0, P_1, P_2])$. Each block accounts for the coefficients related to the monomials of a distinct degree. First, P_0 is a positive scalar that regularizes h_0 in (4). Second, $P_1 \in \mathbb{R}^{n_1 \times n_1}$ accounts for $\boldsymbol{w}_1$, i.e., the coefficient of the Volterra map h_1. The entries of P_1 have been defined exploiting the DC kernel proposed in [27]. Assume that the $\boldsymbol{w}_1$ elements are listed in increasing order of time-lags, i.e., $\boldsymbol{w}_1 = [h_1(0), \dots, h_1(m)]^T$. Then, $P_1(i,j)$, which expresses the similarity between $h_1(i)$ and $h_1(j)$, is defined as

$$P_1(i,j) = c \cdot e^{-\alpha|i-j|} e^{-\beta \frac{|i+j|}{2}},$$

where c is a scale factor, while α and β regulate the h_1 exponential decay. Note that $P_1(i,j)$ is the product of two exponentials. The first one measures the similarity between coefficients, weighting the distance between time-lags: coefficients related to lags that are close are more similar that coefficients related to lags that are far. The second one decreases with the magnitude of the time-lags. The important contribution of [2] is the definition of $P_2 \in \mathbb{R}^{n_2 \times n_2}$, that describes the interactions between the $\boldsymbol{w}_2$ elements, i.e., the coefficients of the Volterra map h_2. The matrix P_2 has been defined extending the regularization strategy described for P_1 to the two dimensional case, to deal with the fact h_2 depends on two time-lags. For a more detailed description of the P_2 definition we refer the reader to [2].

Numerical results reported in [2] show that the addition of the regularization term is crucial to control estimator's variance. Regarding computational issues, (7) shows that the number of operations scales with the cube of n (the number of distinct monomials), while the storage requirements are proportional to the square of n. Thus, there is a direct dependence on the number of coefficients of the Volterra maps. Unfortunately, this number grows rapidly with the system memory and the order of the Volterra series. Despite the valuable implementation adjustments proposed in [4], these requirements makes already hard to introduce monomials of degree three.

2.3 Polynomial kernel and Volterra series

An alternative technique is regularized system identification in a RKHS defined by a kernel function $K(\boldsymbol{u}_t, \boldsymbol{u}_j)$. Under mild assumptions, a kernel function admits a (possibly infinite) expansion in terms of basis functions ψ_q, i.e.

$$K(\boldsymbol{u}_t, \boldsymbol{u}_j) = \sum_q \lambda_q \psi_q(\boldsymbol{u}_t) \psi_q(\boldsymbol{u}_j), \tag{8}$$

where λ_q are positive scalars [8]. Any function in the induced RKHS has then the representation

$$f(\boldsymbol{u}_t) = \sum_q c_q \psi_q(\boldsymbol{u}_t), \tag{9}$$

for suitable coefficients c_q.
A widely used estimator of the input-output map is

$$\hat{f} = \arg\min_f \sum_{t=1}^{N} (y_t - f(\boldsymbol{u}_t))^2 + \gamma \|f\|_H^2, \tag{10}$$

where γ is the regularization parameter that trades-off data fit and the penalty term $\|\cdot\|_H^2$, given by the squared RKHS norm. According to the *representer theorem* [31], one has

$$\hat{f}(\boldsymbol{u}_t) = \sum_{j=1}^{N} \alpha_t K(\boldsymbol{u}_t, \boldsymbol{u}_j), \tag{11}$$

where $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]^T$ is given by

$$\boldsymbol{\alpha} = (\mathbb{K} + \gamma I_N)^{-1} \boldsymbol{y}, \tag{12}$$

with $\mathbb{K}$ the so called kernel matrix whose (t,j) entry is $K(\boldsymbol{u}_t, \boldsymbol{u}_j)$. So, even if the RKHS can implicitly encode a very large (possibly infinite) number of basis functions ψ_q, the estimate always belongs to a finite-dimensional subspace and can be computed with a finite number of operations.

A class of kernel-based solutions for Volterra series identification has been proposed in [11]. The authors rely on the use of polynomial kernels, which are strongly connected with Volterra series since they encode monomials in $\boldsymbol{u}_t$. Let us consider

$$\sum_{i=1}^{r} \rho_i \left(\boldsymbol{u}_t^T \boldsymbol{u}_j\right)^i + \rho_0, \tag{13}$$

where the $\rho_i \geq 0$ are tunable hyperparameters. As discussed in [32], each term $\left(\boldsymbol{u}_t^T \boldsymbol{u}_j\right)^i$ is the homogeneous polynomial kernel of degree i, which encodes all the monomials of degree i. So the ψ_q in (8) are the elements of $\boldsymbol{\phi}_i$ in (4a).

The computational bottleneck of kernel-based methods is the matrix inversion in (12), which requires $O(N^3)$ operations. The memory requirements to store the kernel matrix are $O(N^2)$. However, notice that there is no direct dependence on the number of basis functions encoded in the kernel. So, computational complexity does not depend on the number of Volterra parameters, and this allows to handle also high-order series.

To summarize, kernel-based identification of all the proposed models relies on the following steps:

- kernel selection;
- hyperparameters estimation by maximizing the Marginal Likelihood of the training data, see [29];
- computation of the estime (11);

- if possible, estimation of the Volterra maps coefficients, see Section 3.5.

It is worth mentioning that the last step is not mandatory, since predictions can be done with (11). However, in some cases it might be interesting computing also the estimate of the Volterra maps coefficients, to analyze the properties of the Volterra maps, see for instance [18].

3 Multiplicative Polynomial Kernel

Now, we show that in important cases the kernel in (13) do not provide enough flexibility to penalize the different monomials. For our purposes, it is useful also to recall the following two facts. First, given the kernel expansion (8) in terms of eigenvalues λ_q and independent basis functions ψ_q, one has

$$f(\boldsymbol{u}_t) = \sum_q c_q \psi_q(\boldsymbol{u}_t) \implies \|f\|_H^2 = \sum_q \frac{c_q^2}{\lambda_q}. \tag{14}$$

So, smaller λ_q give more penalty to the coefficients of the corresponding ψ_q. Second, as said, in the context of our polynomial kernels, the ψ_q belong to the set of monomials contained in

$$\boldsymbol{\phi}^T(\boldsymbol{u}_t) = \left[1, \boldsymbol{\phi}_1^T(\boldsymbol{u}_t), \dots \boldsymbol{\phi}_r^T(\boldsymbol{u}_t)\right].$$

It is also useful now to denote the components of the blocks $\boldsymbol{\phi}_i$, i.e., the monomials of degree i, by $\phi_1^{(i)}, \phi_2^{(i)}, \dots$. So, one has

$$\boldsymbol{\phi}_i(\boldsymbol{u}_t) = \left[\phi_1^{(i)}(\boldsymbol{u}_t) \ \dots \ \phi_{n_i}^{(i)}(\boldsymbol{u}_t)\right]^T. \tag{15}$$

From the discussion before (3), we recall also that each monomial of degree i is in one-to-one correspondence with a $m+1$-dimensional vector whose components are the relative degrees of each component of $\boldsymbol{u}_t$. In particular, we will use $\boldsymbol{d}_q^{(i)}$ to denote the vector that contains the relative degrees of the monomial $\phi_q^{(i)}$. We will also indicate with $\mathcal{D}_i$ the following set

$$\mathcal{D}_i = \left\{ \boldsymbol{d}^{(i)} = [d_0^{(i)} \ d_1^{(i)} \ \dots \ d_m^{(i)}] \text{ s.t. } \sum_{\tau=0}^{m} d_\tau^{(i)} = i \right\}, \tag{16}$$

that is thus associated with all the monomials in $\boldsymbol{\phi}_i$ (15).

3.1 Penalties induced by the polynomial kernels

We focus on the penalties induced by the single homogenous kernels $\left(\boldsymbol{u}_t^T \boldsymbol{u}_j\right)^i$ that compose (13). To consider a more general case, we also introduce the positive semidefinite matrix $\Sigma^{(i)} \in \mathbb{R}^{(m+1)\times(m+1)}$, and we define our generalized building block as

$$k_{pk}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \left(\boldsymbol{u}_t^T \Sigma^{(i)} \boldsymbol{u}_j\right)^i, \tag{17a}$$

$$\Sigma^{(i)} = \text{diag}(\boldsymbol{\sigma}^{(i)}), \tag{17b}$$

$$\boldsymbol{\sigma}^{(i)} = [\sigma_0^{(i)}, \dots, \sigma_m^{(i)}]. \tag{17c}$$

The overall kernel turns out to be

$$K_{pk}^{(r)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \sum_{i=1}^{r} \rho_i k_{pk}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) + \rho_0, \tag{18}$$

namely, (18) corresponds to (13) when considering all the $\Sigma^{(i)}$ equal to the identity. Since $k_{pk}^{(i)}$ encodes all the monomials of degree i (if the $\Sigma^{(i)}$ is full-rank), using (15) one must have

$$k_{pk}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \sum_{q=1}^{n_i} \lambda_q^{(i)} \phi_q^{(i)}(\boldsymbol{u}_t) \phi_q^{(i)}(\boldsymbol{u}_j). \tag{19}$$

But we can also find another representation through the multinomial theorem. Using (16) and (17), one has

$$\begin{aligned} k_{pk}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) &= \left(\sum_{\tau=0}^{m} \sigma_\tau^{(i)} u_{t-\tau} u_{j-\tau}\right)^i \\ &= \sum_{\boldsymbol{d}^{(i)} \in \mathcal{D}_i} \binom{i}{\boldsymbol{d}^{(i)}} \prod_{\tau=0}^{m} \left(\sigma_\tau^{(i)} u_{t-\tau} u_{j-\tau}\right)^{d_\tau^{(i)}}, \end{aligned} \tag{20}$$

where here, and in what follows, we have used the fact that $\binom{i}{\boldsymbol{d}^{(i)}}$ is the number of all the different lags vectors that identify the same monomial.
If $\boldsymbol{d}_q^{(i)}$ contains the relative degrees $[d_{q,0}^{(i)} \ d_{q,1}^{(i)} \ \dots \ d_{q,m}^{(i)}]$ of the monomial $\phi_q^{(i)}$, equating the right hand sides of (19) and (20) we have

$$\lambda_q^{(i)} = \binom{i}{\boldsymbol{d}_q^{(i)}} \prod_{\tau=0}^{m} (\sigma_\tau^{(i)})^{d_{q,\tau}^{(i)}}. \tag{21}$$

The expression (21) highlights two interesting issue. First, a part of the penalty is assigned only on the basis of the relative degrees associated to the monomial. In particular, due to the behaviour of the multinomial coefficient, monomials composed by mixed terms are promoted. The gap between the penalties assigned to monomials with mixed terms and monomials composed by just one term becomes also particularly relevant when i grows. The reconstruction of the input-output function based on this kind of penalties might not be suitable. Second, notice that acting on the values of the $\sigma_\tau^{(i)}$, we can promote or penalize the monomials containing $u_{t-\tau}$. However, from (21), we can see that promotions or penalizations are rigid, in the sense that by increasing $\sigma_\tau^{(i)}$ we promote simultaneously all the monomials of degree i in which $u_{t-\tau}$ appears. There might be cases in which more flexibility is needed to penalize $u_{t-\tau}$ by also accounting for its relative degree.

To clarify this concept, we discuss a simple example. Consider the following polynomial function in $\boldsymbol{u}_t = [u_t \, u_{t-1}]$, defined as

$$f(\boldsymbol{u}_t) = u_t^2 u_{k-1} + u_t^3. \tag{22a}$$

Moreover, we define $\boldsymbol{\phi}_3(\boldsymbol{u}_t)$ ordering the monomials as follows,

$$\boldsymbol{\phi}_3(\boldsymbol{u}_t) = \left[u_t^3 \;\; u_t^2 u_{t-1} \;\; u_t u_{t-1}^2 \;\; u_{t-1}^3\right]. \tag{22b}$$

Ideally, given that $u_t u_{t-1}^2$ and u_{t-1}^3 do not compare in $f(\boldsymbol{u}_t)$, and the u_t^3 and $u_t^2 u_{t-1}$ coefficients are equal, we would like to have $\lambda_3^{(3)}$ and $\lambda_4^{(3)}$ equal to zero, and $\lambda_1^{(3)}$ and $\lambda_2^{(3)}$ assuming similar values higher than zero. However, from (21), this is not possible, given that if $\lambda_3^{(3)}$ and $\lambda_4^{(3)}$ are null also $\lambda_2^{(3)}$ is null.

In (21), the presence of the multinomial coefficient $\binom{i}{\boldsymbol{d}^{(i)}}$ is due to repetitions induced by symmetry. To avoid this fact, and for visualization purposes, we express the penalties induced by $\lambda_q^{(i)}$ in the coefficients of the i-th Volterra map that define the input-output formulation (2). Note that the set of ordered lags $\boldsymbol{\tau} = (\tau_1, \dots, \tau_i)$ identifies univocally i, q and, hence, the monomial $\phi_q^{(i)}$ and the vector $\boldsymbol{d}_q^{(i)}$ with its relative degrees. The opposite is not true due to the possible change of ordering: all the permutations of $(\tau_1, \dots, \tau_i)$ lead to the same monomial. Due to the symmetry of the Volterra maps, the $\lambda_q^{(i)}$ contribution is equally divided among all the coefficients of h_i associated to the $\binom{i}{\boldsymbol{d}_q^{(i)}}$ permutations of $(\tau_1, \dots, \tau_i)$. Then, $\lambda_{h_i(\boldsymbol{\tau})}$, the penalty assigned to the coefficient $h_i(\tau_1, \dots, \tau_i) = h_i(\boldsymbol{\tau})$, is inversely proportional to

$$\lambda_{h_i(\boldsymbol{\tau})} = \frac{\lambda_q^{(i)}}{\binom{i}{\boldsymbol{d}_q^{(i)}}}. \tag{23}$$

Hence, combining the last equation with (21), we have that the $\lambda_{h_i(\boldsymbol{\tau})}$ defined by $k_{pk}^{(i)}$ are

$$\lambda_{h_i(\boldsymbol{\tau})} = \prod_{\tau=0}^{m} (\sigma_\tau^{(i)})^{d_{q,\tau}^{(i)}}.$$

The last equality permits to analyze the effects of the $\sigma_\tau^{(i)}$ in the penalties assigned to the Volterra maps. Notice that when considering $\sigma_\tau^{(i)} = 1 \; \forall \tau = 0 \dots, m$, as done in [11], the penalties assigned to the coefficients of the h_i maps are flat, in the sense that all the coefficients are equally penalized.

Remark 1 *An alternative to* (18) *relies on the use of the inhomogeneous polynomial kernel of order r:*

$$K_{ipk}^{(r)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \left(1 + \boldsymbol{u}_t^T \Sigma \boldsymbol{u}_j\right)^r, \tag{24}$$

where $\Sigma \in \mathbb{R}^{(m+1)\times(m+1)}$ is typically diagonal. The inhomogeneous polynomial kernel encodes all the monomials with degree up to r [32]. Similar considerations can be done for this model by recalling that (24) *can be expressed as sum of r homogeneous kernels sharing the same $\Sigma = \Sigma^{(i)}$, multiplied by opportune coefficients, namely,*

$$\left(1 + \boldsymbol{u}_t^T \Sigma \boldsymbol{u}_j\right)^r = \sum_{i=0}^{r} \binom{r}{i} k_{pk}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j).$$

The last equation shows that the use of the inhomogeneous polynomial kernel further reduces the flexibility in determining penalties. In fact, the same hyperparameters determine simultaneously the penalties assigned to monomials with different degrees while, through the model (18), *one can exploit $\boldsymbol{\sigma}^{(i)}$ to tune specific regularization for each block $\boldsymbol{\phi}_i$ in* (15).

3.2 Multiplicative Polynomial Kernel

In view of the limitations of the currently used polynomial kernels described above, the MPK is now introduced. Let $k_{mpk}^{(i)}$ denote a novel kernel that, similarly to the homogeneous polynomial kernel, encodes all the monomials of degree i, i.e., all those contained in $\boldsymbol{\phi}_i$. But its structure is different, being the product of i linear kernels, i.e.,

$$k_{mpk}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \prod_{p=1}^{i} \left(\boldsymbol{u}_t^T \Sigma_p \boldsymbol{u}_j\right), \tag{25a}$$

$$\Sigma_p = \mathrm{diag}(\boldsymbol{\sigma}_p), \tag{25b}$$

$$\boldsymbol{\sigma}_p = [\sigma_0^{(p)}, \dots, \sigma_m^{(p)}]. \tag{25c}$$

Differently from $k_{pk}^{(i)}$ in (17), the $k_{mpk}^{(i)}$ assigns a distinct set of hyperparameters to each factor. It is easy to see that (25) is a well-defined kernel function, being the product of valid kernels [29]. Also, it admits an expansion in terms of the elements in $\boldsymbol{\phi}_i$. Then, we propose the following kernel to model Volterra series of order r:

$$K_{mpk}^{(r)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \sum_{i=1}^{r} \rho_i k_{mpk}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) + \rho_0. \tag{26}$$

3.3 Penalties induced by the MPK

As done with (17), we analyze the regularization properties of (26) focusing on the single bulding blocks $k^{(i)}_{mpk}$. Using the same kind of notation adopted in the previous subsection, an expression of the $\lambda^{(i)}_q$ is obtained by expanding (25), isolating the terms with relative degrees $\boldsymbol{d}^{(i)}_q$, and, finally, summing their coefficients.
To perform such computation, let us introduce the following set

$$\mathcal{T}^{(i)}_q = \left\{ (\tau_1, \dots, \tau_i) \text{ s.t. } \prod_{\tau=\tau_1}^{\tau_i} u_{k-\tau} = \prod_{\tau=0}^{m} u_{k-\tau}^{d^{(i)}_{q,\tau}} \right\}, \tag{27}$$

Each vector in $\mathcal{T}^{(i)}_q$ contains i lags associated with the same $\boldsymbol{d}^{(i)}_q$. Then, summing all the contributions associated to the elements of $\mathcal{T}^{(i)}_q$, one obtains

$$\lambda^{(i)}_q = \sum_{(\tau_1,\dots,\tau_i)\in\mathcal{T}^i_q} \left(\prod_{p=1}^{i} \sigma^{(p)}_{\tau_p} \right). \tag{28}$$

The last equation shows that the MPK parameters allows for a more flexible regularization. In particular, tuning opportunely the hyperparameters, it is possible to promote or penalize the relative degree with which each lagged input appears. For example, consider the lagged input $u_{k-\tau}$, and the coefficients $\sigma^{(1)}_\tau, \dots, \sigma^{(i)}_\tau$. Then, by inspection of (28) it comes that, in order to promote monomials in which $u_{k-\tau}$ appear with relative degree $d \leq i$, at least d of the $\sigma^{(1)}_\tau, \dots, \sigma^{(i)}_\tau$ hyperparameters should assume large values. Otherwise, all the products in (28) and λ_q become small, much penalizing the monomials containing the powers $u^d_{k-\tau}, \dots, u^i_{k-\tau}$. Standard polynomial kernels do not provide such possibility, given that, as showed before, changing the values of the hyperparameters promote, or penalize, simultaneously all the monomials containing $u_{k-\tau}$, regardless of the relative degree.

To clarify this point, we consider again the simple example of Section (3.1), i.e., the third degree polynomial function in $\boldsymbol{u}_t = [u_t \, u_{t-1}]$ reported in (22a), with $\boldsymbol{\phi}_3$ defined in (22b). Let the MPK hyperparameters assuming the following values,

$$\boldsymbol{\sigma}_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}, \boldsymbol{\sigma}_2 = \begin{bmatrix} 1 & 0 \end{bmatrix}, \boldsymbol{\sigma}_3 = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

Hence, from (28) we obtain $\lambda^{(3)}_1 = 1$, $\lambda^{(3)}_2 = 1$, $\lambda^{(3)}_3 = 0$ and $\lambda^{(3)}_4 = 0$. Recalling that λ_q is inversely proportional to the penalty assigned to the q-th element of $\boldsymbol{\phi}_3$, we appreciate that with the set of hyperparameters proposed the MPK highly penalizes the monomials in which u_{t-1} appears with degree greater than one, in accordance with the function in (22a).

It is worth stressing that MPK obtains more flexibility by increasing the number of hyperparameters, possibly leading to overfitting. In particular, when considering the polynomial kernel of degree i, MPK has $(m+1)i$ additional hyparameters. However, as it will be seen through extensive numerical experiments, when considering sufficiently exciting input trajectories, the prediction capability of MPK outperforms constantly that of the homogeneus and inhomogeneus polynomial kernel.

3.4 Re-parametrization of the diagonal Σ_p matrices

To optimize the Σ_p matrices, we will use Marginal Likelihood maximization [29], by adopting a reparametrization. In fact, since (25) is the product of i equal blocks, optimizing directly the $\sigma^{(p)}_\tau$ could lead to undesired behaviors due to the presence of several local maxima. In particular, from the structure (25), it is immediate to see that any permutation of the i vectors $\boldsymbol{\sigma}_p$ that represent the diagonals of the matrices Σ_p define the same $k^{(i)}_{mpk}$. To avoid this situation, we propose a hierarchical parametrization of the $\boldsymbol{\sigma}_p$ vectors, w.r.t. a set of vectors $\{\boldsymbol{a}_p, p = 1, \dots, i\}$, where $\boldsymbol{a}_p = \left[a^{(p)}_0, \dots, a^{(p)}_m\right]$ with all positive entries. More specifically, the $\boldsymbol{\sigma}_p$ vectors are defined iteratively by a backward iteration as follows,

$$\boldsymbol{\sigma}_i = \boldsymbol{a}_i, \tag{29a}$$
$$\boldsymbol{\sigma}_p = \boldsymbol{\sigma}_{p+1} + \boldsymbol{a}_p, \tag{29b}$$

with $p = i-1, \dots, 1$.

Beyond reducing the presence of possible local maxima, this parametrization has an intuitive interpretation also in terms of the penalties and the relative degree of each $u_{t-\tau}$. Indeed, from (29) we see that increasing $a^{(p)}_\tau$ means increasing all the $\sigma^{(d)}_\tau$ with $d \leq p$, and then, recalling (28), this promotes all the monomials in which $u_{k-\tau}$ has degree at least equal to p. Moreover, if $a^{(d)}_\tau$ is close to zero for $d > p$, then the monomials with $u_{t-\tau}$ of degree larger than p are highly penalized. In this way, by tuning opportunely the hyperparameters it is possible to control the maximum degree of each lagged input.

3.5 Computation of the Volterra maps coefficients

Now, we discuss how to estimate the coefficients of the Volterra maps starting from the kernel-based estimate of the model. More precisely, we provide a closed form expression of the estimate $\hat{\boldsymbol{w}}$ of the vector $\boldsymbol{w}$ defined in (4). Recall that the elements of $\boldsymbol{w}$ are in one-to-one correspondence with the monomials in $\boldsymbol{\phi}(\boldsymbol{u}_t)$. So, given

$\hat{\boldsymbol{w}}$, the estimate of $h_i(\tau_1 \dots \tau_i)$ can be recovered dividing by $\binom{i}{\boldsymbol{d}}$ the component of $\hat{\boldsymbol{w}}$ related to the monomial $\prod_{\tau=\tau_1}^{\tau_i} u_{t-\tau}$. For our purposes, we need to assume that the kernel expansion is available in closed form, with its basis functions corresponding to the monomials of the Volterra model. Namely, referring to (14), the number of $\psi_q(\boldsymbol{u}_t)$ is n and they correspond to the elements of $\boldsymbol{\phi}(\boldsymbol{u}_t)$ defined in (4a). In matrix form we have $k(\boldsymbol{u}_t, \boldsymbol{u}_j) = \boldsymbol{\phi}^T(\boldsymbol{u}_t)\Lambda\boldsymbol{\phi}^T(\boldsymbol{u}_j)$, where Λ is a diagonal matrix, with the diagonal elements correspondent to the λ_q coefficients ordered in accordance with the monomials in $\boldsymbol{\phi}$. As previously discussed, these conditions are met with all the polynomial kernels so far introduced, i.e., $K_{ipk}^{(r)}$, $K_{pk}^{(r)}$, and $K_{mpk}^{(r)}$, defined, respectively, in (18), (24), and (25). For convenience, we recall that $\boldsymbol{\alpha}$ is the vector that defines the kernel-based estimator, see (11), while Φ is the matrix staking the vectors $\boldsymbol{\phi}(\boldsymbol{u}_t)$ with $t = 1 \dots N$. Then, the following proposition holds true.

Proposition 1 *Let K be a kernel whose expansion corresponds to the elements of $\boldsymbol{\phi}(\boldsymbol{u}_t)$, and denote by λ_q the coefficient of the expansion related to the* q*-th element of $\boldsymbol{\phi}(\boldsymbol{u}_t)$. Then, the estimate $\hat{\boldsymbol{w}}$ of the Volterra model coefficients defined in* (3)*, is*

$$\hat{\boldsymbol{w}} = \Lambda\Phi\boldsymbol{\alpha}. \tag{30}$$

The proof of the proposition is reported in the Appendix. Notice that in our setting Proposition 1 can be used when using both K_{pk} and K_{mpk} using the expressions of λ_q reported in (21) and (28), respectively.

4 Smooth exponentially decaying MPK

In this section, we extend the framework introduced in the previous subsection by designing the Smooth Exponentially Decaying MPK (SED-MPK). The model incorporates information on smooth exponential decay of Volterra maps coefficients. The kernel has the same structure of the MPK (25), except for the positive definite matrices Σ_p which are no more restricted to be diagonal. In fact, we define the basic building kernel as

$$k_{sed}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \prod_{p=1}^{i} \boldsymbol{u}_t^T \Sigma_p \boldsymbol{u}_j \tag{31a}$$

$$\Sigma_p(i,j) = e^{-\alpha_p|i-j|}e^{-\beta_p|i-1+j-1|} \tag{31b}$$

where $\Sigma_p(i,j)$ is the (i,j)-entry of Σ_p, and the hyperparameters α_p and β_p regulate the smooth exponential decaying behavior. One can thus notice that while in (25) each Σ_p is diagonal with the hyperparameters corresponding to the $m+1$ diagonal elements, now the matrix is full but depends only on α_p and β_p. Similarly to the MPK, to limit the presence of local maxima, the α_p and β_p hyperparameters can be parametrized hierarchically. When considering a Volterra series or order r, the mode based on the SED-MPK is the sum of kernels (31) of increasing order, i.e.,

$$K_{sed}^{(r)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \sum_{i=1}^{r} \rho_i k_{sed}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) + \rho_0. \tag{32}$$

The regularization matrix in (31b) has clear analogies with the regularizer introduced in [2]. But a fundamental difference holds: while in [2] it is used to regularize the Volterra coefficients h_i in an explicit way, i.e. writing down the model in terms of basis functions, the penalty is here embedded in a kernel. This point is crucial, and the rest of the subsection is devoted to illustrate it by building a kernel that generalizes both MPK and SED-MPK.

As said, once an order i is fixed, the $h_i(\boldsymbol{\tau})$ is the coefficient of a monomial of order i defined by the vector $\boldsymbol{\tau} = [\tau_1, \dots, \tau_i]$ containing the input lags. Recall that $H_i(\boldsymbol{u}_t)$ in (2) is the sum of products between h_i and the possible monomials of degree i built with $\boldsymbol{u}_t$. In other words, this is the part of the system output due only to the monomials of order i. With this in mind, let $\mathcal{P}_i(\boldsymbol{\tau}, \boldsymbol{\tau}')$ be a kernel that measures the similarity between two lags vectors. Then, we define

$$k^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \sum_{\boldsymbol{\tau}} \sum_{\boldsymbol{\tau}'} \mathcal{P}_i(\boldsymbol{\tau}, \boldsymbol{\tau}') \prod_{p=1}^{i} u_{t-\tau_p} u_{j-\tau'_p} \tag{33}$$

as the similarity measure between the system outputs $H_i(\boldsymbol{u}_t)$ and $H_i(\boldsymbol{u}_j)$ obtained using two different inputs $\boldsymbol{u}_t$ and $\boldsymbol{u}_j$. Note that (33) accounts for the effect of all the monomials of order i built with $\boldsymbol{u}_t$ and $\boldsymbol{u}_j$ by summing over all the possible couples of vector lags. Now, consider (31a) but without assigning any particular structure to Σ_p. In particular, for t, j in $\{0, 1, \dots, m\}$ let $\sigma_{tj}^{(p)}$ be the $(t+1, j+1)$-entry of Σ_p. Then, it is easy to prove that (31a) generalizes into

$$\sum_{\boldsymbol{\tau}} \sum_{\boldsymbol{\tau}'} \prod_{p=1}^{i} \sigma_{\tau_p \tau'_p}^{(p)} u_{t-\tau_p} u_{j-\tau'_p}, \tag{34}$$

where we are summing over all the possible couples of lags vectors

$$\boldsymbol{\tau} = [\tau_1, \dots, \tau_i], \quad \boldsymbol{\tau}' = [\tau'_1, \dots, \tau'_i].$$

Equating the right hand sides of (34) and (33), we obtain

$$\mathcal{P}_i(\boldsymbol{\tau}, \boldsymbol{\tau}') = \prod_{p=1}^{i} \sigma_{\tau_p \tau'_p}^{(p)}. \tag{35}$$

The equality (35) is important because it shows the correspondence between the similarity measure assigned to two lags vectors (and, hence, also to two Volterra map coefficients) and the elements of the Σ_p. In particular, (35) says that the similarity between $h_i(\boldsymbol{\tau})$ and $h_i(\boldsymbol{\tau}')$ is factorized by i elements, corresponding to the entries of the Σ_p matrices identified by $\boldsymbol{\tau}$ and $\boldsymbol{\tau}'$. Without parametrizing the Σ_p structure, the model complexity could become relevant, in particular when the system memory m and the Volterra order i are large. The MPK reduces complexity by using diagonal matrices exploiting (25b). Instead, inspired by [26] and [2], the SED-MPK uses the option (31b) to limit the number of parameters and enforce smooth exponential decay of the $h_i(\boldsymbol{\tau})$. This feature is so encoded in the kernel, providing information that monomials built with large input lags values should have less influence on output prediction. To clarify this concept, we provide the explicit expression of the $\mathcal{P}(\boldsymbol{\tau}, \boldsymbol{\tau}')$ obtained with the kernel in (31), considering the particular case where all the α_p (resp. β_p) are equal to α (resp. β). By (34) we have

$$\mathcal{P}(\boldsymbol{\tau}, \boldsymbol{\tau}') = e^{-\alpha\|\boldsymbol{\tau}-\boldsymbol{\tau}'\|_1} e^{-\beta\|\boldsymbol{\tau}+\boldsymbol{\tau}'\|_1}. \tag{36}$$

The last expression is the product of two exponentials, where the first term expresses the smoothness of $\mathcal{P}(\boldsymbol{\tau}, \boldsymbol{\tau}')$, while the second defines the decay w.r.t. the increasing of the lags.

The fundamental point is that, thanks to our kernel-based framework, the SED-MPK computational complexity scales with the number of samples, instead that with the number of coefficients of the Volterra maps as in [2]. This allows to model also high-order Volterra series. As proven by numerical results reported in the next section, solutions based on SED-MPK are effective in identifying also Volterra series up to order 5.

5 Experimental results

We tested the proposed kernel functions in several experiments. First, we compare the performance of the MPK-based estimator (26) with that of the estimators based on IPK and PK, i.e., (24) and (17). In particular, we considered the benchmark system introduced in [36], described by a Volterra series with order $r = 3$ and memory $m = 6$. Then, to evaluate the advantages of the SED-MPK (32), we used the estimators based on the proposed kernels to identify a more challenging system, previously considered in [38], and modeled as Volterra series with $m = 70$ and $r = 3, 4, 5$. Finally, we tested the estimator based on the SED-MPK with data coming from a real system, the cascaded water tanks benchmark system [34,33].

5.1 Experiment 1

The system considered in this experiment is described by the following equation

$$\begin{aligned} z_t =& u_t + 0.6u_{t-1} + 0.35(u_{t-2} + u_{t-4}) - 0.25u_{t-3}^2 \\ &+ 0.2(u_{t-5} + u_{t-6}) + 0.9u_{t-3} \\ &+ 0.25u_t u_{t-1} + 0.75u_{t-2}^3 - u_{t-1}u_{t-2} \\ &+ 0.5(u_t^2 + u_t u_{t-2} + u_{t-1}u_{t-3}). \end{aligned} \tag{37}$$

The estimator based on the MPK is compared with the ones based on the homogeneous polynomial kernel (PK) and the inhomogeneous polynomial kernel (IPK), expressed, respectively, by (18) and (24). The input signals are 1000 samples obtained from a realization of Gaussian noise. Concerning m_u^{tr}, m_u^{ts}, σ_u^{tr} and σ_u^{ts}, which are, respectively the input mean and standard deviation of the training and test samples, we considered two different scenarios :

- *Setup 1*: $m_u^{tr} = m_u^{ts} = 0$, $\sigma_u^{tr} = \sigma_u^{ts} = 4$;
- *Setup 2*: $m_u^{tr} = m_u^{ts} = 0$, $\sigma_u^{tr} = 1$, $\sigma_u^{ts} = 4$.

Notice that in *Setup 1* the distribution of the training and test inputs is the same, while in *Setup 2* is different. In particular, we limited the variability of the training samples, increasing the probability of generating test samples that are significantly distant from the training distribution. Consequently, the second scenario is more challenging. In all the experiments the noise standard deviation is $\sigma_n = 4$. For each setup we performed a Monte Carlo experiment composed of 100 simulations. For each simulation we generated a training and test dataset used to train and test the three estimators. Performance is measured by the percentage fit (Fit%), defined as

$$100\% \left(1 - \frac{\|\boldsymbol{z} - \hat{\boldsymbol{z}}\|_1}{\|\boldsymbol{z} - \bar{z}\|_1}\right),$$

where $\boldsymbol{z}$ is a vector collecting the noiseless test outputs, $\hat{\boldsymbol{z}}$ is the vector of the estimated outputs, and, finally, $\bar{z}$ is the mean of $\boldsymbol{z}$. The hyperparameters of the kernels have been trained maximizing the marginal likelihood of the training samples. Concerning the likelihood optimization, we used standard gradient descent algorithm. The algorithms are implemented in Pytorch [24], to exploit its automatic differentiation capabilities for computing the gradient.

Results are reported in Figure 1. As far as *Setup 1* is concerned, all the three estimators perform well. This is due to the fact that the input distribution does not vary between training and test, combined with the low value of m. On the contrary, performance varies significantly in the second scenario, where the MPK-based estimator outperforms the other estimators; the MPK Fit is 10% higher than the IPK and PK Fit. Thanks to the higher flexibility provided by the additional parameters, the MPK extracts more information from the training data, obtaining better out of sample performance.

To closely inspect the regularization properties of the three kernels, we analyzed the penalties assigned to the different monomials. In particular, we focused on $\boldsymbol{\phi}_2$, i.e. the monomials with degree two (see (3)), and we computed the penalty assigned to the h_2 map, i.e. the λ_{h_2} defined by (23). In Figure 2, we compare the module of the h_2 coefficients of the system in (37) with the λ_{h_2} computed by estimators based on the IPK, PK and MPK after tuning the hyperparameters by marginal likelihood maximization. We recall that small values of λ entail high penalization. Results confirm the consideration done in Section 2 and 3. Notice that the IPK and PK estimators are not able to penalize properly the different monomials. In particular, IPK assigns high value only to $\lambda_{h_2(-2,-2)}$, the coefficient of the monomial u_{k-2}^2. This is due to the constraints between penalties assigned to the different orders. Indeed, notice that in (37) the only monomial with degree three is u_{k-2}^3. Instead, PK is not able to select the relative degree of each monomial; for example, the coefficient of u_{k-1}^2 is promoted despite its maximum relative degree in (37) is one. Finally, results confirm that MPK can promote monomials depending on the relative degrees. Indeed, the coefficient of the monomial $u_{k-2}u_{k-3}$ is promoted without promoting u_{k-2}^2 and u_{k-3}^2, differently from the PK, and in accordance with (37). Additional tests with this system has been performed in [20], where we considered also nonzero mean input signals.

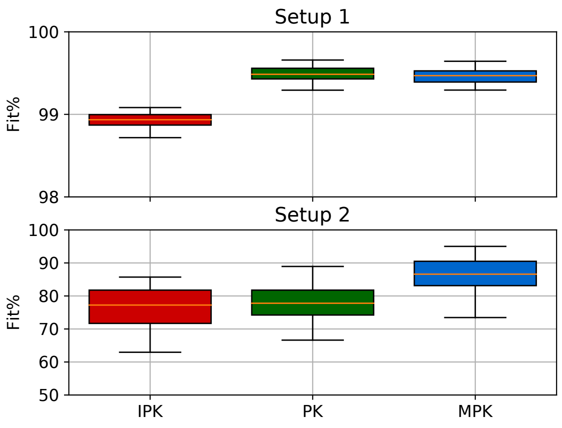

Fig. 1. Boxplot of the Fit obtained with a Monte Carlo simulation composed of 100 simulations of the system described by (37). IPK, PK, and MPK correspond to the estimators based on (24), (18), and (25), respectively.

5.2 Experiment 2

In this set of experiments we compare the performance of the estimators based on MPK and SED-MPK, i.e., (26) and (32), simulating a more complex system. We

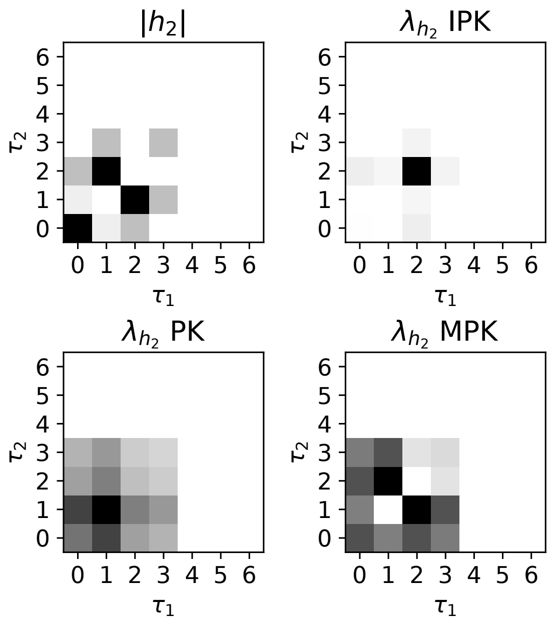

Fig. 2. Visualization of the h_2 coefficients' module related to the system in (37), compared with the λ_{h_2} obtained with the IPK, PK, and MPK. The dark squares represent higher values. For each matrix the values have been normalized w.r.t. the maximum value, i.e., black squares correspond to 1, while white squares to zero.

considered the Wiener system proposed in [38], and described by the following equation,

$$z_t = \sum_{i=1}^{r} \left(\frac{10q^{-1}}{A(q)} u_t \right)^i, \tag{38}$$

where q is the forward shift operator, and $A(q) = 1 - 1.8036q^{-1} + 0.8338q^{-2}$. To analyze the robustness of the two kernels w.r.t. measurement noise, we performed simulations with different level of noise. Specifically, we varied σ_n obtaining simulations with signal to noise ration (SNR) approximately equal to $5dB$ and $20dB$. Concerning the Volterra orders, we considered $r = 2, \ldots 5$, instead, the system memory m has been set to 70. For each Volterra order we performed a Monte Carlo study composed of 40 simulations. In each simulation the training and test inputs are the collection of 3500 samples, generated from a Gaussian distribution with zero mean and unitary standard deviation. Notice that the number of samples is approximately equal to the number of distinct coefficients of a Volterra map with $r = 2$ and $m = 70$, and significantly lower than the number monomials in $\boldsymbol{\phi}_i$ with $i > 2$. As concerns the SED-MPK, we considered the special case reported in (36), with all the α_p and β_p equal

In Figure 3 we visualized the Fit obtained for $r = 2$ and $r = 3$, after training the hyperparameters by marginal likelihood maximization. Results show that, compared

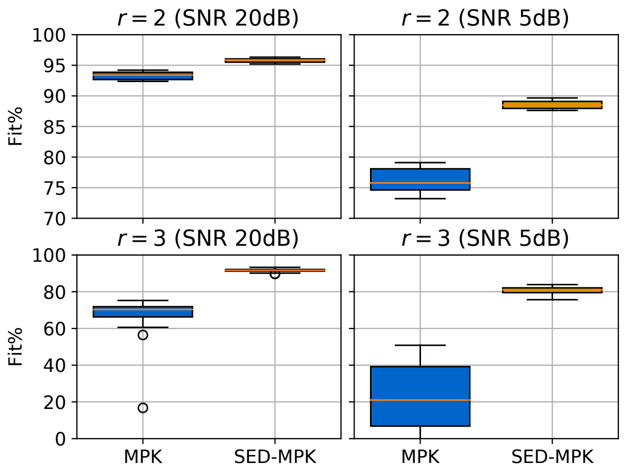

Fig. 3. Comparison of the Fit obtained with the estimators based on the MPK and SED-MPK in a Monte Carlo study composed of 40 simulations of the system in (38), with $r = 2, 3$.

to SED-MPK based estimator, the performance of the estimator based on the MPK decreases faster with the increasing of r, as well as being more sensitive to measurement noise. Due to the high number of parameters, and the small number of samples, the MPK is not able to identify a convenient set of penalties capable of providing generalization and robustness to measurement noise. Instead, results show the effectiveness of the regularization strategy implemented by the SED-MPK. Notice that the SED-MPK based estimator provides accurate estimates of the system output also with $r = 3$, as well as being robust to the presence of noise. This trend is confirmed also by results reported in Figure 4, where we plotted the SED-MPK Fit obtained with $r = 3, 4, 5$. Despite the number of samples is order of magnitudes smaller than the number of elements in ϕ_5, the average Fit is higher that the 80%. Finally, notice that effects of noise become particularly relevant only with $r = 5$, where there have been several outliers. However, we would like to stress that the configuration considered is particularly challenging.

5.3 Experiment 3: cascaded water tanks benchmark

In this experiment we tested the estimators based on the SED-MPK with data coming from a real system, the water tanks benchmark system [34,33]. This system is composed of two vertically cascaded water tanks, a reservoir and a water pump. The water flows from the highest tank to the reservoir, passing through second tank. A voltage controlled pump can move the water from the reservoir to the first tank, while the water level of the second tank can be measured through a capacitive water level sensor which returns a voltage signal. The goal of this benchmark is deriving an input-output model of the water level in the second tank, considering as input the pump voltage. The training and test dataset consists of 1024

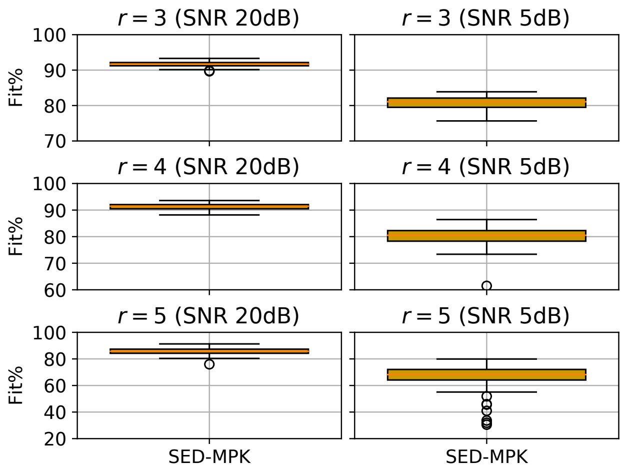

Fig. 4. Boxplot of the Fit obtained with the SED-MPK based estimator in a Monte Carlo study composed of 40 simulations of the system in (38), with $r = 3, 4, 5$.

samples on each, collected exciting the system with two distinct random phase multisine, with frequency ranging $[0, 0.0144]Hz$. A detailed description of the setup, the datasets ,and a video of the experiment are publicly available [2] . Due to the presence of overflows and the water level saturation, the input-output relation is highly nonlinear. Both the training and the test datasets contain at least two overflows at different time instants. The main challenges of this benchmark are not only the presence of strongly nonlinear behaviors but also the limited number of data and the presence of transient.

We used the estimators based on the SED-MPK to derive a system model, assuming that the input-output relation is a Volterra series with $r = 3$, and $m = 100$. Due to the limited number of samples, we did not consider the implementation of the estimators based on the PK and MPK kernel, i.e., (17) and (26). Notice that, when considering a Volterra series with $r = 3$ and $m = 100$, the number of hyperparameters in (17) and (26) is, respectively, 300 and 600. On the contrary, the SED-MPK, i.e., (32), has just 6 hyperparameters. We trained the estimator by maximizing the marginal likelihood of the training dataset. The performance of SED-MPK is compared with the results previously obtained on this benchmark relying on Volterra series. In particular, we considered the results reported in [3,4], where authors also described the evolution of the water level with a third order regularized Volterra series (RVS), applying the strategy described in Section 2 that relies on (6),(7).

Figure 5 plots the test output together with the estimate $\hat{y}$ returned by SED-MPK. We also report the percentage fit, comparing also the root mean squared error (RMSE) and the training time of RVS and SED-MPK. The SED-MPK estimator is effective with fit equal to 93.43%. It

[2] *http://www.nonlinearbenchmark.org/#Tanks*

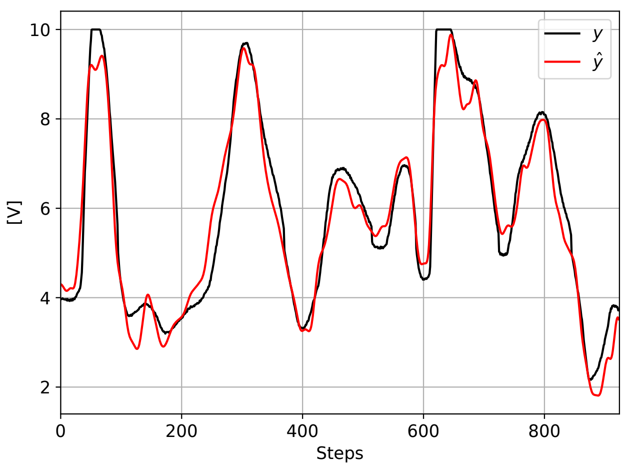

Fig. 5. Evolution of y and $\hat{y}$, i.e., the water tanks system output, and the estimate obtained with the SED-MPK based estimator modeling the input-output relation with a Volterra series with $r = 3$ and $m = 100$. In the table below we compare the numerical values of the RMSEs, the Fit, and the training time of our solution and the RVS approach proposed in [3,4] (the RVS's Fit is not available).

	RVS	SED-MPK
$RMSE[V]$	0.54	0.48
$Fit\%$	N.A	93.43
Training time	6.5[hours]	2[minutes]

performs similarly to RVS (the two RMSEs are close each other) but its computational time is considerably smaller. In fact, while RVS requires 6.5 hours to identify the model, our approach needs only 2 minutes. As explained in Section 2, this is due to the fact that the computational time of the solution proposed in [3,4] depends on the number of coefficients of the Volterra maps while the computational time of SED-MPK depends only on the number of training samples.

6 Conclusion

In this paper, we introduced two new kernels for Volterra series identification. They are named *multiplicative polynomial kernel* and *smooth exponentially decaying multiplicative polynomial kernel*. The MPK of degree i is defined as the product of i basic building blocks, represented by distinct linear kernels. Compared to standard polynomial kernels, MPK identifies the same RKHS, but it is equipped with a richer set of hyperparameters that allows for a flexible regularization, allowing to penalize monomials w.r.t. their maximum relative degree. Similarly to MPK, SED-MPK is the product of distinct linear kernels. However, SED-MPK also encodes the smooth exponentially decaying of the Volterra maps coefficients. In this way, we combine encoding properties of the polynomial kernel with explicit regularization of the Volterra maps, increasing significantly regularization and data-efficiency. In particular, our new approaches allow to identify also high-order Volterra series using a limited number of samples. Experimental results on artificial and real data show their effectiveness.

References

[1] S.A. Billings, A. Chen, and M.J. Korenberg. Identification of MIMO non-linear systems using a forward-regression orthogonal algorithm. *Intern. J. of Control*, 49:2157 – 2189, 1989.

[2] G. Birpoutsoukis, A. Marconato, J. Lataire, and J. Schoukens. Regularized nonparametric volterra kernel estimation. *Automatica*, 82:324 – 327, 2017.

[3] Georgios Birpoutsoukis, Pter Zoltn Csurcsia, and Johan Schoukens. Nonparametric volterra series estimate of the cascaded water tanks using multidimensional regularization. *IFAC-PapersOnLine*, 50(1):476 – 481, 2017. 20th IFAC World Congress.

[4] Georgios Birpoutsoukis, Pter Zoltn Csurcsia, and Johan Schoukens. Efficient multidimensional regularization for volterra series estimation. *Mechanical Systems and Signal Processing*, 104:896 – 914, 2018.

[5] 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.

[6] S. Chen, S. A. Billings, and W. Luo. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control*, 50:1873–1896, 1989.

[7] C.M. Cheng, Z.K. Peng, W.M. Zhang, and G. Meng. Volterra-series-based nonlinear system modeling and its engineering applications: A state-of-the-art review. *Mechanical Systems and Signal Processing*, 87:340 – 364, 2017.

[8] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39:1–49, 2001.

[9] B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[10] M. Espinoza, J. A. K. Suykens, and B. De Moor. Kernel based partially linear models and nonlinear identification. *IEEE Trans. on Automatic Control*, 50(10):1602–1606, 2005.

[11] M.O. Franz and B. Schölkopf. A unifying view of Wiener and volterra theory and polynomial kernel regression. *Neural Computation*, 18:3097–3118, 2006.

[12] R. Frigola, F. Lindsten, T.B. Schon, and C.E. Rasmussen. Bayesian inference and learning in Gaussian process state-space models with particle mcmc. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[13] R. Frigola and C.E. Rasmussen. Integrated pre-processing for Bayesian nonlinear system identification with Gaussian processes. In *Proceedings of the 52nd Annual Conference on Decision and Control (CDC)*, 2013.

[14] R. Haber and H. Unbehauen. Structure identification of nonlinear systems-a survey. *Automatica*, 26:651–677, 1990.

[15] J. Hall, C.E. Rasmussen, and J. Maciejowski. Modelling and control of nonlinear systems using Gaussian processes with partial model information. In *Proceedings of the 51st Annual Conference on Decision and Control (CDC)*, 2012.

[16] T. J. Hastie, R. J. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference and Prediction.* Springer, Canada, 2001.

[17] X. Hong, R. J. Mitchell, S. Chen, C. J. Harris, K. Li, and G. W. Irwin. Model selection approaches for non-linear system identification: A review. *International Journal of Systems Science*, 39(10):925–946, 2008.

[18] Z.Q. Lang, S.A. Billings, R. Yue, and J. Li. Output frequency response function of nonlinear volterra systems. *Automatica*, 43(5):805 – 816, 2007.

[19] Y.F. Li, L.J. Li, H.Y. Su, and J. Chun. Least squares support vector machine based partially linear model identification. In De-Shuang Huang, Kang Li, and GeorgeWilliam Irwin, editors, *Intelligent Computing*, volume 4113 of *Lecture Notes in Computer Science*, pages 775–781. Springer Berlin Heidelberg, 2006.

[20] Alberto Dalla Libera, Ruggero Carli, and Gianluigi Pillonetto. A novel multiplicative polynomial kernel for volterra series identification. *CoRR*, abs/1905.07960, 2019.

[21] I. Lind and L. Ljung. Regressor selection with the analysis of variance method. *Automatica*, 41(4):693 – 700, 2005.

[22] I. Lind and L. Ljung. Regressor and structure selection in NARX models using a structured ANOVA approach. *Automatica*, 44:383–395, 2008.

[23] L. Ljung. *System Identification - Theory for the User.* Prentice-Hall, Upper Saddle River, N.J., 2nd edition, 1999.

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[25] G. Pillonetto and A. Chiuso. Tuning complexity in regularized kernel-based regression and linear system identification: The robustness of the marginal likelihood estimator. *Automatica*, 58:106–117, 2015.

[26] G. Pillonetto and G. De Nicolao. A new kernel-based approach for linear system identification. *Automatica*, 46(1):81–93, 2010.

[27] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung. Kernel methods in system identification, machine learning and function estimation: a survey. *Automatica*, 50(3):657–682, 2014.

[28] T. Poggio and F. Girosi. Networks for approximation and learning. In *Proceedings of the IEEE*, volume 78, pages 1481–1497, 1990.

[29] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, 2006.

[30] W.J. Rugh. *Nonlinear System Theory: The Volterra-Wiener Approach.* Johns Hopkins University Press, 1980.

[31] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. *Neural Networks and Computational Learning Theory*, 81:416–426, 2001.

[32] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* (Adaptive Computation and Machine Learning). MIT Press, 2001.

[33] M. Schoukens, Per Mattsson, Torbjörn Wigren, and J.M.M.G. Noël. Cascaded tanks benchmark combining soft and hard nonlinearities. In *Workshop on Nonlinear System Identification Benchmarks : April 25-27, 2016, Brussels, Belgium*, pages 20–23, 4 2016.

[34] M. Schoukens and J.P. Nol. Three benchmarks addressing open challenges in nonlinear system identification. *IFAC-PapersOnLine*, 50(1):446 – 451, 2017. 20th IFAC World Congress.

[35] J. Sjöberg, Q. Zhang, L. Ljung, B. Delyon A. Benveniste, P. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: A unified overview. *Automatica*, 31(12):1691–1724, December 1995.

[36] W. Spinelli, L. Piroddi, and M. Lovera. On the role of prefiltering in nonlinear system identification. *IEEE Transactions on Automatic Control*, 50(10):1597–1602, Oct 2005.

[37] J. G. Stoddard, J. S. Welsh, and H. Hjalmarsson. Em-based hyperparameter optimization for regularized volterra kernel estimation. *IEEE Control Systems Letters*, 1(2):388–393, 2017.

[38] Jeremy G Stoddard and James S Welsh. Volterra kernel identification using regularized orthonormal basis functions. *arXiv preprint arXiv:1804.07429*, 2018.

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

[40] Y.L. Xu and D.R. Chen. Partially-linear least-squares regularized regression for system identification. *IEEE Trans. Automat. Contr.*, 54(11):2637–2641, 2009.

A Proof of Proposition 1

We prove Proposition 1 exploiting the Bayesian interpretation of kernel-based estimation. We recall that in such context the output y is assumed to be the sum of a function f sampled on some input locations and of independent Gaussian noises. The target function f is modeled as a zero-mean Gaussian random field with the kernel to define its covariance, see [29]. Then, we have

$$\begin{aligned}&\mathbb{E}[f(\boldsymbol{u}_t), f(\boldsymbol{u}_j)] = k(\boldsymbol{u}_t, \boldsymbol{u}_j)\\ &\boldsymbol{y} \sim N\left(0, \mathbb{K} + \gamma I_N\right)\end{aligned}$$

where $\mathbb{K}$ is the kernel matrix previously defined, while the regularization parameter γ corresponds to the variance of the measurement noise. If the kernel admits a closed form expansion in $\boldsymbol{\phi}$, the kernel k can be rewritten as $k(\boldsymbol{u}_t, \boldsymbol{u}_j) = \boldsymbol{\phi}^T(\boldsymbol{u}_t)\Lambda\boldsymbol{\phi}(\boldsymbol{u}_j)$, then we have $f(\boldsymbol{u}_t) = \boldsymbol{\phi}^T(\boldsymbol{u}_t)\boldsymbol{w}$ with $\boldsymbol{w} \sim N(0, \Lambda)$, see [29]. Consequently, the vector composed by the concatenation of $\boldsymbol{y}$ and $\boldsymbol{w}$ has Gaussian distribution, more precisely we have

$$\begin{bmatrix}\boldsymbol{y}\\ \boldsymbol{w}\end{bmatrix} \sim N\left(0, \begin{bmatrix}\mathbb{K} + \gamma I_N & C\\ C^T & \Lambda\end{bmatrix}\right),$$

with $C = \mathbb{E}[\boldsymbol{y}\boldsymbol{w}^T] = \Phi^T\Lambda$. The proof is concluded recalling that conditional distribution of $\boldsymbol{w}$ given $\boldsymbol{y}$ is Gaussian, and its maximum a posteriori estimator corresponds to the mean given by

$$\hat{\boldsymbol{w}} = C^T(\mathbb{K} + \gamma I_N)^{-1}\boldsymbol{y} = \Lambda\Phi\boldsymbol{\alpha}.$$