Learning Nonparametric Volterra Kernels with Gaussian Processes

Anonymous Author(s)
Affiliation
Address
email

Abstract

This paper introduces a method for the nonparametric Bayesian learning of nonlinear operators, through the use of the Volterra series with kernels represented using Gaussian processes (GPs), which we term the nonparametric Volterra kernels model (NVKM). When the input function to the operator is unobserved and has a GP prior, the NVKM constitutes a powerful method for both single and multiple output regression, and can be viewed as a nonlinear and nonparametric latent force model. When the input function is observed, the NVKM can be used to perform Bayesian system identification. We use recent advances in efficient sampling of explicit functions from GPs to map process realisations through the Volterra series without resorting to numerical integration, allowing scalability through doubly stochastic variational inference, and avoiding the need for Gaussian approximations of the output processes. We demonstrate the performance of the model for both multiple output regression and system identification using standard benchmarks.

1 Introduction

Gaussian processes (GPs) constitute a general method for placing prior distributions over functions, with the properties of samples from the distribution being controlled primarily by the form of the covariance function [23]. Process convolutions (PCs) are one powerful method for building such covariance functions [4, 13, 3]. In the PC framework, the function we wish to model is assumed to be generated by the application of some convolution operator to a base GP with some simple covariance, and since linear operators applied to GPs result in GPs, the result is another GP with a covariance we deem desirable. PCs allow models for multiple correlated output functions to be built with ease, by assuming each output is generated by a different operator applied to the same base function, or set of functions [30, 13].

The PC framework unifies a number of different ideas in the GP literature. Latent force models (LFMs) [1] use PCs to include physics based inductive biases in multiple output GP (MOGP) models by using the Green's function of a linear differential operator as the kernel of the convolution. This leads to the interpretation of each output as having been generated by inputting a random latent force into a linear system, with physical properties described by the differential operator. The Gaussian process convolution model (GPCM) of Tobar et al. [29] treats the convolution kernel itself as an unknown function to be inferred from data, and places a GP prior over it. Linear systems are entirely described by their Green's function, so one can interpret the GPCM as a nonparametric, linear LFM, in which the form of the system itself is inferred from data.

In the physical world, nonlinear systems are the norm, and linearity is an approximation. As a consequence it is desirable when dealing with physical data to have models that can incorporate nonlinearity naturally. Often however, given a certain set of data, it is not clear exactly what form this nonlinearity takes, and so introducing specific parametric non-linear operators can be overly

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

restrictive. Álvarez et al. [2] present a model known as the nonlinear convolved MOGP (NCMOGP) which introduces nonlinearity to MOGPs via the Volterra series [8], a nonlinear series expansion used widely for systems identification, whose properties are controlled by a set of square integrable functions of increasing dimensionality known as the Volterra kernels (VKs). The NCMOGP assumes these functions are both separable and homogeneous, and of parametric Gaussian form, it also approximates the outputs as GPs in order to make inference tractable.

The present work introduces a new model which drops the separability and homogeneity assumptions on the VKs, allows their form to be learned directly from data, and makes no approximation on the distribution of the outputs. We refer to it as the nonparametric Volterra kernels model (NVKM). We develop a fast sampling method for the NVKM which leverages the recent results of Wilson et al. [35] on the sampling of explicit functions from GPs to analytically map function realisations through the Volterra series, avoiding the need for computationally expensive and inaccurate high dimensional numerical integration. Fast sampling allows for the application of doubly stochastic variational inference (DSVI) [28] for scalable learning.

The NVKM is well suited to both single and multiple output regression problems, and can be thought of as an extension of the GPCM to both non-linear systems and multiple outputs. The NVKM can also be interpreted as a non-linear LFM in which the operator is learned directly from data. We additionally present a variation of the NVKM that can be used for Bayesian systems identification, where the task is to learn operator mappings between observed input and output data, and show that it allows for considerably better quantification of uncertainty than competing methods which use recurrence [18].

2 Background

In this section we give a brief introduction to the mathematical background of PCs and the Volterra series.

Process convolutions In the PC framework, the set of output functions $\{f_d(t)\}_{d=1}^D$, with D being the number of outputs, is generated by the application of some set of linear operators, specifically convolution operators, to a latent function u represented by a Gaussian process, $f_d(t) = \int_{\mathcal{T}} G_d(t - \tau)u(\tau)d\tau$, where $\mathcal{T}$ is the domain of integration, and the function G_d is known variously as the convolutional kernel, smoothing kernel, impulse response or Green's function, depending on context. This function must be square integrable to ensure the output is finite. A linear operator acting on a GP produces another GP [23], and so we obtain D distinct GPs. Since the latent function u is shared across the outputs, these D GPs are correlated, allowing joint variations to be captured, whilst the convolution with G_d adapts u to each output. Álvarez et al. [3] show that many MOGP models can be recast in terms of the PC framework by particular choices of G_d and u. In LFMs, G_d is taken to be the Green's function of some differential operator. We can then interpret each output as resulting from a shared random force being fed into a distinct linear system, represented by the differential operator. The smoothing kernels are usually taken to have parametric, often Gaussian form. Tobar et al. [29] use the PC framework for a single output, and make the smoothing kernel itself a GP, giving rise to the GPCM. Bruinsma [7] extends the GPCM to the multiple output case, although the model was not applied to data.

Voterra Series The PC framework can be extended to represent a broader class of output functions by instead considering the outputs $\{f_d(t)\}_{d=1}^D$ as being the result of some non-linear system, acting on the latent function u. The Volterra series is a series approximation for non-linear systems that is widely used in the field of system identification [8]. It is given by,

$$f(t) = \sum_{c=1}^{C} \int_{\mathcal{T}} G_c(t - \tau_1, \ldots, t - \tau_c) \prod_{j=1}^{c} u(\tau_j)\mathrm{d}\tau_j, \tag{1}$$

where f is the system output, G_c is the cth order VK, u is the system input, and C is the order of the approximation. We can think of the Volterra series as an extension of the well known Taylor expansion, that allows f to have memory of the past values of u, that is to say f depends on u at all values $t \in \mathcal{T}$, and can approximate a broad class of non-linear operators. Álvarez et al. [2] use Equation (1) to construct the NCMOGP, applying the Volterra series with d distinct sets of

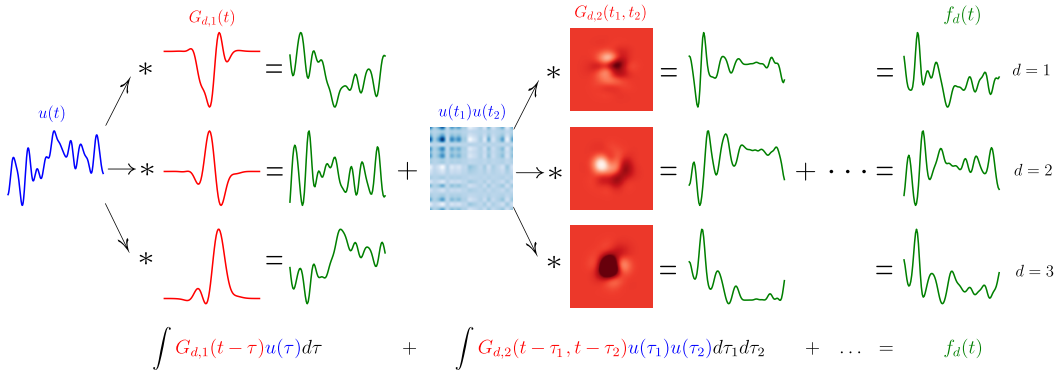

Figure 1: A diagram of the generative process for the NVKM with $C = 2$ and $D = 3$, showing the stages of computation of the first order (on the left side) and second order (in the center) terms of the Volterra series for three outputs, shown in the rows, with the star representing a convolution. The 1D contribution from the second order term is obtained by taking the diagonal of the result of the 2D convolution.

VKs $\{G_{d,c}\}_{c=1}^{C}$ to a shared latent GP input u, to produce outputs $\{f_d(t)\}_{d=1}^{D}$. Since the Volterra series is a nonlinear operator, the output becomes an intractable, non-Gaussian process. The authors perform inference by approximating the outputs as GPs and using the first and second moments of the output process to form its mean and covariance function. To enable to computation of these moments, the authors restrict the set of VKs to those which are both separable and homogeneous, i.e. $G_{d,c}(t_1, \ldots, t_c) = \prod_{i=1}^{c} G_d(t_i)$. Additionally, since the moment computation requires analytically solving a number of non-trivial convolution integrals, the authors only consider a Gaussian form for the VKs.

3 The nonparametric Volterra kernels model

The NVKM relaxes the restrictions of separability and homogeneity which are placed on the VKs in the NCMOGP, and represents these kernels as independent GPs, allowing their form and uncertainty to be inferred directly from data. The generative process for the NVKM can be stated as

$$\begin{aligned} u(t) &\sim \mathcal{GP}(0, k^{(u)}(t, t')), \\ G_{d,c}(\mathbf{t}) &\sim \mathcal{GP}(0, k^{(G_{d,c})}(\mathbf{t}, \mathbf{t}')), \quad \mathbf{t} \in \mathbb{R}^c, \quad \forall c \in 1, \ldots, C, \quad \forall d \in 1, \ldots, D, \\ f_d(t) &= \sum_{c=1}^{C} \int_{-\infty}^{\infty} G_{d,c}(t - \tau_1, \ldots, t - \tau_c) \prod_{j=1}^{c} u(\tau_j) \mathrm{d}\tau_j, \end{aligned} \tag{2}$$

where $k^{(u)}(t, t')$ is the covariance function for the input process, and $k^{(G_{d,c})}(\mathbf{t}, \mathbf{t}')$ is the covariance function for the cth VK of the dth output. We follow Tobar et al. [29] in using the decaying square exponential (DSE) covariance for the VKs, which is a modification to the ubiquitous square exponential (SE) covariance that ensures the samples are square integrable. The DSE covariance has the form

$$k_{DSE}(\mathbf{t}, \mathbf{t}') = \sigma^2 \exp(-\alpha(\|\mathbf{t}\|^2 + \|\mathbf{t}'\|^2) - \gamma\|\mathbf{t} - \mathbf{t}'\|^2), \tag{3}$$

where $\|\cdot\|$ is the or ℓ^2 norm, σ is the amplitude, α controls the rate at which the samples decay away from the origin, and γ is related to the length scale l of the samples by $\gamma = \frac{1}{l^2}$. A diagram of the generative process for the model is shown in Figure 1. Obtaining an exact distribution over the outputs f_d is intractable, since it involves integration over nonlinear combinations of infinite dimensional stochastic processes. In order to sample from the model, and perform inference, approximations must be introduced. In particular, we employ the results of Wilson et al. [35] to sample in linear time, which enables efficient learning through the use of variational inducing points [27] with doubly stochastic variational inference (DSVI) [28].

3.1 Sampling

One could sample from the model by drawing from the input and filter GPs at some finite set of locations, and then using the samples with some method for numerical integration to find the output. As the dimensionality of the filters increases, however, many points would be needed to obtain an accurate answer, this quickly becomes computationally intractable, since sampling exactly from a GP has cubic time complexity with respect to the number of points requested. We can sidestep this problem, and avoid the need for any numerical integration, by representing samples from the GPs explicitly as functions. Using the results from Wilson et al. [35], and following their notation, we can write a sample from a GP $f : \mathbb{R}^c \to \mathbb{R}$, with covariance function $k(\mathbf{t}, \mathbf{t}')$, given M inducing variables $\mathbf{v} \in \mathbb{R}^M$ with corresponding inducing inputs $\{\mathbf{z}_j\}_{j=1}^M$, with $\mathbf{z}_j \in \mathbb{R}^c$, as

$$(f|\mathbf{v})(\mathbf{t}) = \sum_{i=1}^{N_b} w_i \phi_i(\mathbf{t}) + \sum_{j=1}^{M} q_j k(\mathbf{t}, \mathbf{z}_j), \tag{4}$$

where $\{\phi_i\}_{i=1}^{N_b}$ is a random Fourier basis, with N_b being the number of basis functions in the approximation, $\mathbf{w} \in \mathbb{R}^{N_b}$ with entries $w_i \sim \mathcal{N}(0, 1)$, and q_j are the entries of the vector $\mathbf{q} = \mathbf{K}^{-1}(\mathbf{v} - \Phi\mathbf{w})$, where $\mathbf{K} \in \mathbb{R}^{M \times M}$, with elements $K_{m,n} = k(\mathbf{z}_n, \mathbf{z}_m)$, is the covariance matrix of the inducing points, and $\Phi \in \mathbb{R}^{M \times N_b}$ is a feature matrix, with each basis function being evaluated at each inducing location. The random Fourier basis is obtained by first sampling $\beta_i \sim U(0, 2\pi)$, where U is the uniform distribution, and then sampling $\boldsymbol{\theta}_i \sim FT(k)$, where FT is the Fourier transform of the covariance function, which is the spectral density of the process. The basis functions are then given by $\phi_i(\mathbf{t}) = \sqrt{2/N_l} \cos(\boldsymbol{\theta}_i^\top \mathbf{t} + \beta_i)$. We can see Equation (4) as consisting of an approximate GP prior, using a random Fourier features approximation [22], with a correction term which uses Matheron's update rule to account for the inducing points. By using Equation (4), samples can be obtained in linear time with respect to the number of requested points. It should be noted that Equation (4) only applies to GPs with stationary kernels, however, the DSE covariance required for $G_{c,d}$, is non-stationary. It can be shown that the process $\exp(-\alpha||\mathbf{t}||^2) G'_{c,d}(\mathbf{t})$ has the DSE covariance if $G'_{c,d}$ has the SE covariance. We can then write a sample from the output of the NVKM as,

$$(f_d|\mathbf{v}^G_{d,c}, \mathbf{v}^u)(t) = \sum_{c=1}^{C} \int_{-\infty}^{\infty} e^{-\alpha||\mathbf{t}||^2} (G'_{d,c}|\mathbf{v}^G_{d,c})(t - \tau_1, \ldots, t - \tau_c) \prod_{j=1}^{c} (u|\mathbf{v}^u)(\tau_j) \mathrm{d}\tau_j, \tag{5}$$

which can be computed analytically, assuming that u also has the SE covariance, by representing the Fourier basis in complex form, and factorising the integrals, leading to combinations of sums and products of single dimensional integrals of the form $\int_{-\infty}^{\infty} \exp(-ax^2 + bx) dx = \sqrt{\pi/a} \exp(b^2/4a)$. See the supplemental material for details of the computation.

3.2 Inference

Learning with the NVKM implies making inference of the input process u, along with all VKs $\{G_{d,c}\}_{c,d=1}^{C,D}$, from observed output data $\{\mathbf{y}_d\}_{d=1}^D$ with $\mathbf{y}_d \in \mathbb{R}^{N_d}$, which are the functions $\{f_d\}_{d=1}^D$ evaluated at points $\{\mathbf{t}_d\}_{d=1}^D$ with $\mathbf{t}_d \in \mathbb{R}^{N_d}$, corrupted by some i.i.d Gaussian noise. That is to say $y_{d,i} = f_d(t_{d,i}) + \epsilon_{d,i}$ with $\epsilon \sim \mathcal{N}(0, \sigma^2_{y_d})$. Let $\mathbf{v}^G_{d,c} = G_{d,c}(\mathbf{z}^G_{d,c})$ denote the inducing points for the VKs, and $\mathbf{v}^u = u(\mathbf{z}^u)$ denote the inducing points for the input. The joint distribution over these inducing points and the latent functions then has the following form

$$p(\{\mathbf{y}_d\}_{d=1}^D, \{G_{d,c}, \mathbf{v}^G_{d,c}\}_{c,d=1}^{C,D}, u, \mathbf{v}^u) = \prod_{d,i=1}^{D,N_d} p(y_{d,i}|f_d(t_{d,i})) \prod_{d,c=1}^{D,C} p(G_{d,c}|\mathbf{v}^G_{d,c}) p(\mathbf{v}^G_{d,c}) p(u|\mathbf{v}^u) p(\mathbf{v}^u), \tag{6}$$

where $f_d(t_{d,i})$ depends on the VKs and input through Equation (2), the likelihood is $p(y_{d,i}|f_d(t_{d,i})) = \mathcal{N}(y_{d,i}; f_d(t_{d,i}), \sigma^2_{y_d})$, $p(G_{d,c}|\mathbf{v}^G_{d,c})$ and $p(u|\mathbf{v}^u)$ are GP posterior distributions, and $p(\mathbf{v}^u)$ and $p(\mathbf{v}^G_{d,c})$ are the prior distributions over the inducing points. The dependency structure of the model is described in Figure 2. We form an approximate variational distribution, in a similar way to Tobar

et al. [29], using a structured mean field approximation. That is to say, we mirror the form of the true joint distribution, and replace the prior distributions over the inducing points with variational distributions, $q(\mathbf{v}^G_{d,c})$ and $q(\mathbf{v}^u)$, leading to the variational distribution

$$q(\{G_{d,c}, \mathbf{v}^G_{d,c}\}_{c,d=1}^{C,D}, u, \mathbf{v}^u) = \prod_{d,c=1}^{D,C} p(G_{d,c}|\mathbf{v}^G_{d,c}) q(\mathbf{v}^G_{d,c}) p(u|\mathbf{v}^u) q(\mathbf{v}^u). \tag{7}$$

The optimal form of the variational posteriors are multivariate Gaussians, $q(\mathbf{v}^u) = \mathcal{N}(\mathbf{v}^u; \boldsymbol{\mu}^u, \boldsymbol{\Sigma}^u)$ and $q(\mathbf{v}^G_{d,c}) = \mathcal{N}(\mathbf{v}^G_{d,c}; \boldsymbol{\mu}^G_{d,c}, \boldsymbol{\Sigma}^G_{d,c})$, where the mean vectors and covariance matrices of these distributions are variational parameters. This form of variational approximation leads to a variational lower bound,

$$\mathcal{F} = \sum_{d,i=1}^{D,N_d} \mathbb{E}_q[\log p(y_{d,i}|f_d(t_{d,i}))] - \sum_{d,c=1}^{D,C} \mathrm{KL}[q(\mathbf{v}^G_{d,c}))||p(\mathbf{v}^G_{d,c}))] - \mathrm{KL}[q(\mathbf{v}^u)||p(\mathbf{v}^u)], \tag{8}$$

where KL[.||.] represents the Kullback-Lieber (KL) divergence. The expression above is optimised using gradient descent. The KL divergences have closed form. The derivation of the bound and KL divergences are given in the supplementary material. The expectation of the log likelihood of the outputs, given in the first term, is intractable, due to the nature of nonlinearity introduced by the Volterra series. We instead compute a stochastic estimate of the log likelihood by sampling from the model and using,

$$\mathbb{E}_q[\log p(y_{d,i}|f_d(t_{d,i}))] \approx \frac{1}{S} \sum_{s=1}^{S} \log p(y_{d,i}|(f_d|\mathbf{v}^G_{d,c}, \mathbf{v}^u)(t_{d,i})), \tag{9}$$

where $\mathbf{v}^G_{d,c}$ and $\mathbf{v}^u$ are first sampled from their respective variational distributions, and then used in Equation (5) to generate a sample from f_d. To make the inference scheme scalable, we compute the bound on randomly sub-sampled mini batches of the data set, which alone is known as stochastic variational inference [14]. When this source of stochasticity is combined with the stochastic estimate of the expected log likelihood, we have DSVI [28].

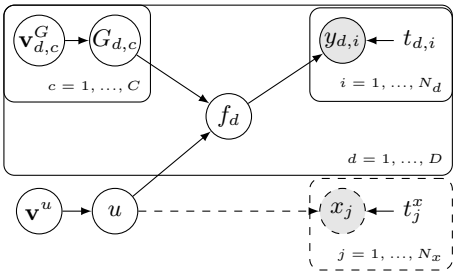

Figure 2: A graphical model for the NVKM, where the dashed elements are added to form the IO-NVKM. Note that nodes u, f_d, and $G_{d,c}$ are not random variables, they are random processes, but the distinction is not made in the diagram for the sake of clarity.

In the standard NVKM model, the input process u is a latent function with no observed data associated with it. There are many situations in which, instead of learning a distribution over some output functions alone, we wish to learn an operator mapping between an input function and an output function, or functions. That is to say, in addition to observing data $\{\mathbf{y}_d\}_{d=1}^D$ we also observed the input process u at locations $\mathbf{t}^x \in \mathbb{R}^{N_x}$ corrupted with i.i.d noise, which we denote $\mathbf{x}$, so $x_i = u(t^x_i) + \epsilon^x_i$ with $\epsilon^x \sim \mathcal{N}(0, \sigma^2_x)$. We apply a simple modification to the inference scheme of the NVKM, in order to form a new model which we term the input/output NVKM (IO-NVKM). For the IO-NVKM, we pick up an additional likelihood term in Equation (6), with the variational distribution remaining unchanged, and we obtain a new bound,

$$\mathcal{F}_{IO} = \mathcal{F} + \sum_{j=1}^{N_x} \mathbb{E}_q[p(x_j|u(t^x_j))], \tag{10}$$

where $p(x_j|u(t^x_j)) = \mathcal{N}(x_j; u(t^x_j), \sigma^2_x)$, $\mathcal{F}$ is given by Equation (8) and we approximate the expectation as in Equation (9). The relationship between the NVKM and IO-NVKM is illustrated in Figure 2 where the dashed sections are added to form the IO model.

4 Related Work

In the this section, we give a brief overview of existing ideas in the literature that have some connection to the NVKM.

Non-parametric kernels Covariance function design and selection is key to achieving good performance with GPs. Much work has been done on automatically determining covariance functions from data, for example by building complex covariances by composing simpler ones together [10], or by using deep neural networks to warp the input space in complex ways before the application of a simpler kernel [34]. Flexible parametric covariances can also be designed in frequency space [33]. Alternatively, some efforts have been made to learn covariance functions using GPs themselves. In addition the the GPCM, another model that learns covariances nonparametrically is due to Benton et al. [5], who use GPs to represent the log power spectral density, and then apply Bochner's theorem to convert this to a representation of the covariance function. GPs are fully specified by their first two moments, so by learning the covariance function and mean function, one knows all there is to know about the process. The present work uses the formalism of the Volterra series to learn the properties of more complex, non-Gaussian processes, non-parametrically. We can think of this as implicitly learning not just the first and second order moments of the process, but also the higher moments, depending on the value of C. In the case of $C = 1$ and $D = 1$, the NVKM and the GPCM are the same, except for the fact the GPCM uses the white noise as the input process, whereas we use an SE GP.

LFMs and MOGPs As discussed in Section 2, we can interpret the first order filter function as the Green's function of some linear operator or system, and so by placing a GP prior over it, we implicitly place a prior over some set of linear systems [29]. Since standard LFMs use an operator of fixed form, we can interpret the NVKM in the case $C = 1$, $D \geq 1$ as being an LFM in which the generating differential equation itself is learned from data. LFMs can be extended to cases in which the differential operator is nonlinear. Hartikainen et al. [12] recast a specific non-linear LFM in terms of a state space formalism allowing for inference in linear time. Lawrence et al. [17] use Markov chain Monte Carlo to infer the parameters of a specific nonlinear ODE describing gene regulation, using a GP as a latent input function. Ward et al. [31] use black box VI with inverse auto-regressive flows to infer parameters of the same ODE. When $C > 1$, the NVKM can be interpreted as a nonlinear nonparametric LFM. Álvarez et al. [2] use the fixed, parametric VKs to build an MOGP model. In contrast to the NVKM, they approximate the outputs as GPs and use analytical expressions for the moments to perform exact GP inference.

Nonlinear system identification The IO-NVKM falls into the class of models which aim to perform system identification. A key concern of systems identification is determining how the output of certain systems, often represented by differential operators, respond to a given input. GPs have long been used for the identification of both linear and nonlinear systems. Many models exist which use GPs to recurrently map between previous and future states, including GP-NARX models [16], various state space models [26, 25] and recurrent GPs (RGPs) [19]. The thesis of Mattos [18] gives a summary of these methods. Worden et al. [36] detail a method for the combination of the GP-NARX model with a mechanistic model based on the physical properties of the system under study, leading to improved performance over purely data driven approaches. The IO-NVKM differs from these models in that instead of learning a mapping from the state of the system at a given point to the next state, we use GPs to learn an operator that maps the whole input function to the whole output function.

5 Experiments

For all the following experiments we place the inducing locations for both the input process and VKs on a fixed grid. The GP representing the cth VK has input dimension c, which means that the number of inducing points required to fully characterise it scales exponentially with c if they are placed on a grid. For all experiments we use 15, 10, 6 and 4 inducing points per axis for each of the 1st to 4th order filters respectively, centered on zero. We treat the range of the points of each VK as a hyperparameter, and fix α such that the decaying part of the DSE covariance causes

Table 1: Performance on the synthetic data set, showing mean and standard deviation for 10 repeats.

Model	NMSE	NLPD
GPCM	0.199 ±0.023	1.080 ±0.130
NVKM ($C = 1$)	0.196 ±0.047	2.084 ±0.398
NVKM ($C = 2$)	0.108 ±0.065	0.638 ±0.580
NVKM ($C = 3$)	**0.055 ±0.016**	**0.124 ±0.107**
NVKM ($C = 4$)	0.084 ±0.087	0.149 ±0.331

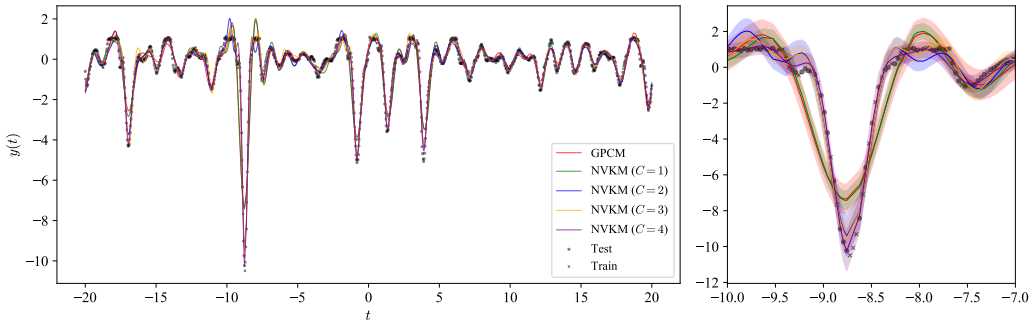

Figure 3: Model predictions on the synthetic data, with crosses indicating training points and dots indicating test points. The right plot shows an enhanced view of the peak around $t = -9$, and the shaded regions show 2σ confidence.

samples to be near zero at the edge of the range. For u we use approximately 1/10 of the number of inducing points as data points (average number of data per output for multi-output problems). The VK GP length scales, VK GP amplitudes, and input GP amplitude are optimised along with the variational parameters by maximising the variational bound using gradient descent. For computational reasons, the input process length scale is fixed based on the spacing of the input inducing points, and the noise hyperparameters are fixed to a small value whilst optimisation is taking place, and then fit alone afterwards. The model is implemented using the Jax framework [6]. For all experiments we use Adam [15]. All models were trained on a single Nvidia K80 GPU.

5.1 Synthetic data

To illustrate the advantage of including non-linearity in the model we generate a synthetic single output regression problem which includes both hard and soft nonlinearities by sampling g from an SE GP with length scale 2, computing $f_i(t) = \int e^{-2\tau^2} h_i(\tau) g(t-\tau) d\tau$ for $h_1(t) = \sin(6t)$, $h_2(t) = \sin^2(5t)$ and $h_3(t) = \cos(4t)$ by numerical integration, then computing the output as,

$$y(t) = \min(5 f_1(t) f_2(t) + 5 f_3^3(t), 1) + \epsilon, \tag{11}$$

with $\epsilon \sim \mathcal{N}(0, 0.05^2)$. We generate 1200 points in the range $t = [-20, 20]$ and use a random subset of a third for training and the rest for testing. Table 1 shows the normalised mean square errors (NMSEs) and negative log probability densities (NLPDs) on the test set for the NVKM with various values of C as well as the GPCM, with repeats using a different random train/test split, and different random seeds.[1] As we would expect, the NMSE values are very similar for the NVKM with $C = 1$ and the GPCM, since the models are nearly equivalent except for the prior on the input GP. Interestingly the NLPD values are better for the GPCM than the NVKM with $C = 1$, likely due to the fact we do not optimise the noise jointly with the bound. As C increases the performance of the NVKM improves until $C = 4$. The fact performance does not improve after $C = 3$ illustrates the difficulty of identifying higher order nonlinearities in a relatively small training set, an effect supported by the results of the *Cascaded Tanks* experiment in the following section. Although the $C = 4$ model does have more capacity to represent nonlinearities, the optimisation procedure is challenging, illustrated by the high variance of the results. Plots of the predictions for the model can be seen in Figure 3. We can see that increasing the non-linearity for the NVKMs allows the sharp spike and the finer grained features, as well as the hard nonlinearities, to be captured simultaneously.

5.2 Cascaded tanks

To demonstrate the IO-NVKM , we use a standard benchmark for non-linear systems identification know as *Cascaded Tanks* [24].[2] The system comprises two vertically stacked tanks filled with water, with water being pumped from a reservoir to the top tank, which then drains into the lower tank

[1] Results generated using the implementation available at `github.com/wesselb/gpcm`

[2] Available at `sites.google.com/view/nonlinear-benchmark/`

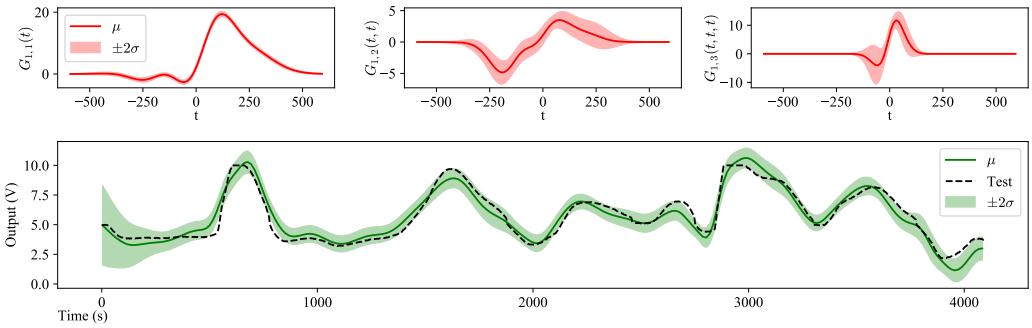

Figure 4: (Top) Diagonal of the inferred Volterra kernels for the the IO-NVKM with $C = 3$, showing 2σ confidence region. (Bottom) The predicted output for the test set, with the dashed line showing the true values.

and finally back to the reservoir. The training data is two time series of 1024 points, one being the input to the system, which is the voltage fed into the pump, and the second being the output, which is the measured water level in the lower tank. For testing, an additional input signal, again of 1024 points, is provided, and the task is to predict the corresponding output water level. The system is considered challenging because it contains hard nonlinearities when the tanks reach maximum capacity and overflow (see the regions around 600s and 2900s in Figure 4), it has unobserved internal state, and has a relatively small training set. Table 2 shows the predictive root mean square errors (RMSEs) and NLPDs for the IO-NVKM with various C, as well as four other GP based models for system identification from [18]. For each C, five random settings of VK ranges were tested, and each training was repeated three times with different initialisations. The setting and initialisation with the lowest combined NLPD on the training input and output data is shown. Although the RGP with $H = 2$ provides the best RMSE of the model, this comes at the cost of poor NLPD values. All IO-NVKMs achieve considerably better NLPD values than the alternatives indicating much better quantification of uncertainty. Of the IO-NVKMs, $C = 3$ performs best in both metrics. Figure 4 show the predictions of the $C = 3$ model on the test set, as well as the inferred VKs. The uncertainty in the VKs increases with their order, which is natural given the difficulty of estimating higher order nonlinear effects from a small training set. It should be noted that Worden et al. [36] achieve a much lower RMSE of 0.191 by using a specific physics model of the system in tandem with a GP-NARX model, but since we are considering purely data driven approaches here, it is not directly comparable.

5.3 Weather data

To illustrate the utility of the NVKM for multiple output regression problems, we consider a popular benchmark in MOGP literature, consisting of multiple correlated time series of air temperature measurements taken at four nearby locations on the south coast of England, originally described by Nguyen et al. [21], which we refer to as *Weather*.[3] The four series are named Bramblemet, Sotonmet, Cambermet and Chimet, with 1425, 1097, 1441, and 1436 data points, respectively. Bramblemet and Sotonmet both contain regions of truly missing data, 173 and 201 points in a continuous region are artificially removed form Cambermet and Chimet with the task being to predict them based on the all the other data. Table 3 shows the performance of the multiple output NVKM on the *Weather* dataset, along with the best performing NCMOGP model of Álvarez et al. [2]. For each C, five random settings of VK ranges were tested, with each training being repeated four times with different initialisations, the setting with the best average NLPD value on the training data is shown. All NVKM models show better or equivalent performance than the NCMOGP on the Cambermet output, but all show worse performance on the Chimet output, although on the Chimet

Table 2: Comparison of performance on the *Cascaded Tanks* dataset, with the last four models reported in [18]. H indicates the number of hidden layers in the RGP.

Model	RMSE	NLPD
IO-NVKM ($C = 1$)	0.835	1.724
IO-NVKM ($C = 2$)	0.716	1.311
IO-NVKM ($C = 3$)	0.532	**0.879**
IO-NVKM ($C = 4$)	0.600	0.998
RGP ($H = 1$)	0.797	2.33
RGP ($H = 2$)	**0.308**	7.79
GP-NARX	1.50	1080
Var. GP-NARX	0.504	119.3

[3] Available for download in a convenient from using the `wbml` package, `github.com/wesselb/wbml`

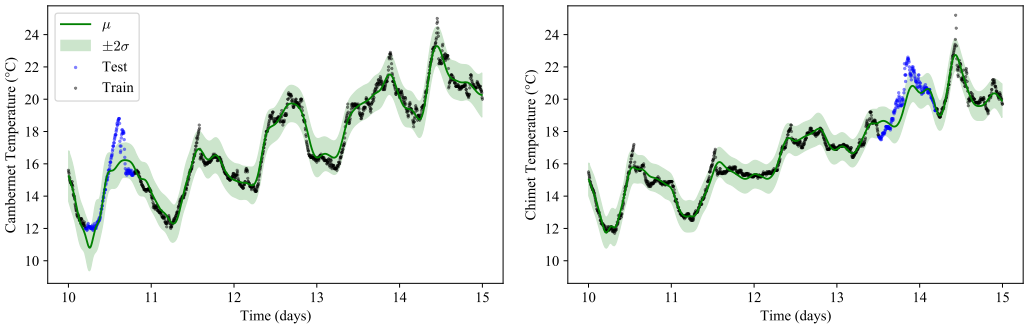

Figure 5: Predictive means and 2σ confidence regions from the NVKM with $C = 3$, on the Cambermet and Chimet outputs from the *Weather* data set. Blue points are the artificially removed test data, black points are training data.

Table 3: Comparison of performance on the *Weather* data set, for the NVKM mean and standard deviation of three initialisation is shown, along with the best model from [2]

	Cambermet		Chimet	
Model	NMSE	NLPD	NMSE	NLPD
NVKM ($C = 1$)	**0.212±0.085**	**2.182±0.743**	1.669±0.052	7.148±0.111
NVKM ($C = 2$)	0.440±0.286	3.884±2.380	0.939±0.216	4.143±1.197
NVKM ($C = 3$)	0.253±0.002	2.390±0.123	0.871±0.394	3.994±1.924
NCMOGP ($C = 3$)	0.44	2.33	**0.43**	**2.18**

output the variance between repeats is high. It should be noted that the LFM reported by Guarnizo and Álvarez [11] achieves much lower scores, having NMSEs of 0.11 and 0.19 on Cambermet and Chimet respectively, but that model uses six latent functions as opposed to a single latent function for the NVKM and NCMOGP. Including multiple latent functions may lead to large performance improvements for the NVKM and is a promising direction for future work.

6 Discussion

Societal Impacts Accurate methods for system identification are key to the functioning of modern aircraft [20], this includes military aircraft, and specifically unmanned aerial vehicles equipped with weapons. It is possible that improved models for system identification could lead such aircraft to be more effective, and thus more deadly. GP and MOGP models have long been applied to problems in robotics [9, 32]. Better inclusion of nonlinearities in these models may enhance the ability of robots, potentially leading to loss of jobs and livelihoods to automation.

Future Work The are a number of extensions to both the NVKM and IO-NVKM that could lead to substantial improvements in performance. As briefly mentioned in Section 5, the number of inducing points required for the VKs scales exponentially with the order of the series, meaning it is difficult to represent complex features in the higher order terms, without using a computationally intractable number of points. Whilst initially we saw the increased flexibility of non-separable VKs as a virtue, it may be that introducing separability leads to more powerful models, since the number of points needed to specify separable VKs scales linearly. Currently the models do not support multidimensional inputs, but this could be easily added, requiring the computation of a few extra integrals. For the multiple output model, allowing a shared set of latent functions, with the input to each output's Volterra series being a trainable linear combination, in a similar way to LFMs, is highly likely to improve performance especially for problems with a large number of outputs.

Conclusions We have presented a novel model which uses Gaussian processes to learn the kernels of the Volterra series non-parametrically, allowing for the effective modeling of data with nonlinear properties. We have developed fast and scalable sampling and inference methods for the the model and show its performance on single and multiple output regression problems. Additionally, a modification to the model was presented that achieves significantly better uncertainty quantification than competitors on a challenging benchmark for nonlinear systems identification.

References

[1] M. Alvarez, D. Luengo, and N. D. Lawrence. Latent force models. In *Artificial Intelligence and Statistics*, pages 9–16. PMLR, 2009.

[2] M. Álvarez, W. Ward, and C. Guarnizo. Non-linear process convolutions for multi-output Gaussian processes. pages 1969–1977, 2019.

[3] M. A. Álvarez, L. Rosasco, and N. D. Lawrence. Kernels for vector-valued functions: A review. *Found. Trends Mach. Learn.*, 4(3):195–266, Mar. 2012. ISSN 1935-8237. doi: 10.1561/2200000036. URL `https://doi.org/10.1561/2200000036`.

[4] R. P. Barry and J. M. Ver Hoef. Blackbox kriging: spatial prediction without specifying variogram models. *Journal of Agricultural, Biological and Environmental Statistics*, 1(3):297–322, 1996.

[5] G. Benton, W. J. Maddox, J. Salkey, J. Albinati, and A. G. Wilson. Function-space distributions over kernels. In *Advances in Neural Information Processing Systems*, pages 14965–14976, 2019.

[6] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

[7] W. Bruinsma. *The Generalised Gaussian Process Convolution Model*. MPhil. thesis, University of Cambridge, 2016.

[8] C. Cheng, Z. Peng, W. 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. ISSN 0888-3270. doi: https://doi.org/10.1016/j.ymssp.2016.10.029. URL `http://www.sciencedirect.com/science/article/pii/S0888327016304393`.

[9] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):408–423, 2013.

[10] D. K. Duvenaud, H. Nickisch, and C. Rasmussen. Additive Gaussian processes. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL `https://proceedings.neurips.cc/paper/2011/file/4c5bde74a8f110656874902f07378009-Paper.pdf`.

[11] C. Guarnizo and M. A. Álvarez. Fast kernel approximations for latent force models and convolved multiple-output Gaussian processes. In *Uncertainty in Artificial Intelligence: Proceedings of the Thirty-Fourth Conference (2018)*. AUAI Press, 2018.

[12] J. Hartikainen, M. Seppänen, and S. Särkkä. State-space inference for non-linear latent force models with application to satellite orbit prediction. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 723–730, 2012.

[13] D. Higdon. Space and space-time modeling using process convolution. In *Quantitative methods for current environmental issues*, pages 37–56. Springer, 2002.

[14] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

[16] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith. Dynamic systems identification with Gaussian processes. *Mathematical and Computer Modelling of Dynamical Systems*, 11(4):411–424, 2005. doi: 10.1080/13873950500068567. URL `https://doi.org/10.1080/13873950500068567`.

[17] N. Lawrence, M. Rattray, and M. Titsias. Efficient sampling for Gaussian process inference using control variables. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2009. URL `https://proceedings.neurips.cc/paper/2008/file/5487315b1286f907165907aa8fc96619-Paper.pdf`.

[18] C. L. C. Mattos. *Recurrent Gaussian processes and robust dynamical modeling*. PhD thesis, 2017.

[19] C. L. C. Mattos, Z. Dai, A. Damianou, J. Forth, G. A. Barreto, and N. D. Lawrence. Recurrent Gaussian processes. In H. Larochelle, B. Kingsbury, and S. Bengio, editors, *Proceedings of the International Conference on Learning Representations*, volume 3, Caribe Hotel, San Juan, PR, 2016. URL `http://inverseprobability.com/publications/mattos-recurrent16.html`.

[20] E. A. Morelli and V. Klein. *Aircraft system identification: theory and practice*, volume 2. Sunflyte Enterprises Williamsburg, VA, 2016.

[21] T. V. Nguyen, E. V. Bonilla, et al. Collaborative multi-output Gaussian processes. In *UAI*, pages 643–652. Citeseer, 2014.

[22] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *NIPS*, 3, 2007.

[23] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

[24] M. Schoukens and J. P. Noël. Three benchmarks addressing open challenges in nonlinear system identification. *IFAC-PapersOnLine*, 50(1):446–451, 2017.

[25] A. Svensson and T. B. Schön. A flexible state–space model for learning nonlinear dynamical systems. *Automatica*, 80:189–199, 2017.

[26] A. Svensson, A. Solin, S. Särkkä, and T. Schön. Computationally efficient bayesian learning of Gaussian process state space models. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 213–221, Cadiz, Spain, 09–11 May 2016. PMLR. URL `http://proceedings.mlr.press/v51/svensson16.html`.

[27] M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

[28] M. Titsias and M. Lázaro-Gredilla. Doubly stochastic variational bayes for non-conjugate inference. In *International conference on machine learning*, pages 1971–1979. PMLR, 2014.

[29] F. Tobar, T. D. Bui, and R. E. Turner. Learning stationary time series using Gaussian processes with nonparametric kernels. In *Advances in Neural Information Processing Systems*, pages 3501–3509, 2015.

[30] J. M. Ver Hoef and R. P. Barry. Constructing and fitting models for cokriging and multivariable spatial prediction. *Journal of Statistical Plannig and Inference*, 69:275–294, 1998.

[31] W. Ward, T. Ryder, D. Prangle, and M. Alvarez. Black-box inference for non-linear latent force models. In *International Conference on Artificial Intelligence and Statistics*, pages 3088–3098. PMLR, 2020.

[32] C. Williams, S. Klanke, S. Vijayakumar, and K. Chai. Multi-task Gaussian process learning of robot inverse dynamics. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2009. URL `https://proceedings.neurips.cc/paper/2008/file/15d4e891d784977cacbfcbb00c48f133-Paper.pdf`.

[33] A. Wilson and R. Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International conference on machine learning*, pages 1067–1075, 2013.

[34] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378, 2016.

[35] J. Wilson, V. Borovitskiy, A. Terenin, P. Mostowsky, and M. Deisenroth. Efficiently sampling functions from Gaussian process posteriors. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10292–10302. PMLR, 13–18 Jul 2020. URL `http://proceedings.mlr.press/v119/wilson20a.html`.

[36] K. Worden, R. Barthorpe, E. Cross, N. Dervilis, G. Holmes, G. Manson, and T. Rogers. On evolutionary system identification with applications to nonlinear benchmarks. *Mechanical Systems and Signal Processing*, 112:194–232, 2018. ISSN 0888-3270. doi: https://doi.org/10.1016/j.ymssp.2018.04.001. URL `https://www.sciencedirect.com/science/article/pii/S0888327018301912`.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See sections 5 and 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] No theoretical results
   (b) Did you include complete proofs of all theoretical results? [N/A] No theoretical results
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code provided in supplemental material, links to data in Section 5

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See section 5

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See section 5

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See section 5

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] See section 5

(b) Did you mention the license of the assets? [Yes] All code used freely available under permissive licences, all data publicly available

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Code provided in supplemental material

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] No personal data used

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] No personal data used

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] No human subjects

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] No human subjects

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] No human subjects