ISA Transactions 109 (2021) 141–151

Contents lists available at ScienceDirect

ISA Transactions

journal homepage: www.elsevier.com/locate/isatrans

Research article

Simulation of variational Gaussian process NARX models with GPGPU

Tadej Krivec [a,b,*], Gregor Papa [a,b], Juš Kocijan [a,c]

[a] Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia
[b] Jožef Stefan International Postgraduate School, Jamova cesta 39, Ljubljana, Slovenia
[c] University of Nova Gorica, Vipavska 13, Nova Gorica, Slovenia

ARTICLE INFO

Article history:
Received 18 February 2020
Received in revised form 27 August 2020
Accepted 4 October 2020
Available online 9 October 2020

Keywords:
Gaussian process regression
Nonlinear identification of dynamical systems
Variational Gaussian process
Monte Carlo simulation
General-purpose computing on graphics processing units

ABSTRACT

Gaussian processes (GP) regression is a powerful probabilistic tool for modeling nonlinear dynamical systems. The downside of the method is its cubic computational complexity with respect to the training data that can be partially reduced using pseudo-inputs. The dynamics can be represented with an autoregressive model, which simplifies the training to that of the static case. When simulating an autoregressive model, the uncertainty is propagated through a nonlinear function and the simulation cannot be evaluated in closed-form. This paper combines the variational methods of GP approximations with a nonlinear autoregressive model with exogenous inputs (NARX) to form variational GP (VGP-NARX) models. We show how VGP-NARX models, on average, better approximate a full GP-NARX model than more commonly used GP-NARX (FITC) model on 10 chaotic time-series. The modeling capabilities of VGP-NARX models are compared with the existing approaches on two benchmarks for modeling nonlinear dynamical systems. The advantage of general-purpose computing on graphics processing units (GPGPU) for Monte Carlo simulation on large validation data sets is addressed.

1. Introduction

The idea of Gaussian processes (GPs) for modeling from data was first proposed in the geostatistics field by D. G. Krige [1]. It caught the attention of the machine learning community with the doctoral thesis of C. E. Rasmussen [2]. GPs provide an alternative to a more conventional approach to nonlinear regression by neural networks and can be seen as a generalization of them. Given suitable priors on the weights and one hidden layer with an infinite number of nodes, neural networks converge to a GP [3]. Some of the early work on GPs in machine learning can be found in [4,5], and a more recent monograph in [6]. The advantage of the method is its posterior distribution over functions given the observed data. When simulating with uncertain inputs, the posterior over functions cannot be evaluated in closed-form. Since the model training suffers from cubic computational complexity with respect to the input data, the scalability of the method presents a significant challenge to overcome. Consequently, since the idea of GPs was brought to machine learning, this problem has been largely addressed in the literature.

Sparse approximations methods using pseudo-inputs were proposed to reduce the cubic computational complexity. A unifying view of sparse approximations was presented by J. Quiñonero-Candela and C. E. Rasmussen [7]. A sparse approximation, i.e. fully independent training conditional (FITC), was proposed by E. Snelson and Z. Ghahramani [8], where pseudo-inputs and hyperparameters are jointly inferred through gradient-based optimization. Another approach is to learn the parameters with variational inference, where pseudo-inputs and hyperparameters are determined rigorously by maximizing the lower bound of the true marginal log-likelihood, equivalent to minimizing the Kullback–Leibler (KL) divergence between the true and approximated posterior. The parameters of the optimal variational distribution can be analytically derived resulting in variational free energy (VFE) approximation [9]. The parameters of the variational distribution can also be retained as a model parameter, forming a scalable variational GP (SVGP) [10]. SVGP enables stochastic optimization and gives an unbiased estimation of the lower bound of the true marginal log-likelihood from random subsets of training data to improve the scalability of GPs. Another approach to reducing the computational complexity is with a structured kernel interpolation (SKI) framework, which produces kernel approximation for fast computations. The recent kernel interpolation for scalable structured GP (KISS-GP) [11] uses local cubic interpolation and enables the use of Kronecker and Toeplitz algebra to improve the computational complexity of modeling with GPs.

Probabilistic modeling can be seen as a generalization of the maximum likelihood approach in data-driven models of nonlinear dynamical systems. This approach results in a distribution

* Corresponding author at: Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia.
E-mail addresses: tadej.krivec@ijs.si (T. Krivec), gregor.papa@ijs.si (G. Papa), jus.kocijan@ijs.si (J. Kocijan).

https://doi.org/10.1016/j.isatra.2020.10.011

over the model parameters rather than in a point estimation. Consequently, the probabilistic models are more robust, especially in real-world problems where the data might contain large amounts of noise, outliers and missing values [12–15]. GP regression is a black-box probabilistic approach to modeling nonlinear dynamical systems, where the uncertainty is inferred over the estimated function rather than the model parameters. An overview of the use of GPs for dynamical system identification can be found in [16]. Dynamical systems can be represented with an autoregressive model (such as a Nonlinear AutoRegressive eXogenous (NARX) model), where a GP is used for modeling of the mapping from input to output space [17,18]. For reducing the computational complexity in dynamical modeling, an FITC approximation was generally used in the literature (e.g. [19,20]). The limitation of the FITC approach is that it is prone to overfitting and underestimates the predicted variance [21]. Autoregressive models suffer from error-in-variables but are relatively simple to train and work well in practice. In a simulation, the predicted uncertainty is propagated through a nonlinear function and cannot be evaluated in closed-form [22–24]. A generalization of the autoregressive model is the state-space model, which can separate between the process and observation noise. In a GP state-space model (GP-SSM), the mapping functions are modeled with a GP [25–27]. Using the variational formulation of the GP-SSM, one can obtain a tractable approximated posterior [28]. Another approach to modeling nonlinear dynamical systems is with a family of Bayesian nonparametric models, Recurrent GPs (RGPs), with recurrent GP priors that are able to learn the dynamical patterns from sequential data [29–31]. When dealing with high dimensional input space, a GP latent variable model (GPLVM) was extended for modeling dynamical systems. The family of GPLVM models for modeling dynamical systems are called GP dynamical system (GPDS) models and are presented in [32–35].

Meta-parameters of the autoregressive models are found with resampling methods (e.g. cross-validation), which emphasizes the need for computationally efficient training and simulation. Demanding computational requirements can be reduced with the use of advanced parallel computer architectures. Popular GP libraries [36,37] utilize the parallel computational capabilities of general-purpose computing on graphics processing units (GPGPU), and support simple and flexible scientific computation with automatic differentiation. Other popular GP libraries can be found in [38–40].

This paper aims at modeling nonlinear dynamical systems with a probabilistic approach using GPs. The goal is to provide a model that is simple to train, does not require expert knowledge, estimates the uncertainty of prediction or simulation and can be used on large datasets. In brief, the main contributions of this paper are listed as follows.

(a) We join the variational approximations of GPs with a NARX model in a variational GP-NARX (VGP-NARX) model.
(b) An investigation is conducted on how the VGP-NARX models, on average, better approximate a full GP-NARX model when compared to an FITC approximation of GP-NARX models, which was usually used in the literature to improve the scalability of GP-NARX models.
(c) We extend the existing solutions for training GPs using GPGPU for simulation and show how careful implementation results in significant computational gains even when the nature of the problem is sequential.

The remainder of this paper is organized as follows: In Section 2, we revisit modeling nonlinear dynamical systems using GP-NARX models. The mathematical formulation of the FITC and variational approximations of GP-NARX models are presented. In Section 3, we provide three case studies. Multiple synthetic problems are used to show the modeling capabilities of variational approximations for modeling chaotic time-series. We also evaluate VGP-NARX models on large dynamical datasets and compare the results to other machine learning methods on two benchmarks for dynamical system identification. We investigate the advantages of GPGPU for simulation in large validation datasets. Lastly, we conclude with the final remarks.

2. Modeling nonlinear dynamical systems with Gaussian process regression

GP regression represents a probabilistic model of nonlinear dynamical systems. This section shows how it can be used in a combination with an autoregressive model, and how to simulate such models. Sparse variational approximations are introduced to scale the solution for large datasets. Hardware acceleration with GPGPU is also considered to reduce the computational time of Monte Carlo simulation for autoregressive GP models.

2.1. Autoregressive Gaussian process regression

Modeling dynamical systems is a multiple-stage process including experiment design, the collection of empirical data, data processing, model definition, optimization, parameter estimation, and validation of the model. Consider that the empirical data are given and pre-processed. We model the system by

$$\boldsymbol{y} = f(\boldsymbol{Z}(k), \boldsymbol{\theta}) + \boldsymbol{\epsilon}, \tag{1}$$

where k is a sampling instant, f a nonlinear mapping, $\boldsymbol{\theta}$ is a finite-dimensional parameter vector, and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma_n^2)$ represents a Gaussian noise. Input regressor $\boldsymbol{Z}$ is represented with a NARX model [16] where the rows of the matrix are defined by

$$\boldsymbol{Z}(k) = [y(k-n_p), \ldots, y(k-n_p-n_a), u(k-n_k), \ldots, u(k-n_k-n_b)]. \tag{2}$$

The observed output at k is denoted with $y(k)$ and the excitation sample with $u(k)$. Meta-parameters n_p, n_k, n_a, n_b denote the number of lagged and delayed samples. In general, arbitrary user-defined functions of past data are also permitted. The goal is to find the nonlinear mapping f, model parameters $\boldsymbol{\theta}$, and meta-parameters of the NARX model. Eqs. (1) and (2), without the noise $\boldsymbol{\epsilon}$, can then be used to make future predictions and the simulation can be obtained in the form of a nonlinear output-error (NOE) model [16].

The nonlinear mapping f can be modeled with a GP. Let $\boldsymbol{f} = [f_1, f_2, \ldots, f_n]^T$ denote the vector of latent function values, where $\boldsymbol{y}$ are noisy observations of $\boldsymbol{f}$. The observed data are defined with $\mathcal{D} = (\boldsymbol{Z}, \boldsymbol{y})$. Compared to the formulation in the static case [6], the difference between the dynamical and the static model is only in the definition of the input regressor $\boldsymbol{Z}$. GP regression combined with a NARX input regressor forms a GP-NARX model. A GP prior is defined by a multivariate Gaussian $p(\boldsymbol{f}|\boldsymbol{Z}, \boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K})$, where the entries to the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{K}$ are completely specified by its mean function $m(\boldsymbol{z}_i)$ and covariance function $k(\boldsymbol{z}_i, \boldsymbol{z}_j)$

$$m(\boldsymbol{z}_i) = \mathbb{E}[f(\boldsymbol{z}_i)], \tag{3a}$$

$$k(\boldsymbol{z}_i, \boldsymbol{z}_j) = \mathbb{E}[(f(\boldsymbol{z}_i) - m(\boldsymbol{z}_i))(f(\boldsymbol{z}_j) - m(\boldsymbol{z}_j))]. \tag{3b}$$

Mean function is commonly selected as $m(\boldsymbol{z}_i) = 0$. The choice of a covariance function is more important since it defines our prior belief of the modeled function. It can be represented as a function that generates a symmetric, semi-positive, and square matrix. A popular choice is the squared exponential function

$$k(\boldsymbol{z}_i, \boldsymbol{z}_j) = \sigma_f^2 e^{-\frac{1}{2l^2}\|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2}, \tag{4}$$

where l represents a lengthscale parameter and σ_f is a scaling factor. Other covariance functions can be found in [16]. Automatic relevance determination property can be used for weighting the input regressor columns. Covariance function with an automatic relevance determination property is defined by

$$k(\boldsymbol{z}_i, \boldsymbol{z}_j) = \sigma_f^2 e^{-\frac{1}{2}(\boldsymbol{z}_i - \boldsymbol{z}_j)^T \boldsymbol{\Lambda}^{-1}(\boldsymbol{z}_i - \boldsymbol{z}_j)}, \tag{5}$$

where $\boldsymbol{\Lambda}^{-1} = \text{diag}([l_1^{-2}, \ldots, l_D^{-2}])$, and D is the number of dimensions in $\boldsymbol{Z}$.

Probabilistic modeling allows us to infer the posterior probability over the vector of latent function values, where a GP prior is transformed through the Bayes theorem to the posterior

$$p(\boldsymbol{f}|\boldsymbol{y}, \boldsymbol{Z}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{Z}, \boldsymbol{\theta}) p(\boldsymbol{f}|\boldsymbol{Z}, \boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{Z}, \boldsymbol{\theta})}. \tag{6}$$

For convenience, we will omit the conditional dependence on $\boldsymbol{Z}$ and $\boldsymbol{\theta}$ in future notation. In the case of a squared exponential function with an automatic relevance determination property, the vector of model parameters is defined with $\boldsymbol{\theta} = [\sigma_n, \sigma_f, l_1, \ldots, l_D]$, and can be found with the maximization of marginal log-likelihood with respect to the parameters. Marginal log-likelihood is given by

$$\log p(\boldsymbol{y}) = -\frac{1}{2}\log(|\boldsymbol{K}|) - \frac{1}{2}\boldsymbol{y}^T \boldsymbol{K}^{-1}\boldsymbol{y} - \frac{N}{2}\log(2\pi). \tag{7}$$

Prediction $\boldsymbol{f}_*$ at an unseen input $\boldsymbol{z}_*$ is given by

$$p(\boldsymbol{f}_*|\boldsymbol{y}) = \int p(\boldsymbol{f}, \boldsymbol{f}_*|\boldsymbol{y}) df, \tag{8}$$

where the vector of latent values $\boldsymbol{f}$ is marginalized out. The posterior distribution has a closed-form solution resulting in a multivariate Gaussian distribution

$$p(\boldsymbol{f}_*|\boldsymbol{y}) \sim \mathcal{N}(\boldsymbol{K}_{*f}[\boldsymbol{K}_{ff} + \sigma_n^2 \boldsymbol{I}]^{-1}\boldsymbol{y}, \boldsymbol{K}_{**} - \boldsymbol{K}_{*f}[\boldsymbol{K}_{ff} + \sigma_n^2 \boldsymbol{I}]^{-1}\boldsymbol{K}_{f*}), \tag{9}$$

where $\boldsymbol{K}_{ff}, \boldsymbol{K}_{f*}, \boldsymbol{K}_{**}$ are the covariance matrices between training data, training and test data, and test data respectively. The problem of simulation reduces to predicting with uncertain inputs where an unseen input vector $\boldsymbol{z}_*$ is stochastic. This results in propagation of the Gaussian distribution through a nonlinear mapping and evaluating the expression

$$p(\boldsymbol{f}_*|\boldsymbol{y}) = \iint p(\boldsymbol{f}, \boldsymbol{f}_*|\boldsymbol{y}, \boldsymbol{z}_*) p(\boldsymbol{z}_*) df dz_*, \tag{10}$$

which does not have a closed-form solution. We evaluate the integral with a numerical approximation by a simple Monte Carlo approach. We obtain the output in the form of a Gaussian mixture model

$$p(\boldsymbol{f}_*|\boldsymbol{y}) \approx \frac{1}{S}\sum_{i=1}^{S} \int p(\boldsymbol{f}, \boldsymbol{f}_*|\boldsymbol{y}) df, \tag{11}$$

where S is the number of samples taken from the uncertain vector of inputs $\boldsymbol{z}_*$.

2.2. Sparse autoregressive Gaussian process regression

The downside of GP regression is its cubic computational complexity, which originates from calculating the inverse of $\boldsymbol{K}_{ff}$. GP regression in a full GP-NARX model can be approximated to reduce computational complexity. Consider m pseudo-inputs $\boldsymbol{z}_m$ and the corresponding vector of latent values $\boldsymbol{u} = [u_1, \ldots, u_m]$ drawn from the same joint distribution as $\boldsymbol{f}$ and $\boldsymbol{f}_*$ [7]. Latent vectors $\boldsymbol{f}$ and $\boldsymbol{f}_*$ are assumed to be conditionally independent given $\boldsymbol{u}$. Their joint prior is approximated by

$$p(\boldsymbol{f}, \boldsymbol{f}_*, \boldsymbol{u}) \cong \int p(\boldsymbol{f}|\boldsymbol{u}) p(\boldsymbol{f}_*|\boldsymbol{u}) p(\boldsymbol{u}) d\boldsymbol{u}. \tag{12}$$

The joint distribution is fully specified with the given conditionals

$$p(\boldsymbol{f}|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{u}, \boldsymbol{K}_{nn} - \boldsymbol{Q}_{nn}), \tag{13a}$$

$$p(\boldsymbol{f}_*|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\boldsymbol{u}, \boldsymbol{K}_{**} - \boldsymbol{Q}_{**}), \tag{13b}$$

where $\boldsymbol{Q}_{ab} = \boldsymbol{K}_{am}\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{mb}$. To obtain the predictive distribution, $\boldsymbol{f}$ and $\boldsymbol{u}$ are marginalized out of the joint distribution to obtain

$$p(\boldsymbol{f}_*|\boldsymbol{y}) = \iint p(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{y}) p(\boldsymbol{f}_*|\boldsymbol{u}) p(\boldsymbol{u}) d\boldsymbol{f}\, d\boldsymbol{u}. \tag{14}$$

The fully independent training conditional (FITC) approximation method considers an exact $p(\boldsymbol{f}_*|\boldsymbol{u})$, but an approximated $p(\boldsymbol{f}|\boldsymbol{u})$. This results in a covariance matrix $\boldsymbol{K}$ with exact values on the diagonal while the others are approximated. Model parameters and pseudo-inputs are jointly inferred with gradient-based optimization [8]. This approximation reduces the computational complexity to $\mathcal{O}(nm^2)$, where n and m represent the number of rows in $\boldsymbol{Z}$ and the number of pseudo-inputs respectively. In combination with a NARX model, this approximation forms the GP-NARX (FITC) model.

The downside of the approximation is that the distance between the true posterior and the approximated posterior is never directly minimized. Sparse approximations that use variational learning rigorously define the distance between the true posterior and approximated posterior. Variational learning lower bounds the marginal log-likelihood where the posterior distribution of a variational model is defined by

$$q(\boldsymbol{f}, \boldsymbol{f}_*, \boldsymbol{u}) \cong \int p(\boldsymbol{f}|\boldsymbol{u}) p(\boldsymbol{f}_*|\boldsymbol{u}) q(\boldsymbol{u}) d\boldsymbol{u}, \tag{15}$$

where $q(\boldsymbol{u}) \sim \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}, \boldsymbol{\Lambda})$ is chosen to be a free variational distribution. The parameters of the variational model are obtained by lower bounding the marginal likelihood which is equivalent to minimizing the KL divergence between variational distribution $q(\boldsymbol{f}, \boldsymbol{f}_*, \boldsymbol{u})$ and exact distribution $p(\boldsymbol{f}, \boldsymbol{f}_*, \boldsymbol{u}|\boldsymbol{y})$. This consequently pushes $\boldsymbol{u}$ to become a sufficient statistic for $\boldsymbol{f}$ such that $p(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{y}) \approx p(\boldsymbol{f}|\boldsymbol{u})$. The lower bound [10] is given by

$$l(\boldsymbol{z}_m, \boldsymbol{\theta}) = \mathbb{E}_{q(\boldsymbol{f})}\Big[\log p(\boldsymbol{y}|\boldsymbol{f})\Big] - \text{KL}\Big[q(\boldsymbol{u}) \parallel p(\boldsymbol{u})\Big]. \tag{16}$$

To find the optimal parameters of the free variational distribution $q(\boldsymbol{u})$, the bound can be maximized with respect to the free variational distribution [9] to obtain $q(\boldsymbol{u}) \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{\Lambda})$, where

$$\boldsymbol{m} = \boldsymbol{\Lambda}^{-1}\boldsymbol{K}_{uu}^{-1}\boldsymbol{K}_{uf}\boldsymbol{y}\sigma_n^{-2}, \tag{17a}$$

$$\boldsymbol{\Lambda} = \boldsymbol{K}_{uu}^{-1} + \boldsymbol{K}_{uu}^{-1}\boldsymbol{K}_{uf}\boldsymbol{K}_{fu}\boldsymbol{K}_{uu}^{-1}\sigma_n^{-2}. \tag{17b}$$

The lower bound is given by

$$l(\boldsymbol{z}_m, \boldsymbol{\theta}) = \log\left[\mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{Q}_{ff} + \sigma_n^2 \boldsymbol{I})\right] - \frac{1}{2\sigma_n^2} tr(\boldsymbol{K}_{ff} - \boldsymbol{K}_{fu}\boldsymbol{K}_{uu}^{-1}\boldsymbol{K}_{uf}), \tag{18}$$

where $\boldsymbol{Q}_{ff} = \boldsymbol{K}_{fu}\boldsymbol{K}_{uu}^{-1}\boldsymbol{K}_{uf}$. The details of the derivation are given in a technical report by Titsias [41]. This bound is computed in $\mathcal{O}(nm^2)$ and allows the model parameters and pseudo-inputs to be jointly learned. This variational approximation in combination with a NARX model forms the VGP-NARX variational free energy (VFE) model. A scalable variational Gaussian process (SVGP) [10], which retains the variational distribution $q(\boldsymbol{u})$, was derived by Hensman et al. The likelihood in Eq. (16) factors to $p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{N} p(y_i|f_i)$, and the bound can be rewritten

$$l(\boldsymbol{z}_m, \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\Lambda}) = \sum_{n=1}^{N} \mathbb{E}_{q(f_n)}\Big[\log p(y_n|f_n)\Big] - \text{KL}\Big[q(\boldsymbol{u}) \parallel p(\boldsymbol{u})\Big]. \tag{19}$$

This bound can be computed in $\mathcal{O}(m^3)$. Unbiased estimation of the lower bound can be obtained in batches of training data where the model parameters and pseudo-inputs are jointly learned in an

online fashion. This variational approximation in a combination with a NARX model forms a VGP-NARX (SVGP) model. Prediction in both VGP-NARX models is obtained with marginalization over $\boldsymbol{f}$ and $\boldsymbol{u}$

$$p(\boldsymbol{f}_*|\boldsymbol{y}) = \iint p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{f}_*|\boldsymbol{u})q(\boldsymbol{u})d\boldsymbol{f}\,d\boldsymbol{u} = \int p(\boldsymbol{f}_*|\boldsymbol{u})q(\boldsymbol{u})d\boldsymbol{u}, \quad (20)$$

where $\int p(\boldsymbol{f}|\boldsymbol{u})d\boldsymbol{f} = 1$. To marginalize over $\boldsymbol{u}$, the integral following the last equality in Eq. (20) has to be computed. Distributions $p(\boldsymbol{f}_*|\boldsymbol{u})$ and $q(\boldsymbol{u})$ are both Gaussian and fully specified. Consequently, the integral has a closed-form solution and the simulation can be obtained in the form of an NOE model.

2.3. Autoregressive GP simulation with general-purpose computing on graphics processing units

GPUs are capable of processing a stream of data and effectively apply a single task to different data. They are optimized for parallel, whereas central processing units (CPUs) are optimized for sequential processing. Simulation with Monte Carlo sampling can be accelerated with GPGPU, since the samples of the simulated response are independent and can be evaluated in parallel. The simple Monte Carlo algorithm from [22] was, in our investigation, extended where we replace the outer loop with a large test matrix. Samples of simulated responses are represented in matrix form and can be evaluated on existing frameworks for GPGPU. The computational implementation of the predictive distribution for full GP can be found in [16], and for sparse and variational GPs in [36,42].

We trained the models with GPflow [36], a GP library that wraps around Tensorflow [43]. Tensorflow allows the end-users to perform scientific computations flexibly with the support of GPU acceleration. We implemented a custom simulation algorithm in Tensorflow based on prediction algorithms from GPflow. We used the following system specifications:

- CPU: Intel(R) Core(TM) i7-8700K CPU 3.70 GHz,
- GPU: Quadro P4000 8 GB GDDR5 [GP104GL],
- RAM: 2 × 16 GB DIMM DDR4 3200 MHz,
- OS: Ubuntu 18.04.2 LTS,
- Software: GPflow 1.3.0, Tensorflow 1.12.0.

3. Experimental evaluation

In the case of large datasets, modeling with the full GP-NARX is not feasible in practice (computation in reasonable time) due to the cubic computational complexity with respect to the dataset size. For that reason, we firstly show the differences between the simulation of GP-NARX (FITC) and the proposed VGP-NARX models compared to the full GP-NARX on 10 chaotic time-series problems in Section 3.2. The synthetic nature of the data allows us to choose a flexible dataset size. Consequently, we can train a full GP-NARX model, which is used as the ground truth for comparison. Secondly, we compare the VGP-NARX models with existing machine learning approaches on Silverbox and Bouc–Wen benchmarks in Sections 3.3 and 3.4 . They provide large training and validation datasets which allow us to evaluate the modeling capabilities of VGP-NARX models for large data and show the advantages of GPGPU for simulation.

3.1. Model validation

In Section 3.2 we compare the simulated response of the VGP-NARX models to a GP-NARX (FITC) model, taking the simulation of a full GP-NARX model as the ground truth. We compare the vector of Monte Carlo samples at sampling instant k based on the first Wasserstein distance, also known as the earth mover's distance [44], which does not impose a parametric structure on the simulated distributions. Intuitively, it represents the minimal cost of transforming one distribution into another. If U and V are the respective cumulative distribution functions of random variables u and v, the distance can be defined by

$$W_1(u, v) = \int_{-\infty}^{\infty} |U - V|. \quad (21)$$

In our case, u represents the simulated distribution of the full GP-NARX model and v represents the approximated distribution at the time step k. We also evaluate the model capabilities of the approximated GP-NARX models and compare them to the past work on the benchmarks in Sections 3.3 and 3.4 . We use the root-mean-square error (RMSE) [6] as the measure, usually used with the Silverbox and Bouc–Wen benchmarks. We also consider the mean standardized log loss (MSLL) [6], which takes into account not only the mean of the model simulation but an entire distribution.

3.2. Chaotic time-series

Chaotic time-series belong to a class of nonlinear dynamical systems, that is, the systems that evolve over time. Chaotic (in this sense) means that the smallest change in the initial conditions produces a very different outcome, even when the governing equations are known exactly. When the governing equations are nonlinear, this is known as a deterministic chaos [45].

In this section, we validated the approximated VGP-NARX models for modeling chaotic time-series. We selected 10 chaotic problems with different properties, e.g. multi-output, continuous, and discrete. The description of the modeled chaotic time-series can be found in [45]. Since the difficulty of the aforementioned problems is specific to each dataset, we validate the models for prediction, which considerably simplifies the analysis for multiple datasets. Firstly, we standardized the datasets and injected a relatively high Gaussian noise to the time-series with a zero mean and a standard deviation of 0.5. For GP-NARX (FITC) and VGP-NARX models, we selected 300 pseudo-inputs, which were initialized at random. We used a whole training dataset as the batch in VGP-NARX (SVGP). We estimated the simulated response with a Monte Carlo simulation of a 100 samples. By 100 samples, we mean that the response is computed in parallel with only one execution, but can be seen equivalently as 100 independent executions. The order n_a of the NARX model was selected as 5 for all datasets, except for the Mackey–Glass time-series, which was 20. Here it is important to choose an order that is higher than optimal, since the ARD property can weight the columns of the input regressor and can effectively remove the unnecessary predictors. Secondly, we ran the experiments with 10 random restarts. In each random restart, we validated the model with a 5-fold cross-validation. The hyper-parameters of the model were optimized using Adam [46]. Lastly, we computed the mean and the variance of the RMSE over all random restarts and different folds in cross-validation.

3.2.1. Discussion

The results of the prediction experiments on the chaotic time-series are presented in Table 1. The table shows the means and the respective 95% confidence interval of the RMSE over random restarts. We can see that, in the case of relatively noisy measurements, the variational methods perform better in regards to RMSE. The results from a full GP-NARX model are shown for comparison, where we can observe that VGP-NARX models perform similarly to a full GP-NARX.

Table 1
Predictive RMSE and the corresponding 95% confidence interval for modeling chaotic time-series.

	GP-NARX (FITC)	VGP-NARX (VFE)	VGP-NARX (SVGP)	full GP-NARX
Hénon map	0.272 ± 0.037	0.205 ± 0.025	**0.195± 0.026**	0.198 ± 0.026
Ikeda map	0.383 ± 0.043	0.357 ± 0.036	**0.355 ± 0.036**	0.354 ± 0.037
Logistic map	0.824 ± 0.097	0.748 ± 0.077	**0.744± 0.077**	0.746 ± 0.077
Lorenz attractor	0.264 ± 0.040	**0.203 ± 0.039**	0.206 ± 0.039	0.205 ± 0.041
Mackey–Glass	0.392 ± 0.060	**0.363 ± 0.054**	0.365 ± 0.057	0.365 ± 0.056
Quadratic map	0.730 ± 0.075	**0.674 ± 0.066**	0.676 ± 0.065	0.675 ± 0.066
Rössler attractor	0.254 ± 0.056	**0.168 ± 0.086**	0.175 ± 0.128	0.171 ± 0.102
Gauss map	0.267 ± 0.048	**0.225 ± 0.039**	**0.225± 0.039**	0.226 ± 0.039
Tent map	0.719 ± 0.099	**0.661 ± 0.078**	**0.661± 0.073**	0.661 ± 0.074
Driven pendulum	0.438 ± 0.056	**0.405 ± 0.048**	**0.405± 0.048**	0.407 ± 0.047

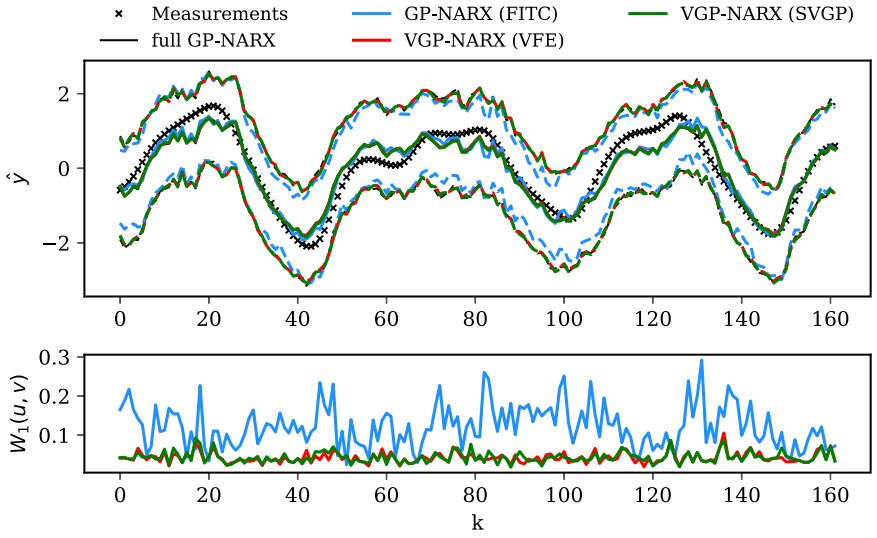

Fig. 1. The top figure represents the 20 step-ahead simulation for the Mackey–Glass time-series initialized from consecutive starting points. Dashed lines show the 2 standard deviation intervals. The bottom figure shows the first Wasserstein distance between the approximated GP-NARX models and the full GP-NARX model at sampling instant k.

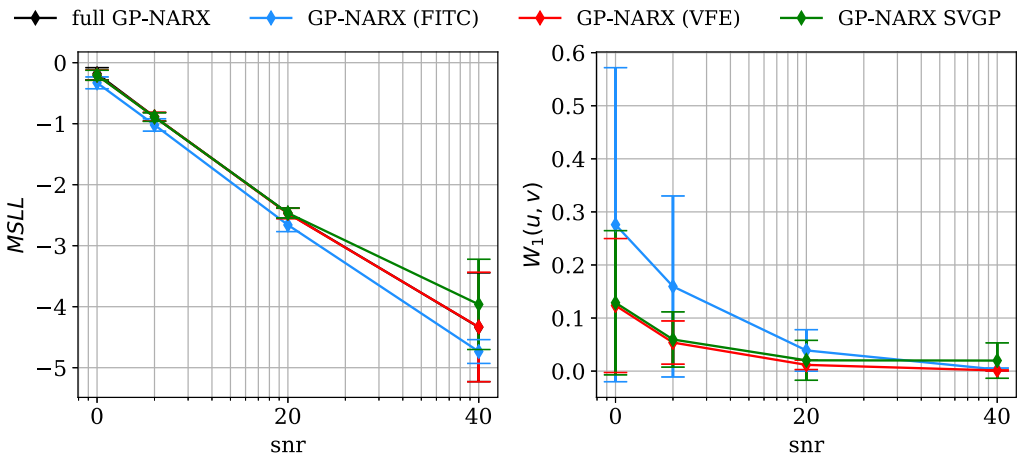

Fig. 2. The means and a 95% confidence interval for MSLL and the first Wasserstein distance for 20 step-ahead prediction on the Mackey–Glass time-series with different signal-to-noise ratio.

A more detailed experiment for multi-step ahead prediction was realized on the Mackey–Glass time-series. Fig. 1 shows the 20 step-ahead simulated response with the corresponding interval of two standard deviations and the first Wasserstein distance between the approximated GP-NARX models and the full GP-NARX. We observe that VGP-NARX (VFE) and VGP-NARX (SVGP) models retrieve a better simulation response than a GP-NARX (FITC) model when compared to a full GP-NARX. This can be best seen with a Wasserstein distance per prediction step on the bottom figure on Fig. 1. Fig. 2 shows the MSLL and the first Wasserstein distance for a 20 step-ahead prediction for various signal-to-noise ratios which were averaged over 10 random restarts and 5-fold cross validation. VGP-NARX models, on average, better model the posterior distribution when compared to a full GP-NARX.

GP-NARX (FITC) can underestimate the simulated variance, as previously shown in [21], and can result in a better MSLL as depicted on Fig. 2, although it performs worse in regards to RMSE. One of the conclusions of the aforementioned article is

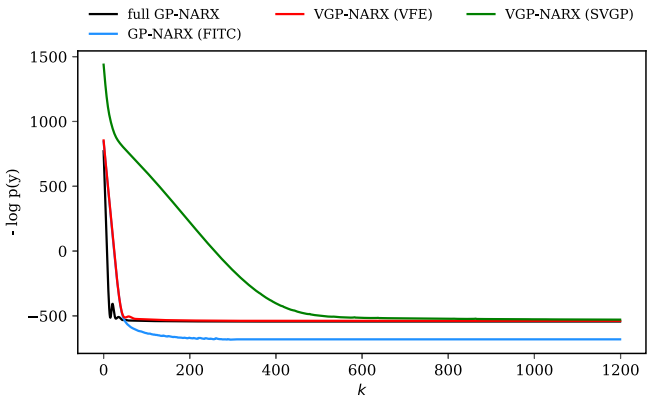

Fig. 3. Marginal log-likelihood for optimization on the Mackey–Glass time-series with different GP-NARX models. The marginal log-likelihood values after optimization are 543.69, 681.17, 538.30, 535.36 for full GP-NARX, GP-NARX (FITC), VGP-NARX (VFE) and VGP-NARX (SVGP), respectively.

that VFE approximation always improves with the addition of new pseudo-inputs, maximizing the given resources, whereas FITC approximation may ignore them. An interesting observation can be seen on Fig. 3, which shows that the optimization of a GP-NARX (FITC) model results in a higher marginal log-likelihood compared to the full GP-NARX. Marginal log-likelihood, in general, penalizes overly complex models, since they spread their probability mass widely and can, therefore, be used for model selection. In this case, one would be inclined to choose GP-NARX (FITC) over a full GP-NARX, whereas, on the validation dataset, the full GP-NARX model generalized better. The GP-NARX (FITC) model is susceptible to overfitting in this regard, whereas the VGP-NARX (VFE) model contains a regularization term in the definition of the lower bound and can penalize model complexity. Here we emphasize our point; VGP-NARX models better model the expected value of the simulation (RMSE) when subjected to a relatively high noise, better retrieve the simulated distribution of a full GP-NARX (Wasserstein distance), but may result in a worse MSLL. We observed that the VGP-NARX (SVGP) model performs similarly to VGP-NARX (VFE) and can reduce the computational time for one step of optimization, but can suffer from sensitivity to the parameters of the optimization. Consequently, this can result in no gain in computational time required (more iterations to converge) and sub-optimal hyperparameters, even if the optimization is computed in batches of data. With careful initialization and selection of optimization parameters, we observe that VGP-NARX (SVGP), similarly as VGP-NARX (VFE), converges to the marginal log-likelihood of the full GP-NARX as presented on Fig. 3.

3.3. Silverbox benchmark

Silverbox benchmark represents the 2nd order linear time-invariant system with the 3rd degree polynomial static nonlinearity around it in feedback. It is a real-world problem, where the data were collected from an electric circuit. The system theoretically obeys the equation

$$m\frac{d^2y(t)}{dt^2} - d\frac{dy(t)}{dt} + ay(t) + by(t)^3 = u(t). \tag{22}$$

We selected the training data from samples 40,586 to 127,410, and the validation data as the first 40,495 samples [47]. The system was excited with a white Gaussian noise sequence filtered by a 9th order discrete-time Butterworth filter with a cut-off frequency of 200 Hz for validation data. The amplitude was varied linearly over the interval from zero to its maximal value. Training data were obtained with excitation of 10 successive realizations of a random odd multi-sine signal. More detailed information about the benchmark can be found in [48]. The meta-parameters of the NARX model were selected as $n_k = 0, n_b = 10, n_p = 1, n_a = 10$ with an additional cubic expansion of the output. The sum of a linear and squared exponential function with automatic relevance detection was selected for the covariance function. We used 1000 pseudo-inputs initialized with the k-means clustering algorithm as suggested in [21]. We used 1000 points for batch size in VGP-NARX (SVGP) model training. Adam optimizer was used for marginal log-likelihood optimization.

Table 2
RMSE [mV] and MSLL of free-run simulation on the validation dataset for different modeling approaches on Silverbox benchmark.

Approach	Reference	RMSE	MSLL
NARX MLP	Ljung et al. [47]	0.30	–
NARX LSSVM	Espinoza et al. [49]	0.32	–
Local Linear State-Space	Verdult [50]	1.3	–
Best Linear Approximation	Paduart et al. [51]	13.7	–
PNLSS	Paduart et al. [51]	0.26	–
Nonlinear State-Space	Marconato et al. [52]	0.34	–
Wiener–Schetzen	Tiels and Schoukens [53]	9.2	–
Extended fuzzy logic	Sabahi and Akbarzadeh-T [54]	9.1	–
TCN	Andersson et al. [55]	4.88	–
GP-NARX (FITC)	This paper	0.9	−2.35
VGP-NARX (VFE)	This paper	1.1	−2.44
VGP-NARX (SVGP)	This paper	1.7	−2.81

3.3.1. Discussion

Table 2 presents our results in reference to the existing machine learning algorithms for system identification on the Silverbox benchmark for RMSE and MSLL metrics. MLP stands for multilayer perceptron, LSSVM for least squares support vector machines, PNLSS for polynomial nonlinear state-space, and TCN for the temporal convolutional network. The results are compared to the original studies and conducted with the experiment conditions proposed in [48]. As we were not able to compute the Wasserstein distance for the Silverbox benchmark (dataset size is too big to train a full GP-NARX model), RMSE and MSLL metrics allowed us to evaluate the model capabilities and compare them to the existing work on the benchmarks.

We obtained comparable results to other approaches, where the state-space model showed the best results. This is expected since it is able to separate between the process and the observation noise. This approach requires a more careful analysis which can be time-consuming and (usually) does not translate well between different problems. A simple approach that can be used by non-experts, NARX multilayer perceptron (MLP) also achieves a better RMSE compared to our GP-NARX models, is computationally less demanding, but fails to provide a confidence estimation of the simulation. VGP-NARX (FITC) performs best among the GP-NARX models for RMSE and VGP-NARX (SVGP) for MSLL. VGP-NARX (SVGP) is more sensitive to the parameters of the optimization and converges slowly as depicted on Fig. 4. The results do not provide statistically significant differences between the GP-NARX models. The reason for this also lies in a carefully defined experiment, where the training data were collected in multiple periods, leading to multiple noisy measurements of the same data point. This reduces the influence of noise on the modeling and can favor the flexible models on this benchmark since the risk of overfitting is reduced.

The simulation was realized in Tensorflow, where we constructed a computational graph for prediction. We precomputed all the operations that do not depend on the iteration step of the

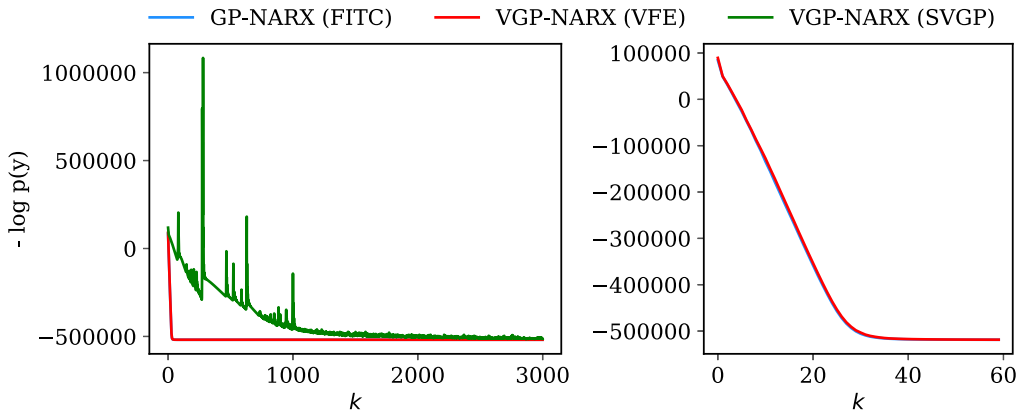

Fig. 4. Marginal log-likelihood for optimization on the Silverbox benchmark with approximated GP-NARX models. The figure on the left shows the comparison between the VGP-NARX and GP-NARX models, where there is no visible difference between VGP-NARX (VFE) and GP-NARX (FITC). The figure on the right shows a more detailed comparison between GP-NARX (FITC) and VGP-NARX (VFE), which shows their marginal log-likelihood is nearly identical in this case.

Table 3
RMSE [$\times 10^{-5}$ m] and MSLL of the free-run simulation on the validation datasets for different approaches when modeling the Bouc–Wen benchmark.

Approach	Reference	RMSE †	RMSE⋆	MSLL †	MSLL⋆
Best Linear Approximation	Schoukens and Scheiwe [56]	15.1	16.6	–	–
Volterra Feedback	Schoukens and Scheiwe [56]	8.76	6.39	–	–
PNLSS	Esfahani et al. [57]	1.87	1.20	–	–
Decoupled PNLSS	Esfahani et al. [57]	1.90	1.40	–	–
LMN	Belz et al. [58]	16.36	13.8	–	–
D.P. NARX	Westwick et al. [59]	5.36	1.67	–	–
LMSSN	Max Schüssler [60]	2.66	1.76	–	–
GP-NARX (FITC)	This paper	1.60	0.97	−3.40	−3.45
VGP-NARX (VFE)	This paper	1.72	0.88	−3.40	−3.39
VGP-NARX (SVGP)	This paper	2.17	1.04	−3.32	−3.33

simulation. We then reused the constructed computational graph for the prediction in each iteration. In the case of GPGPU, Tensorflow allowed us to compile and execute the prediction graph on a GPU in a simple fashion. Fig. 5 shows the computational (wall) time required to obtain the Monte Carlo simulation on a CPU and GPU for different sizes of Monte Carlo samples and different numbers of pseudo-inputs. As the number of pseudo-inputs and Monte Carlo samples increases, we observe that the GPGPU requires less computational time in comparison to the CPU. In the case of simulation with a small number of pseudo-inputs and Monte Carlo samples, the GPU is not utilized well since the computational requirements are not significant. In that case, we observe that the simulation can be computed faster with a CPU, since the CPU itself can partially parallelize the computations without the overhead of data transfer.

3.4. Bouc-Wen benchmark

Bouc–Wen benchmark represents the vibrations of a single degree of freedom Bouc–Wen system, representing a hysteretic process. Hysteresis is a dynamical nonlinearity, where the input–output loop persists when the input frequency approaches zero [61]. The problem is mathematically defined by

$$m_L \frac{d^2y(t)}{dt^2} + r(y(t), \frac{dy(t)}{dt}) + z(y(t), \frac{dy(t)}{dt}) = u(t), \tag{23a}$$

$$r(y(t), \frac{dy(t)}{dt}) = k_L y(t) + c_L \frac{dy(t)}{dt}, \tag{23b}$$

$$z(y(t), \frac{dy(t)}{dt}) = \alpha \frac{dy(t)}{dt} - \beta(\gamma|\frac{dy(t)}{dt} \parallel z|^{\upsilon-1} z + \delta \frac{dy(t)}{dt}|z|^{\upsilon}). \tag{23c}$$

Training data consist of a multi-sine excitation signal, where 5 periods of the input and output signal were recorded with 40,960 samples. Validation data consist of two samples, one excited with a multi-sine signal, the other with a sine-sweep signal, with the lengths of 8,192 and 153,000 samples, respectively. For a detailed explanation on model parameters and how the data were generated see [62]. We selected the meta-parameters of the GP-NARX model as $n_k = 0, n_b = 10, n_p = 1, n_a = 10$, with an expansion of absolute values of the output. A sum of a linear and squared exponential function with automatic relevance detection property was selected for the covariance function; and, the number of pseudo-inputs and batch size were selected exactly as in the Silverbox case study.

3.4.1. Discussion

Table 3 presents our results in reference to the existing machine learning algorithms for system identification on the Bouc–Wen benchmark for RMSE and MSLL metrics. PNLSS stands for polynomial nonlinear state-space, LMN for local model networks, D.P. NARX for decoupled polynomial NARX, and LMSSN for local model state-space networks. The results are compared to the original studies and conducted with the experiment conditions proposed in [62]. RMSE† and MSLL† represent the measures for the multi-sine validation dataset, whereas RMSE⋆ and MSLL⋆ represent the measures for the sine-sweep validation dataset.

Similarly, as in the Silverbox benchmark, we were not able to compute the Wasserstein distance. We obtained state-of-the-art results (to our knowledge) compared to the past work and provided a confidence estimation (although at the cost of demanding computation). We only considered the papers which use the RMSE as a metric for model validation. GP-NARX models performed similarly with no statistically significant differences between them. VGP-NARX (SVGP) again proved more sensitive

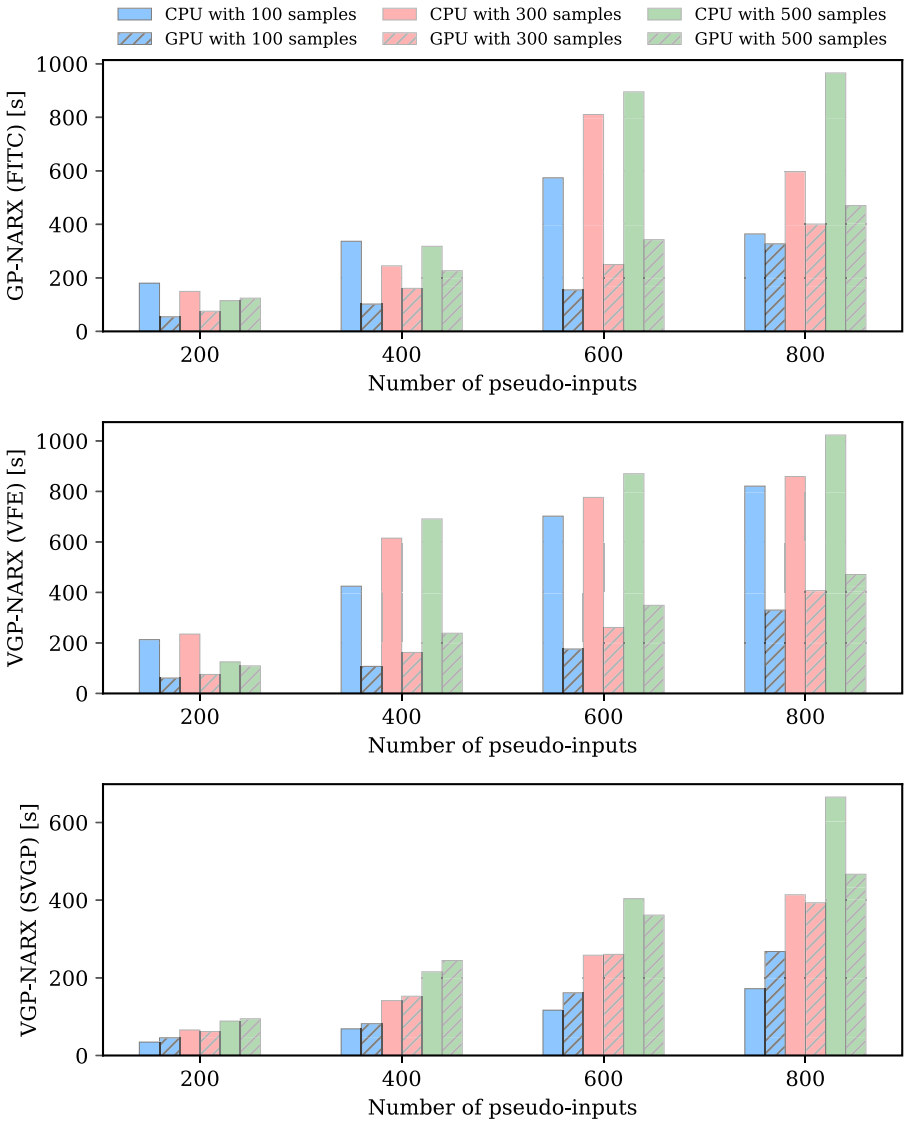

Fig. 5. Wall time for simulation on GPUs and CPUs for a different number of pseudo-inputs and a different number of Monte Carlo samples on the validation dataset of the Silverbox benchmark. The *x*-axis represents the number of pseudo-inputs and the *y*-axis represents the simulation wall time. Ordinary colored bars represent the wall time of the corresponding simulation on a CPU with a different number of Monte Carlo samples. Dashed colored bars represent the aforementioned simulation wall time on a GPU.

to the parameters of the optimization with slow convergence as depicted on Fig. 6.

Simulation was realized in Tensorflow as in the Silverbox case study. Fig. 7 shows the computational time required to obtain the Monte Carlo simulation on a CPU and GPU for a different size of Monte Carlo samples and a different number of pseudo-inputs. Similarly, as with the Silverbox benchmark, as the number of pseudo-inputs and the number of Monte Carlo samples increases, we observe that GPGPU requires (significantly) less computational time in comparison to CPU.

4. Conclusion

In this paper, we introduced VGP-NARX models, a probabilistic approach to modeling nonlinear dynamical systems. They improve the scalability of existing GP-NARX models for training and simulation on large datasets. On average, VGP-NARX models retrieve a better-simulated response than the existing GP-NARX (FITC) approximation for smaller datasets when compared to a full GP-NARX; but, they do not provide statistically significant modeling improvements over GP-NARX (FITC) as the data grow larger and are measured in multiple periods. VGP-NARX (SVGP), with its sensitivity to optimization parameters in model training, may result in a sub-optimal solution and converge slowly. Approximated GP-NARX models achieve comparable results to other approaches on nonlinear dynamical benchmarks and additionally provide uncertainty estimation of the simulation on the cost of increased computational complexity. The advantage of the VGP-NARX models is that they are as simple to use as GP-NARX (FITC), but have better statistical properties and can scale better (in the case of GP-NARX (SVGP)). At this point, we want to emphasize that the simple structure is also a limitation of a GP-NARX model. For optimal performance, one has to use a more complex approach, such as the state-space model, which can separate between the process and the observation noise.

The problem of uncertainty propagation in the simulation was addressed with Monte Carlo integration. The computational time of Monte Carlo simulation was reduced with the use of GPGPU

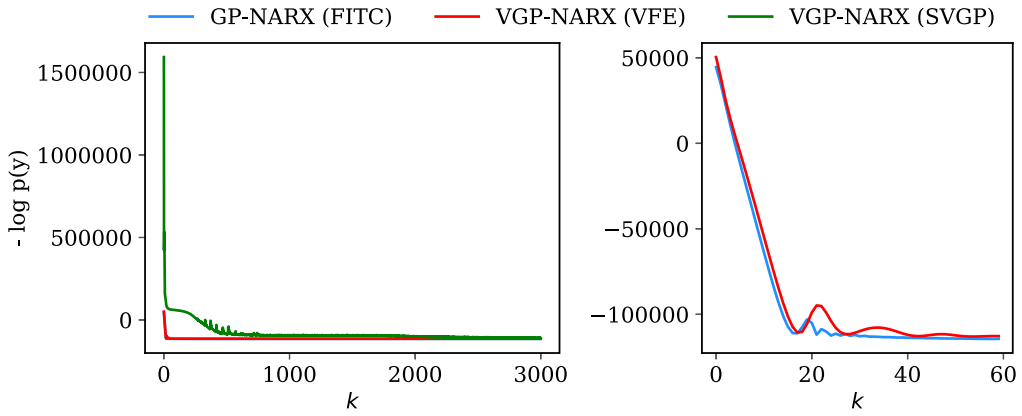

Fig. 6. Marginal log-likelihood for optimization on the Bouc–Wen benchmark with approximated GP-NARX models. The figure on the left shows the comparison between the VGP-NARX and GP-NARX models, where there is no visible difference between VGP-NARX (VFE) and GP-NARX (FITC). The figure on the right shows a more detailed comparison between GP-NARX (FITC) and VGP-NARX (VFE).

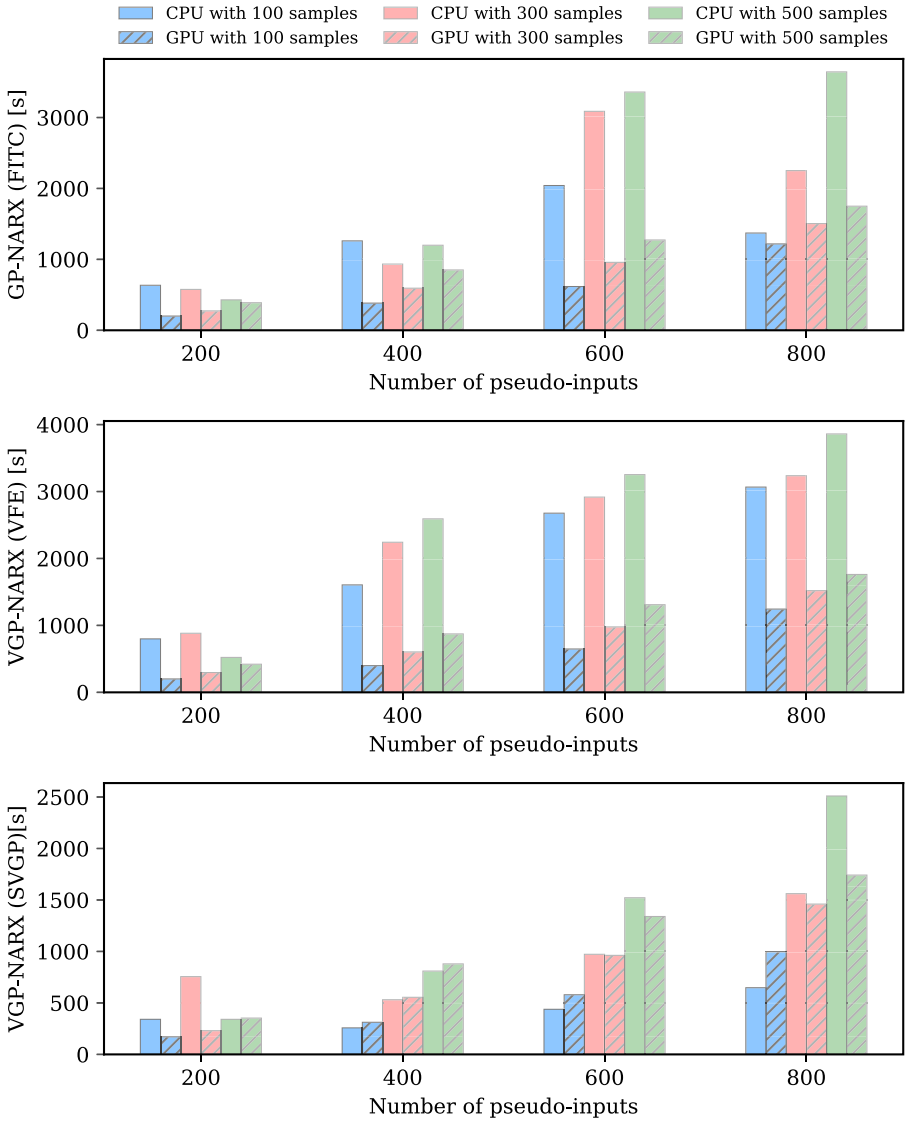

Fig. 7. Wall time for simulation on GPUs and CPUs for a different number of pseudo-inputs and a different number of Monte Carlo samples on the sine-sweep validation dataset of the Bouc–Wen benchmark. The *x*-axis represents the number of pseudo-inputs and the *y*-axis represents the simulation wall time. Ordinary colored bars represent the wall time of the corresponding simulation on a CPU with a different number of Monte Carlo samples. Dashed colored bars represent the aforementioned simulation wall time on a GPU.

when the number of pseudo-inputs and the number of Monte Carlo samples increased. These computationally efficient algorithms can be used in a simple fashion by non-experts with no additional work to improve the scalability of simulation for large validation datasets. VGP-NARX models with GPGPU accelerated training and simulation can, therefore, be used by non-experts to find solutions relatively quickly and at different scales.

An interesting challenge for future work in modeling nonlinear dynamical systems with GP-NARX models is with a more flexible approach such as deep GPs, which introduce a functional composite of GPs similarly as in deep neural networks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Slovenian Research Agency [Ph.D. grant for Tadej Krivec, and core funding grant numbers P2-0001, P2-0098].

References

[1] Krige DG. A statistical approach to some basic mine valuation problems on the Witwatersrand. J South Afr Inst Min Metall 1951;52(6):119–39.
[2] Rasmussen CE. Evaluation of gaussian processes and other methods for non-linear regression. University of Toronto; 1999.
[3] Neal RM. Bayesian learning for neural networks vol. 118. Springer Science & Business Media; 2012.
[4] Neal RM. Monte Carlo Implementation of gaussian process models for bayesian regression and classification. 1997, Personal communication. arXiv:physics/9701026.
[5] Gibbs MN. Bayesian gaussian processes for regression and classification [Ph.D. thesis], Citeseer; 1998.
[6] Williams CK, Rasmussen CE. Gaussian processes for machine learning, no. 3. MA: MIT press Cambridge; 2006.
[7] Quiñonero-Candela J, Rasmussen CE. A unifying view of sparse approximate gaussian process regression. J Mach Learn Res 2005;6(Dec):1939–59.
[8] Snelson E, Ghahramani Z. Sparse gaussian processes using pseudo-inputs. In: Advances in neural information processing systems. 2006, p. 1257–64.
[9] Titsias M. Variational learning of inducing variables in sparse gaussian processes. In: Artificial intelligence and statistics. 2009, p. 567–74.
[10] Hensman J, Matthews AG, Ghahramani Z. Scalable variational Gaussian process classification. J Mach Learn Res 2015. arXiv:1411.2005.
[11] Wilson A, Nickisch H. Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). In: International conference on machine learning, 2015, p. 1775–84.
[12] Bottegal G, Aravkin AY, Hjalmarsson H, Pillonetto G. Outlier robust system identification: A Bayesian Kernel-based approach. IFAC Proc Vol 2014;47(3):1073–8.
[13] Yang X, Yin S. Variational Bayesian inference for FIR models with randomly missing measurements. IEEE Trans Ind Electron 2016;64(5):4217–25.
[14] Yang X, Liu X, Yin S. Robust identification of nonlinear systems with missing observations: The case of state-space model structure. IEEE Trans Ind Inf 2018;15(5):2763–74.
[15] Yang X, Liu X, Li Z. Multimodel approach to robust identification of multiple-input single-output nonlinear time-delay systems. IEEE Trans Ind Inf 2019;16(4):2413–22.
[16] Kocijan J. Modelling and control of dynamic systems using gaussian process models. Springer; 2016.
[17] Kocijan J, Girard A, Banko B, Murray-Smith R. Dynamic systems identification with gaussian processes. Math Comput Model Dyn Syst 2005;11(4):411–24.
[18] Worden K, Manson G, Cross EJ. On gaussian process NARX models and their higher-order frequency response functions. In: Solving computationally expensive engineering problems. Springer; 2014, p. 315–35.
[19] Frigola R. Bayesian Time series learning with gaussian processes. [Ph.D. thesis], University of Cambridge; 2015.
[20] Kocijan J, Perne M, Grašic B, Božnar MZ, Mlakar P. Sparse and hybrid modelling of relative humidity: the Krško Basin case study. CAAI Trans Intell Technol 2020;5(1):42–8.
[21] Bauer M, van der Wilk M, Rasmussen CE. Understanding probabilistic sparse gaussian process approximations. In: Advances in neural information processing systems. 2016, p. 1533–41.
[22] Girard A. Approximate methods for propagation of uncertainty with gaussian process models. [Ph.D. thesis], Citeseer; 2004.
[23] Girard A, Rasmussen CE, Candela JQ, Murray-Smith R. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In: Advances in neural information processing systems. 2003, p. 545–52.
[24] Candela JQ, Girard A, Larsen J, Rasmussen CE. Propagation of uncertainty in bayesian kernel models-application to multiple-step ahead forecasting. In: 2003 IEEE international conference on acoustics, speech, and signal processin, vol. 2. IEEE; 2003, p. II–701.
[25] Deisenroth MP, Turner RD, Huber MF, Hanebeck UD, Rasmussen CE. Robust filtering and smoothing with gaussian processes. IEEE Trans Automat Control 2011;57(7):1865–71.
[26] Sarkka S, Solin A, Hartikainen J. Spatio-temporal learning via infinite-dimensional bayesian filtering and smoothing: A look at gaussian process regression through kalman filtering. IEEE Signal Process Mag 2013;30(4):51–61.
[27] Frigola R, Lindsten F, Schön TB, Rasmussen CE. Bayesian Inference and learning in gaussian process state-space models with particle MCMC. In: Advances in neural information processing systems. 2013, p. 3156–64.
[28] Frigola R, Chen Y, Rasmussen CE. Variational gaussian process state-space models. In: Advances in neural information processing systems. 2014, p. 3680–8.
[29] Mattos CLC, Dai Z, Damianou A, Forth J, Barreto GA, Lawrence ND. Recurrent Gaussian processes. In: 4th International conference on learning representations. 2016, arXiv:1511.06644.
[30] Mattos CLC, Dai Z, Damianou A, Barreto GA, Lawrence ND. Deep recurrent gaussian processes for outlier-robust system identification. J Process Control 2017;60:82–94.
[31] Mattos CLC, Barreto GA. A stochastic variational framework for recurrent gaussian processes models. Neural Netw 2019;112:54–72.
[32] Lawrence ND, Moore AJ. Hierarchical Gaussian process latent variable models. In: Proceedings of the 24th international conference on machine learning, 2007, p. 481–488.
[33] Wang J, Hertzmann A, Fleet DJ. Gaussian process dynamical models. In: Advances in neural information processing systems. 2006, p. 1441–8.
[34] Wang JM, Fleet DJ, Hertzmann A. Gaussian process dynamical models for human motion. IEEE Trans Pattern Anal Mach Intell 2007;30(2):283–98.
[35] Damianou A, Titsias MK, Lawrence ND. Variational gaussian process dynamical systems. In: Advances in neural information processing systems. 2011, p. 2510–8.
[36] Matthews AGdG, van der Wilk M, Nickson T, Fujii K, Boukouvalas A, Le'on-Villagr'a P, et al. GPflow: A gaussian process library using tensorflow. J Mach Learn Res 2017;18(40):1–6.
[37] Gardner J, Pleiss G, Weinberger KQ, Bindel D, Wilson AG. Gpytorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration. In: Advances in neural information processing systems. 2018, p. 7576–86.
[38] Rasmussen CE, Nickisch H. Gaussian processes for machine learning (GPML) toolbox. J Mach Learn Res 2010;11(Nov):3011–5.
[39] Vanhatalo J, Riihimäki J, Hartikainen J, Jylänki P, Tolvanen V, Vehtari A. GPstuff: Bayesian modeling with gaussian processes. J Mach Learn Res 2013;14(Apr):1175–9.
[40] GPY: A gaussian process framework in python. 2012, http://github.com/SheffieldML/GPy.
[41] Titsias MK. Variational model selection for sparse gaussian process regression. Report, UK: University of Manchester; 2009.
[42] Hensman J. Derivation of SGPR equations. 2019, https://nbviewer.jupyter.org/github/GPflow/GPflow/blob/develop/doc/source/notebooks/theory/SGPR_notes.ipynb. [accessed 12 December 2019].
[43] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. Tensorflow: Large-scale machine learning on heterogeneous systems. 2015, URL: http://tensorflow.org/.
[44] Ramdas A, Trillos N, Cuturi M. On Wasserstein two-sample testing and related families of nonparametric tests. Entropy 2017;19(2):47.
[45] Sprott JC, Sprott JC. Chaos and time-series analysis, vol. 69. Citeseer; 2003.
[46] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2014, Personal communication. arXiv:1412.6980.
[47] Ljung L, Zhang Q, Lindskog P, Juditski A. Estimation of grey box and black box models for non-linear circuit data. IFAC Proc Vol 2004;37(13):399–404.
[48] Wigren T, Schoukens J. Three free data sets for development and benchmarking in nonlinear system identification. In: 2013 European control conference. IEEE; 2013, p. 2933–8.
[49] Espinoza M, Pelckmans K, Hoegaerts L, Suykens JA, De Moor B. A comparative study of LS-SVM's applied to the silver box identification problem. IFAC Proc Vol 2004;37(13):369–74.
[50] Verdult V. Identification of local linear state-space models: the silver-box case study. IFAC Proc Vol 2004;37(13):393–8.

[51] Paduart J, Lauwers L, Swevers J, Smolders K, Schoukens J, Pintelon R. Identification of nonlinear systems using polynomial nonlinear state space models. Automatica 2010;46(4):647–56.
[52] Marconato A, Sjöberg J, Suykens J, Schoukens J. Identification of the silverbox benchmark using nonlinear state-space models. IFAC Proc Vol 2012;45(16):632–7.
[53] Tiels K, Schoukens J. Wiener system identification with generalized orthonormal basis functions. Automatica 2014;50(12):3147–54.
[54] Sabahi F, Akbarzadeh-T MR. Extended fuzzy logic: Sets and systems. IEEE Trans Fuzzy Syst 2015;24(3):530–43.
[55] Andersson C, Ribeiro AH, Tiels K, Wahlström N, Schön TB. Deep convolutional networks in system identification. In: 2019 IEEE 58th conference on decision and control. IEEE; 2019, p. 3670–6.
[56] Schoukens M, Scheiwe FG. Modeling Nonlinear Systems using a Volterra Feedback Model, In: Workshop on nonlinear system identification benchmarks, 2016.
[57] Esfahani AF, Dreesen P, Tiels K, Noël J-P, Schoukens J. Polynomial state-space model decoupling for the identification of hysteretic systems. IFAC-PapersOnLine 2017;50(1):458–63.
[58] Belz J, Münker T, Heinz TO, Kampmann G, Nelles O. Automatic modeling with local model networks for benchmark processes. IFAC-PapersOnLine 2017;50(1):470–5.
[59] Westwick DT, Hollander G, Karami K, Schoukens J. Using decoupling methods to reduce polynomial NARX models. IFAC-PapersOnLine 2018;51(15):796–801.
[60] Schüssler M, Münker T, Nelles O. Local model networks for the identification of nonlinear state space models. In: 2019 IEEE 58th conference on decision and control. IEEE; 2019, p. 6437–42.
[61] Noël J, Schoukens M. Hysteretic Benchmark with a Dynamic Nonlinearity. In: Workshop on nonlinear system identification benchmarks, 2016, p. 7–14.
[62] Noël J-P, Esfahani AF, Kerschen G, Schoukens J. A nonlinear state-space approach to hysteresis identification. Mech Syst Signal Process 2017;84:171–84.