# Gaussian process based recursive system identification

**Jakub Prüher and Miroslav Šimandl**
University of West Bohemia, Czech Republic
E-mail: jacobnzw@ntis.zcu.cz, simandl@ntis.zcu.cz

**Abstract.** This paper is concerned with the problem of recursive system identification using non-parametric Gaussian process model. Non-linear stochastic system in consideration is affine in control and given in the input-output form. The use of recursive Gaussian process algorithm for non-linear system identification is proposed to alleviate the computational burden of full Gaussian process. The problem of an online hyper-parameter estimation is handled using proposed ad-hoc procedure. The approach to system identification using recursive Gaussian process is compared with full Gaussian process in terms of model error and uncertainty as well as computational demands. Using Monte Carlo simulations it is shown, that the use of recursive Gaussian process with an ad-hoc learning procedure offers converging estimates of hyper-parameters and constant computational demands.

## 1. Introduction

Throughout the sciences the task of building models from observed data is of fundamental importance. In the area of control this task is known as system identification. The way of obtaining the model is in direct contrast to mathematical modelling, where the knowledge from wide variety of disciplines, such as physics, biology, economy and engineering, is exploited. In control engineering, the need for system identification arises whenever there is uncertainty in the knowledge of the system central to the task.

Identification methods can be divided according to the way they process data into: *batch* methods and *recursive* methods. By the batch identification we will understand a method that processes all the measured data at once to produce the system model. On the contrary, recursive identification algorithms process the measurements as they become available during the course of the system operation and gradually refine the system model. These algorithms are at the core of adaptive control mechanisms.

Models of the system can be *parametric* or *non-parametric*. Parametric models typically assume fixed structure with finite number of free parameters, which need to be estimated. Well known model structures include for example ARX and ARMAX models for linear systems. For non-linear systems various types of neural networks can be named. On the other hand, non-parametric models assume no fixed structure and, from the viewpoint of parametric models, their number of 'parameters' can be potentially infinite. Spectral and correlation approaches fall into this category along with increasingly popular Gaussian process regression models.

Contrary to the parametric models, such as the neural networks, Gaussian process (GP) models do not possess a fixed structure and thus are much more flexible. Neal showed [1], that a neural network, where the number of hidden units goes to infinity, is equivalent to GP model with certain covariance function. Specifying a GP can be effectively thought of as putting a prior distribution over functions themselves, where functions can be informally regarded as vectors of function values of infinite dimension. Various communities pay increasing attention to the GP models due to their effectiveness in modelling of highly non-linear phenomena. They are attractive, also because they combine tractable Bayesian inference with


Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

non-parametric nature of the model. GP models have been utilised in many applications. These include non-linear filtering [2–5], model predictive control [6], non-linear system identification [7], time series forecasting [8], reinforcement learning [9, 10], numerical quadrature [11] and many others.

Great advantage of parametric models is that they lend themselves nicely to the possibility of devising a recursive estimation algorithm. As GPs are non-parametric models, the situation is much more complicated. Typically, it is necessary to consider some kind of GP approximation to counteract the problem of increasing data size. Särkkä [12] proposed a solution to this problem based on Kalman filtering and smoothing applied to the state-space representation of the GP with specific covariance functions. This solution, however, works for one-dimensional inputs only and is thus inappropriate for our purposes. Very frequently used alternative is the Sparse Online Gaussian Process (SOGP) [13–15]. SOGP however does not solve the problem of online learning of hyper-parameters.

Our goal is the investigation of the recently proposed *Recursive Gaussian Process* (RGP) algorithm [16] for the purpose of non-linear system identification. Ultimately, we envision the future incorporation of RGP system identification algorithm into the functional dual adaptive control framework [17, 18].

The rest of the paper will present a description of the problem formulation in section 2, followed by an RGP based non-linear system identification in section 3. In section 4, a simple numerical illustration comparing the full GP and the recursive GP in non-linear system identification is presented. Finally, section 5 concludes the paper.

## 2. Problem formulation

In this work, we are dealing with the non-linear stochastic discrete time-invariant system given in an input-output representation

$$\mathcal{S}: \qquad y_{k+1} = f\left(\boldsymbol{x}_k^a\right) + g\left(\boldsymbol{x}_k^a\right) u_k + e_{k+1}, \tag{1}$$

where $f, g : \mathbb{R}^{n_y+n_u+1} \to \mathbb{R}$ are unknown non-linear functions, $\boldsymbol{x}_k^a = \left[y_k, \ldots, y_{k-n_y}, u_{k-1}, \ldots, u_{k-n_u}\right]^\top \in \mathbb{R}^{n_y+n_u+1}$ is the state vector, $u_k$ and $y_k$ are input and output signals respectively at discrete time instants $k \in 0, 1, \ldots,$ N-1 and $e_k$ is the noise term. In addition, the following assumptions are considered:

**A1:** The non-linear functions are smooth, i.e. $f\left(\boldsymbol{x}_k^a\right), g\left(\boldsymbol{x}_k^a\right) \in C^\infty$.

**A2:** The structural parameters $n_y$ and $n_u$ of the system are known.

**A3:** $e_k \in \mathbb{R}$ is a white zero-mean Gaussian noise with known variance $\sigma_e^2$.

The unknown non-linear system is modelled by a non-parametric GP model as an alternative to the parametric neural network model [19]. The structural information about affinity of the system input $u_k$ is tackled by employing a specific form of covariance function. We postpone discussion of the specifics to the section 3, where modelling of the non-linear system with the help of GPs is described in detail in system identification setting. For now, the focus will be given to the GP regression in it's original formulation to illustrate the shortcomings, when used for sequential data processing. This approach is termed *full GP* (FGP) in section 4.

To motivate the need for recursive GP regression algorithm, the Gaussian process regression problem will be described. *Gaussian process regression* refers to the estimation of an underlying function $h : \mathbb{R}^D \to \mathbb{R}$ of a noisy static mapping

$$y_i = h(\boldsymbol{x}_i) + \varepsilon_i \tag{2}$$

from a given set of input vectors $\boldsymbol{X} = \left\{\boldsymbol{x}_i \in \mathbb{R}^D\right\}_{i=1}^n$ and corresponding set of observations $\boldsymbol{y} = \left\{y_i \in \mathbb{R}\right\}_{i=1}^n$, where $\{\varepsilon_i \in \mathbb{R}\}_{i=1}^n$ is the noise sequence. The key concept here is that the estimated unknown function $h$ is modelled using a GP prior

$$h \sim \mathcal{GP}(m, k) \tag{3}$$

which, akin to the Gaussian distribution, is fully determined by its first two moments defined as

$$m(\boldsymbol{x}_p \mid \boldsymbol{\theta}) \triangleq \mathrm{E}_h\Big[h(\boldsymbol{x}_p) \mid \boldsymbol{\theta}\Big], \tag{4}$$

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q \mid \boldsymbol{\theta}) \triangleq \mathrm{E}_h\Big[\Big(h(\boldsymbol{x}_p) - m(\boldsymbol{x}_p)\Big)\Big(h(\boldsymbol{x}_q) - m(\boldsymbol{x}_q)\Big) \mid \boldsymbol{\theta}\Big], \tag{5}$$

where $\mathrm{E}_h$ is expectation over function values $h$, $m(\boldsymbol{x}_p \mid \boldsymbol{\theta}) : \mathbb{R}^D \to \mathbb{R}$ is a *mean function* and $k(\boldsymbol{x}_p,\ \boldsymbol{x}_q \mid \boldsymbol{\theta}) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is a *covariance function* for all $\boldsymbol{x}_p, \boldsymbol{x}_q \in \mathbb{R}^D$. The covariance function is parametrised by a vector of free *hyper-parameters* $\boldsymbol{\theta}$. To keep the notation uncluttered, dependence of $m(\cdot)$ and $k(\cdot)$ on $\boldsymbol{\theta}$ will be omitted from now on. Designers have an incredible variety of covariance functions at their disposal each expressing different assumptions about the modelled phenomena. For comprehensive account of the wide range of covariance functions and their relationship to other models reader should consult [20]. Compared to parametric regression models, where the assumptions on the modelled function are typically expressed in the form of fixed model structure, the assumptions of non-parametric GP models, expressed by the choice of covariance function, are much less restrictive.

With a GP prior in hand, Bayesian techniques of inference can further be employed to refine the GP prior $p(h|\boldsymbol{\theta})$ by incorporating the evidence from the set of input vectors $\boldsymbol{X}$ and observations $\boldsymbol{y}$ to produce the GP posterior

$$p(h|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{y}|h, \boldsymbol{X}, \boldsymbol{\theta})\, p(h|\boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})}, \tag{6}$$

where the GP posterior $h|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta} \sim \mathcal{GP}(m^+, k^+)$ is fully described by a posteriori mean function $m^+$ and covariance function $k^+$ given by

$$m^+(\boldsymbol{x}) = m(\boldsymbol{x}) + k(\boldsymbol{x}, \boldsymbol{X})k(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{y}, \tag{7}$$

$$k^+(\boldsymbol{x}, \boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - k(\boldsymbol{x}, \boldsymbol{X})k(\boldsymbol{X}, \boldsymbol{X})^{-1}k(\boldsymbol{X}, \boldsymbol{x}), \tag{8}$$

where $k(\boldsymbol{x}, \boldsymbol{X}) = \Big[k(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}, \boldsymbol{x}_n)\Big]$, $k(\boldsymbol{X}, \boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{X})^\top$ and $k(\boldsymbol{X}, \boldsymbol{X})$ is an $n \times n$ prior covariance matrix of elements $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. In other words, for any fixed $\boldsymbol{x} \in \mathbb{R}^D$ the equations (7) and (8) define the Gaussian posterior predictive distribution over the values of function $h$ at this point.

Looking at equations (7), (8) one can notice the fact that the prediction is calculated using all available data, which is the very feature of non-parametric GP modelling. In offline settings, where all the data are available beforehand, this results in the necessity to invert large matrices. Moreover, in situations, where the real-time data processing is required, the size of the matrix $k(\boldsymbol{X}, \boldsymbol{X})$ grows, which leads to ever increasing computational demands for matrix inversion in each time step as more and more data are being processed. To alleviate this problem, we propose the use of RGP regression algorithm for system identification task, which is described in the next section.

## 3. Recursive Gaussian process regression

This section covers the identification of a non-linear stochastic system by means of the RGP regression algorithm. RGP algorithm is briefly outlined along with the ad-hoc procedure for hyper-parameter learning. Last section covers the application of RGP to the system identification task.

### *3.1. Recursive Gaussian Process algorithm*

Recursive Gaussian process regression algorithm (RGP) was proposed by Huber [16] as a solution to the problem of GP regression, where the data arrive sequentially. The main idea employed in RGP is to use a predefined set of *basis vectors* as a sparsification element [21], from which the predictions are computed. The number of basis vectors $s \ll N$ (where $N$ is the number of data points) is fixed throughout the operation of the algorithm, which enables to keep the computational demands in check. On the other hand, this acts as an approximation to the full GP approach characterized by the equations (7) and (8) as the prediction is no longer a function of all the previously seen data.

Keeping with the notation of [16], let $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_s]$ denote the matrix of locations of the basis vectors and $\boldsymbol{g} = g(\boldsymbol{X})$ the corresponding vector of values of the unknown latent function $g$. Because $g$ is considered to be a GP, the distribution $p_0\left(\boldsymbol{g}\right) = \mathcal{N}\left(\boldsymbol{g};\, \boldsymbol{\mu}_0^g, \boldsymbol{C}_0^g\right)$ at initial time step $k = 0$ is Gaussian. For any time step $k > 0$, new set of $nk$ observations $\boldsymbol{y}_k = \left[y_{k,1}, y_{k,2}, \ldots, y_{k,nk}\right]$ at input locations $\boldsymbol{X}_k = \left[\boldsymbol{x}_{k,1}, \boldsymbol{x}_{k,2}, \ldots, \boldsymbol{x}_{k,nk}\right]$ is processed. The goal is then to calculate posterior distribution

$$p_k\left(\boldsymbol{g}\,|\,\boldsymbol{y}_{1:k}\right) = \mathcal{N}\left(\boldsymbol{g};\, \boldsymbol{\mu}_k^g, \boldsymbol{C}_k^g\right) \tag{9}$$

at time $k$, where $\boldsymbol{y}_{1:k} = \left[\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots \boldsymbol{y}_k\right]$, by combining the new observations $\boldsymbol{y}_k$ with the distribution

$$p_{k-1}\left(\boldsymbol{g}\,|\,\boldsymbol{y}_{1:k-1}\right) = \mathcal{N}\left(\boldsymbol{g};\, \boldsymbol{\mu}_{k-1}^g, \boldsymbol{C}_{k-1}^g\right) \tag{10}$$

from the previous time step, which functions as a prior in Bayesian setting. Calculation of the posterior $p_k\left(\boldsymbol{g}\,|\,\boldsymbol{y}_{1:k}\right)$ is performed in two steps:

**Inference:** calculating the joint prior $p_{k-1}\left(\boldsymbol{g},\, \boldsymbol{g}_k\,|\,\boldsymbol{y}_{1:k-1}\right)$ given the prior (10),
**Update:** updating the joint prior with new observations and integrating out $\boldsymbol{g}_k$.

In summary, RGP algorithm operates by means of the following set of equations

$$\boldsymbol{J}_k = k(\boldsymbol{X}_k, \boldsymbol{X})\, k(\boldsymbol{X}, \boldsymbol{X})^{-1}, \tag{11}$$

$$\boldsymbol{\mu}_k^p = m(\boldsymbol{X}_k) + \boldsymbol{J}_k \left(\boldsymbol{\mu}_{k-1}^g - m(\boldsymbol{X})\right), \tag{12}$$

$$\boldsymbol{C}_k^p = k(\boldsymbol{X}_k, \boldsymbol{X}_k) + \boldsymbol{J}_k \left(\boldsymbol{C}_{k-1}^g - k(\boldsymbol{X}, \boldsymbol{X})\right) \boldsymbol{J}_k^\top, \tag{13}$$

$$\boldsymbol{G}_k = \boldsymbol{C}_{k-1}^g \boldsymbol{J}_k \left(\boldsymbol{C}_k^p + \sigma^2 \boldsymbol{I}\right)^{-1}, \tag{14}$$

$$\boldsymbol{\mu}_k^g = \boldsymbol{\mu}_{k-1}^g + \boldsymbol{G}_k \left(\boldsymbol{y}_k - \boldsymbol{\mu}_k^p\right), \tag{15}$$

$$\boldsymbol{C}_k^g = \boldsymbol{C}_{k-1}^g - \boldsymbol{G}_k \boldsymbol{J}_k \boldsymbol{C}_{k-1}^g. \tag{16}$$

The equations (11) – (13) perform the inference step and serve for function value predictions $\boldsymbol{g}_k$ at locations $\boldsymbol{X}_k$ using previous estimate $\boldsymbol{\mu}_{k-1}^g$. Equation (13) provides covariance matrix of the predictions. The equations (14) – (16) perform the update step, where the first two moments of the posterior (9) are updated using the last observations $\boldsymbol{y}_k$. For detailed explanations and derivation of the equations (11) – (16) reader is referred to the original article [16].

### *3.2. Ad-hoc online learning procedure*

RGP algorithm is able to perform GP regression recursively. However, the problem of online hyper-parameter learning still stands. One possibility would be utilization of the RGP* algorithm [22], which is able to learn the hyper-parameters in recursive manner in addition to recursively performing the GP regression. Nonetheless, RGP* algorithm proved to be challenging for implementation. For this reason we resort to tackling the online hyper-parameter learning problem by the following ad-hoc procedure.

Typically, to find an optimal setting of the hyper-parameters $\boldsymbol{\theta}$, the logarithm of marginal likelihood

$$\log p\left(\boldsymbol{Y}_k\,|\,\boldsymbol{X}_k, \boldsymbol{\theta}\right) = -\frac{1}{2}\left[\boldsymbol{Y}_k^\top \left(\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_e^2 \boldsymbol{I}\right)^{-1} \boldsymbol{Y}_k + \log\left|\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_e^2 \boldsymbol{I}\right| + n\log(2\pi)\right] \tag{17}$$

is maximised, where $\boldsymbol{K}_{\boldsymbol{\theta}} = k(\boldsymbol{X}_k, \boldsymbol{X}_k)$. The advantage of the marginal likelihood maximisation is that it produces optimal values that do not result in the model over-fitting or under-fitting. The disadvantage is that the optimisation problem is non-linear which makes finding of the global maxima practically

impossible. It is helpful to point out, that the objective function (17) is a function of $\boldsymbol{\theta}$, where the data (noisy observations $\boldsymbol{Y}_k$ at locations $\boldsymbol{X}_k$) are given. In sequential framework these quantities represent all the currently available data up to a certain time step.

The idea of the proposed ad-hoc approach lies in the treatment of the current estimate of the latent function values $\boldsymbol{\mu}_k^g$ as perfect observations. In practical terms this entails modification of the marginal likelihood objective function (17), whereby the noisy observations $\boldsymbol{Y}_k$ are replaced with the perfect observations $\boldsymbol{\mu}_k^g$ at basis vector locations $\boldsymbol{X}$. Thus we obtain the following objective function

$$\log p\left(\boldsymbol{\mu}_k^g \,|\, \boldsymbol{X}, \boldsymbol{\theta}\right) = -\frac{1}{2}\left[\left(\boldsymbol{\mu}_k^g\right)^\top \boldsymbol{K}_{\boldsymbol{\theta}}^{-1} \boldsymbol{\mu}_k^g + \log\left|\boldsymbol{K}_{\boldsymbol{\theta}}\right| + s\log(2\pi)\right], \tag{18}$$

which is maximized in every time step using iterative optimization solver, with the initial starting point set to the hyper-parameter estimate obtained in the previous time step. With maximum number of solver iterations fixed, the computational burden associated with hyper-parameter estimation is kept bounded. The notation $\boldsymbol{K}_{\boldsymbol{\theta}}$ emphasizes the fact that the prior covariance matrix is a function of hyper-parameters $\boldsymbol{\theta}$.

### *3.3. Non-linear system identification with RGP*

The unknown non-linear stochastic system to be identified (1) is modelled by the GP prior. This can be written in the following way

$$\mathcal{M}: \qquad \hat{y}_{k+1} \sim \mathcal{GP}\left(m, k\right), \tag{19}$$

where $\hat{y}_{k+1}$ is the prediction of system output at the next time step. Let $\boldsymbol{X}^a = \left[\boldsymbol{x}_1^a, \ldots, \boldsymbol{x}_s^a\right]$ be a $D \times s$ matrix, where each column is a vector of regressors of dimension $D = ny + nu + 1$ and let $\boldsymbol{X} = \left[\left(\boldsymbol{X}^a\right)^\top, \boldsymbol{U}^\top\right]^\top$, where $\boldsymbol{U} = [u_1, \ldots, u_s]$ is the last row of $(D+1) \times s$ basis vector matrix $\boldsymbol{X}$. The first two moments of the GP prior were chosen as

$$m\left(\boldsymbol{x}_p\right) \equiv 0, \tag{20}$$

$$k\left(\boldsymbol{x}_p, \boldsymbol{x}_q\right) = k_f\left(\boldsymbol{x}_p^a, \boldsymbol{x}_q^a\right) + k_g\left(\boldsymbol{x}_p^a, \boldsymbol{x}_q^a\right) u_p u_q + k_n\left(\boldsymbol{x}_p^a, \boldsymbol{x}_q^a\right), \tag{21}$$

where

$$k_f\left(\boldsymbol{x}_p^a, \boldsymbol{x}_q^a\right) = s_f^2 \exp\left(-\frac{1}{2}\left(\boldsymbol{x}_p^a - \boldsymbol{x}_q^a\right)^\top \boldsymbol{\Lambda}_f^{-1}\left(\boldsymbol{x}_p^a - \boldsymbol{x}_q^a\right)\right), \tag{22}$$

$$k_g\left(\boldsymbol{x}_p^a, \boldsymbol{x}_q^a\right) = s_g^2 \exp\left(-\frac{1}{2}\left(\boldsymbol{x}_p^a - \boldsymbol{x}_q^a\right)^\top \boldsymbol{\Lambda}_g^{-1}\left(\boldsymbol{x}_p^a - \boldsymbol{x}_q^a\right)\right), \tag{23}$$

$$k_n\left(\boldsymbol{x}_p^a, \boldsymbol{x}_q^a\right) = \sigma_e^2 \delta_{pq}. \tag{24}$$

The quantities $s_f$, $s_g$ are vertical lengthscales, $\boldsymbol{\Lambda}_f = \operatorname{diag}\left(\lambda_{f1}^2, \ldots, \lambda_{fD}^2\right)$ are horizontal lengthscales and $\sigma_e^2$ is the system output noise variance. The symbol $\delta_{pq}$ stands for Kronecker delta. The above mentioned quantities are GP hyper-parameters forming a vector $\boldsymbol{\theta} = \left[s_f^2, \lambda_{f1}, \ldots, \lambda_{fD}, s_g^2, \lambda_{g1}, \ldots, \lambda_{gD}\right]^\top$. Choice of the form of the covariance function $k\left(\boldsymbol{x}_p, \boldsymbol{x}_q\right)$ follows from the system description (1) and is inspired by [23], where this particular structure of covariance was exploited to ease the derivation of control law. The term $k_n$ models the system output disturbance as a white Gaussian noise with variance $\sigma_e^2$. The form of the covariance functions $k_f$ and $k_g$ is often called *squared exponential* with *automatic relevance determination*, which is a common choice in a number of applications [4, 7]. This choice expresses the assumption that the functions $f$ and $g$ involved in the system equation (1) are smooth, i.e. they meet the assumption A1 from section 2.

With GP prior specified by (20) – (24), we can now derive expressions for system output prediction and prediction variance. Using the equations (11) – (16) we obtain

$$\mathrm{E}\left[\hat{y}_{k+1}\right] = \boldsymbol{\mu}^p_{k+1} = \boldsymbol{J}_k \boldsymbol{\mu}^g_k, \tag{25}$$

$$\mathrm{var}\left[\hat{y}_{k+1}\right] = \boldsymbol{C}^p_{k+1} = k_{ss} + \boldsymbol{J}_k \left(\boldsymbol{C}^g_k - k(\boldsymbol{X}, \boldsymbol{X})\right) \boldsymbol{J}_k, \tag{26}$$

where the quantities $\boldsymbol{J}_k$, $\boldsymbol{k}_s$ and $k_{ss}$ are computed as

$$\boldsymbol{J}_k = \boldsymbol{k}_s \boldsymbol{K}^{-1}, \tag{27}$$

$$\boldsymbol{K} = k_f\left(\boldsymbol{X}^a, \boldsymbol{X}^a\right) + k_g\left(\boldsymbol{X}^a, \boldsymbol{X}^a\right) \circ \boldsymbol{U}, \tag{28}$$

$$\boldsymbol{k}_s = \boldsymbol{k}_{sf} + \boldsymbol{k}_{sg}\, u_k + \boldsymbol{k}_{sn}, \tag{29}$$

$$k_{ss} = k_{ssf} + k_{ssg}\, u_k^2 + k_{ssn} \tag{30}$$

with

$$\boldsymbol{k}_{sf} = k_f\left(\boldsymbol{X}^a, \boldsymbol{x}^a_k\right), \qquad k_{ssf} = k_f\left(\boldsymbol{x}^a_k, \boldsymbol{x}^a_k\right), \tag{31}$$

$$\boldsymbol{k}_{sg} = k_g\left(\boldsymbol{X}^a, \boldsymbol{x}^a_k\right) \circ \boldsymbol{U}, \qquad k_{ssg} = k_g\left(\boldsymbol{x}^a_k, \boldsymbol{x}^a_k\right), \tag{32}$$

$$\boldsymbol{k}_{sn} = k_n\left(\boldsymbol{X}^a, \boldsymbol{x}^a_k\right), \qquad k_{ssn} = k_n\left(\boldsymbol{x}^a_k, \boldsymbol{x}^a_k\right), \tag{33}$$

where the symbol $\circ$ denotes the element-wise product of two vectors. Finally, update of $\boldsymbol{\mu}^g$ and $\boldsymbol{C}^g$ is performed by the already familiar equations (14) – (16). Note, that we also get an estimate of the uncertainty associated with the system output prediction, which can be of use in the control design. Also note, that the RGP algorithm was formulated in section 3.1 as being able to process $nk$ observations at a time. For the system identification we only consider processing of one observation at a time; thus $nk = 1$.

To fully elucidate the workings of RGP regression algorithm with ad-hoc hyper-parameter learning in the system identification process, the following summary is provided.

**RGP identification algorithm:**

| | |
|---|---|
| Step 0: | Set $k \leftarrow 0$, initialise hyper-parameters $\boldsymbol{\theta}_0$, choose basis vector locations $\boldsymbol{X}$, set $\boldsymbol{\mu}^g_0 = \boldsymbol{0}$, $\boldsymbol{C}^g_0 = k(\boldsymbol{X}, \boldsymbol{X}; \boldsymbol{\theta}_0)$, |
| Step 1: | Generate input $u_k$ and measure the system output $y_{k+1}$, |
| Step 2: | Calculate system output prediction $\boldsymbol{\mu}^p_{k+1}$ by (25) and (26), utilizing (27) – (33). |
| Step 3: | Calculate updated $\boldsymbol{\mu}^g_{k+1}$, $\boldsymbol{C}^g_{k+1}$ from $\boldsymbol{\mu}^p_{k+1}$, $\boldsymbol{C}^p_{k+1}$ using (14) – (16), |
| Step 4: | Optimise hyper-parameters: $\boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}} \log\left(p\left(\boldsymbol{\mu}^g_{k+1} \mid \boldsymbol{X}, \boldsymbol{\theta}\right)\right)$. |
| Step 5: | $k \leftarrow k+1$, go to Step 1. |

## 4. Numerical simulations

In numerical simulations the RGP was compared with the performance of the FGP on the following benchmarking example of non-linear stochastic discrete-time system

$$y_{k+1} = \frac{1.5\, y_k}{1 + y_k^2} + \left(2 + \cos\left(y_k\right)\right) u_k + e_{k+1}, \tag{34}$$

where the sequence $e_k$ is a zero-mean white Gaussian noise with variance $\sigma_e^2 = 0.025$. In this case the vector of regressors $\boldsymbol{x}_k = \left[y_{k-1}, u_{k-1}\right]$ is two dimensional. Hyper-parameters were initialised with values $\boldsymbol{\theta}_0 = [0, 0, 0, 0]^\top$ for both algorithms. As the excitation signal $u_k$ we considered uniformly random noise within the interval $[-3, 3]$.

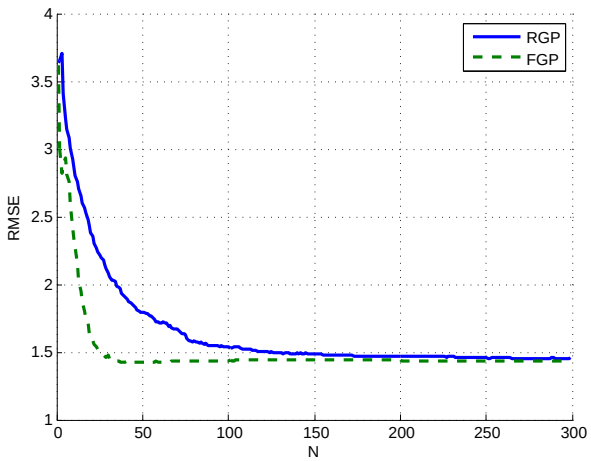


Figure 1: Root mean square error.

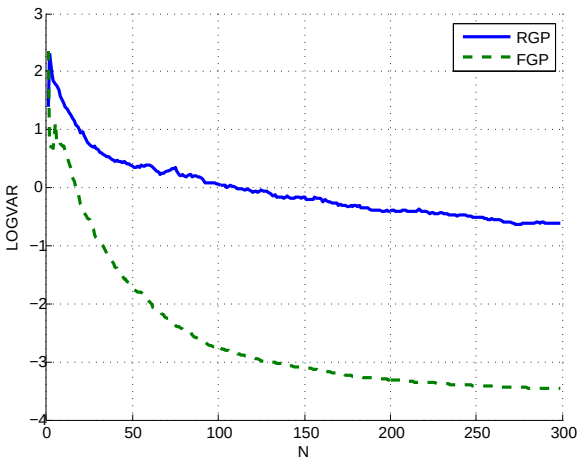


Figure 2: Logarithm of average predictive variance.

In case of RGP algorithm, 75 basis vectors were used to create a $15 \times 5$ grid, that covered the area of state space, where the system operates. Initialisation $\boldsymbol{\mu}_0^g = \mathbf{0}$ and $\boldsymbol{C}_0^g = k(\boldsymbol{X}, \boldsymbol{X})$ was performed with initial hyper-parameter values in mind. Ad-hoc hyper-parameter optimisation routine was started with the estimate obtained in previous iteration. The FGP was trained using all available data up to the current time step, where the hyper-parameter optimisation routine was always started with initial values $\boldsymbol{\theta}_0$. Hyper-parameter optimisation was realised using MATLAB's `fminunc()` solver with the maximum number of iterations set to 100.

The experiments performed were focused on the comparison of the two approaches in terms of obtained GP model error and computational demands for prediction in every time step. Root mean square error (RMSE)

$$\mathrm{RMSE} = \sqrt{\frac{1}{N_{ts}} \sum_{i=1}^{N_{ts}} \left(y_i - \hat{y}_i\right)^2} \tag{35}$$

was used to measure the error in model validation process, where $y_i$ is the measurement of system output and $\hat{y}_i$ is the GP model prediction. Furthermore, the logarithm of average predictive variance

$$\mathrm{LOGVAR} = \log\left(\frac{1}{N_{ts}} \sum_{i=1}^{N_{ts}} \mathrm{var}\left(\hat{y}_i\right)\right) \tag{36}$$

was used to quantify the overall model prediction uncertainty. Both metrics were calculated at each of the $N = 300$ time steps using $N_{ts} = 1600$ test points, which were obtained from the system operation. The computational demand is assumed to be the sum of the time required for obtaining the current hyper-parameter estimate and the time required for computing the one-step ahead prediction (model response). This was implemented by employing MATLAB's `tic/toc` command functionality and results recorded for every time step. 100 MC simulations were run with the results shown in figures.

As seen in figure 1, RMSE of the FGP is slightly lower than that of the RGP algorithm. This can be caused by the fact, that the RGP uses for prediction a *fixed amount* of predefined basis vectors. The FGP, however, makes predictions based on *all* available past data points. The main drawback of the RGP is that, when prediction is to be made further away from the area covered by the basis vectors, it's quality will be significantly worse than that of the FGP. Development of the LOGVAR in figure 2 suggests, that predictions of the RGP algorithm are less confident than those of the FGP. Figure 3 compares the system and model response during training. Note how the model response approaches the true system response

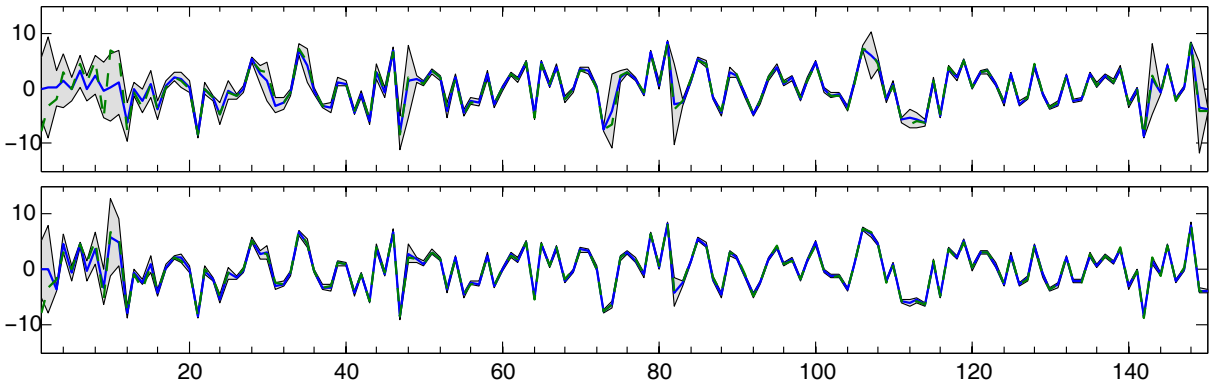


Figure 3: System response (dashed) compared to the RGP (top) and FGP (bottom) model response with the increasing amount of data used for system identification (GP model training). The grey band represents the model uncertainty.

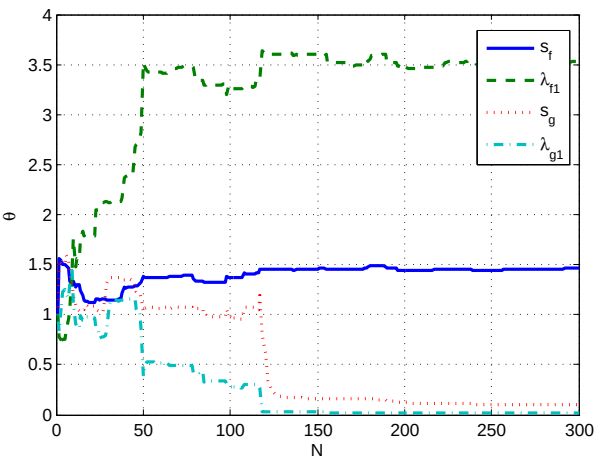


Figure 4: Hyper-parameter development in time.

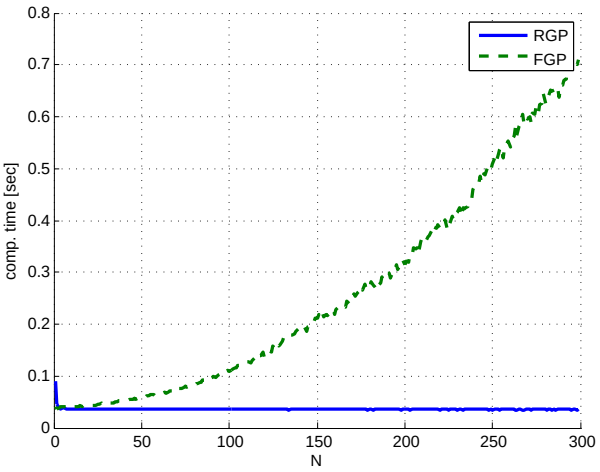


Figure 5: Computation time per iteration.

and the model uncertainty is reduced as the amount of data used for identification increases. Figure 4 shows converging estimates of each of the hyper-parameters as the number of available data for system identification (GP model training) increases. Estimates may not always converge to the same value since the marginal likelihood has multiple local extrema. The figure 5 demonstrates the unsurprising fact, that the RGP algorithm, owing to it's recursive nature and limited number of solver iterations, has constant computational demands per time step. On the contrary, time requirements of the FGP algorithm for prediction increase with every time step regardless of the limited number of solver iterations.

## 5. Conclusions

In this paper, we proposed use of the RGP algorithm with an ad-hoc hyper-parameter learning procedure for the non-linear system identification. The RGP algorithm was compared with the FGP approach in terms of computational demands per iteration as well as the RMSE and the logarithm of average predictive variance. The ad-hoc learning procedure provides converging hyper-parameter estimates. In performed simulations it was demonstrated that use of the RGP algorithm is clearly superior to the FGP approach in terms computational demands. Compared to the FGP, the RGP algorithm provides more conservative predictions and achieves comparable RMSE, though exhibits slower convergence. Main

disadvantage of the RGP algorithm is that the number of basis vectors increases exponentially with the dimension of the state space (vector of regressors). Also, quality of predictions at the locations further away from the area of state space covered by the basis vectors is worse than those of the FGP.

Future research activity will be aimed at incorporating the RGP identification algorithm into the functional dual adaptive control loop.

## Acknowledgments

The work was supported by the European Regional Development Fund (ERDF), project "NTIS - New Technologies for the Information Society", European Centre of Excellence, CZ.1.05/1.1.00/02.0090 and by the Czech Science Foundation, project GACR P103/13/07058J.

## References

[1] Neal R M 1995 *Bayesian learning for neural networks* Ph.D. thesis University of Toronto
[2] Ko J and Fox D 2008 *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* vol 27 (Springer) pp 3471–3476 ISBN 9781424420582
[3] Ko J, Klein D J, Fox D and Haehnel D 2007 *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on* vol 54 (IEEE) pp 1901–1907 ISBN 1424409128
[4] Deisenroth M P, Turner R D, Huber M F, Hanebeck U D and Rasmussen C E 2012 *IEEE Trans. on Automatic Control* **57** 1865–1871
[5] Särkkä S, Hartikainen J, Svensson L and Sandblom F 2014 *Information Fusion (FUSION), 2014 17th International Conference on* pp 1–8
[6] Kocijan J, Murray-Smith R, Rasmussen C and Girard A 2004 *Proc. of the American Control Conf.* vol 3 pp 2214–2219
[7] Kocijan J, Girard A, Banko B and Murray-Smith R 2005 *Mathematical and Computer Modelling of Dynamic Systems* **11** 411–424
[8] Girard A, Rasmussen C E, Quiñonero Candela J and Murray-Smith R 2003 *Advances in Neural Information Processing Systems 15* ed Becker S, Thrun S and Obermayer K (MIT Press) pp 545–552
[9] Deisenroth M and Rasmussen C E 2011 *Proc. of the 28th Int. Conf. on Machine Learning (ICML-11)* pp 465–472
[10] Park S, Mustafa S K and Shimada K 2013 *Robotic Intelligence In Informationally Structured Space (RiiSS), 2013 IEEE Workshop on* (IEEE) pp 120–127 ISBN 978-1-4673-5877-4
[11] Rasmussen C E and Ghahramani Z 2003 *Advances in Neural Information Processing Systems 15* 1 ed Becker S, Thrun S and Obermayer K (MIT Press) pp 505—-512
[12] Särkkä S, Solin A and Hartikainen J 2013 *IEEE Signal Processing Magazine* **30** 51–61 ISSN 1053-5888
[13] Csató L and Opper M 2002 *Neural Computation* **14** 641–668
[14] Chowdhary G, Kingravi H A, How J P and Vela P A 2014 *Neural Networks and Learning Systems, IEEE Transactions on* **PP** 1–1 ISSN 2162-237X
[15] Ranganathan A, Yang M H and Ho J 2011 *IEEE Transactions on Image Processing* **20** 391–404
[16] Huber M 2013 *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE Int. Conf. on* pp 3362–3366 ISSN 1520-6149
[17] Fabri S and Kadirkamanathan V 2001 *Functional Adaptive Control: An Intelligent Systems Approach* (Springer London) ISBN 978-1-4471-1090-3
[18] Šimandl M, Král L and Hering P 2005 *IFAC Proc. Volumes (IFAC-PapersOnline)* vol 16 pp 58–63
[19] Sbarbaro D, Murray-Smith R and Valdes A 2004 *Proc. of Int. Symp. on Neural Networks* pp 52–58
[20] Rasmussen C E and Williams C K 2006 *Gaussian Processes for Machine Learning* (The MIT Press) ISBN 978-262-18253-9
[21] Quiñonero-Candela J and Rasmussen C E 2005 *The Journal of Machine Learning Research* **6** 1939–1959
[22] Huber M F 2014 *Pattern Recognition Letters* **45** 85–91 ISSN 01678655
[23] Sbarbaro D and Murray-Smith R 2005 *Switching and Learning in Feedback Systems* vol 3355 (Springer) pp 140–157