White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of Sheffield.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/139521/

Version: Accepted Version

Article:
Yang, L., Wang, K. and Mihaylova, L.S. (2019) Online sparse multi-output Gaussian process regression and learning. IEEE Transactions on Signal and Information Processing over Networks, 5 (2). pp. 258-272. ISSN: 2373-776X

https://doi.org/10.1109/TSIPN.2018.2885925

© 2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works. Reproduced in accordance with the publisher's self-archiving policy.

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Online Sparse Multi-Output Gaussian Process Regression and Learning

Le Yang, *Member, IEEE,* Ke Wang, and Lyudmila Mihaylova*, *Senior Member, IEEE*

***Abstract*—This paper proposes an approach for online training of a sparse multi-output Gaussian process (GP) model using sequentially obtained data. The considered model combines linearly multiple latent sparse GPs to produce correlated output variables. Each latent GP has its own set of inducing points to achieve sparsity. We show that given the model hyperparameters, the posterior over the inducing points is Gaussian under Gaussian noise since they are linearly related to the model outputs. However, the inducing points from different latent GPs would become correlated, leading to a full covariance matrix cumbersome to handle. Variational inference is thus applied and an approximate regression technique is obtained, with which the posteriors over different inducing point sets can always factorize. As the model outputs are non-linearly dependent on the hyperparameters, a novel marginalized particle filer (MPF)-based algorithm is proposed for the online inference of the inducing point values and hyperparameters. The approximate regression technique is incorporated in the MPF and its distributed realization is presented. Algorithm validation using synthetic and real data is conducted, and promising results are obtained.**

***Index Terms*—Multi-output Gaussian Processes, Sparse approximation, online regression and learning, marginalized particle filter, Kullback-Leibler divergence.**

I. Introduction

Gaussian process (GP) regression provides a flexible and powerful non-parametric framework for Bayesian inference. It imposes a GP prior, which is an infinite-dimensional Gaussian distribution, over the function to be modeled. It is capable of capturing complex relationships between the input data and (noisy) output data [1], [2]. The GP is completely characterized by its mean and covariance functions. Their parameters together with other model parameters such as the measurement noise precisions are collectively called *hyperparameters* that also need to be learned from the training data. GP regression has found diverse applications in indoor localization [3]–[5], image processing and visual tracking [6], [7], change point detection [8] and sensor (robotic) network management [9], [10]. It has also been used for wireless channel prediction/identification [11], [12] and state transition dynamics modeling in Bayesian filtering [13]–[15].

Le Yang was with the School of Internet of Things (IoT) Engineering, Jiangnan University, Wuxi 214122, China and is now with the Department of Electrical and Computer Engineering, University of Canterbury, Christchurch 8020, New Zealand (e-mail: le.yang.le@gmail.com). Ke Wang is with the School of Engineering, University of Glasgow, Glasgow G12 8QQ, Scotland (e-mail: k.wang.1@research.gla.ac.uk). Lyudmila Mihaylova is with the Department of Automatic Control and System Engineering (ACSE), University of Sheffield, Sheffield S1 3JD, U.K. (e-mail: l.s.mihaylova@sheffield.ac.uk).

Manuscript received December 23, 2017; revised September 8, 2018.

* Corresponding author

A. Multi-output Gaussian Processes

The standard GP model assumes a single output variable only. However, in practice, multi-output functions arise widely in geostatistics [16], robotics [17], sensor networks [18], chemometrics [19], traffic flow forecasting [20] and computational physics [21], [22], just to name a few. A typical example of multi-output regression is the robot inverse dynamics problem [17], [23]. Modeling the inverse dynamics accurately is essential for e.g., computing the torques needed at the joints of a robot arm to drive it along the planned trajectory.

The conventional approach for multi-output regression consists of applying an independent GP to each output variable (see e.g., [24], [25]). This is in general a suboptimal solution, because the cross correlation among the output variables is not taken into account. Simultaneously modeling multiple output variables can improve prediction performance and enable the recovery of missing data due to sensor failure and/or malfunction [18], [26]. One straightforward way for generalizing single-output GP regression to the multi-output case is to directly make the GP priors for different output variables correlated with one another [1]. An alternative method relies on convolving the latent GPs with different kernels to produce multiple outputs and introduce cross correlation among them [27]–[29]. It is also popular to establish a multi-output GP model by linearly combining the latent GPs [26], [30]–[33].

B. Complexity Reduction via Sparse Approximation

As in the single-output GP regression, the computational cost of performing exact inference with multi-output GP models would become prohibitive when the training data set is large. This is mainly due to the need of inverting a large covariance matrix. To reduce complexity, [19], [34] constrained the covariance structure to establish a more parsimonious multi-output model and speed up the computations. In [28], [29], the *sparse* approximation [35] was integrated into the convolved multi-output GP regression and two approximate inference techniques were developed. They have very similar functional forms as the partially independent training conditional (PITC) [35] and fully independent training conditional (FITC) [35], [36] approximations originally proposed for single-output GPs. Both of them rely on a set of *inducing points*, also referred to as inducing variables [36], [37] and basis vectors [38], whose number is usually much smaller than that of the training data points. The use of inducing points reduces the exact GP inference to learning the posterior over the inducing points and model hyperparameters, resulting in significant reduction in the computational cost.

The variational formulation of sparse approximation was proposed in [39]–[43] and shown in [44] to outperform the FITC approximation. It was introduced into multi-output GP regression, which lead to the establishment of the collaborative multi-output GP (COGP) model [26] and variational dependent multi-output GP dynamic system (VDM-GPDS) [45]. The former combines multiple latent sparse GPs to produce correlated outputs while the latter is based on the convolved multi-output GP model.

The aforementioned sparse multi-output GPs are all trained in a batch mode, i.e., both the posterior over the inducing points and model hyperparameters are learned in an offline manner using all the training data points.

C. Contributions

This paper has the following contributions:

- *Online regression and hyperparameter learning.* We propose an on-line approach for training the sparse multi-output GP model that is a linear combination of latent sparse GPs, each having its own inducing point set. The joint learning of the posterior over the inducing points and model hyperparameters using streaming data is achieved with the newly developed marginalized particle filter (MPF). The MPF is adopted because the training of the considered multi-output model can be cast into a Bayesian filtering framework with a mixed linear/nonlinear state-space model [46]. The use of MPF leads to flexibility and to the marginalization of the inducing points since they are the conditionally linear states given the nonlinear states, the hyperparameters.
- *Approximate model regression.* We show that given the hyperparameters, the posterior over inducing points is Gaussian and the inducing points of different latent GPs would become correlated with one another. This results in a large and full covariance matrix. We approximate the true posterior by applying the variational inference to make different inducing point sets independent, reduce complexity and enable efficient *distributed implementation*. The obtained approximate regression technique is incorporated into the MPF for online model training.
- *Performance verification.* The design is validated using both synthetic and real-world data and promising results are obtained.

D. Related Works

We highlight the difference between our work and some previous attempts on the online training of GP models. In [47], a technique called sparse online GP (SOGP) was proposed for training single-output GPs. It maintains a fixed-size *active set* whose elements are selected from sequentially arriving data and uses exact GP inference for model learning. The local GP (LGP) technique [48], [49], on the other hand, partitions the streaming data into non-overlapping groups, each of which is exploited to train a separate single-output GP model. The LGP learning is still based on exact GP inference and no inducing point-based sparse approximations were considered.

To train sparse GPs with inducing points, online regression techniques were developed in [50] for FITC and PITC approximations. They update the posterior over the inducing points once a new measurement is available. However, there is no learning of the model hyperparameters. In [38], the joint posterior of the inducing points and model hyperparameters was assumed to be Gaussian, and it applied a Gaussian filter for inference. We extend this work to the multi-output case but use a MPF to update the posterior of the model hyperparameters, thus eliminating the Gaussian posterior assumption. The MPF-based regression and learning proposed in this work is similar to the one developed in [51], [52] for updating the joint posterior of the model prediction at *given* inputs and hyperparameters. However, in [51], [52], sparse approximation was not used and the model has a single output only. Moreover, it requires the locations of the model predictions are known *before* training, which could limit its application scope. The considered sparse multi-output GP model is close to COGP [26] that relies on offline inference. In this paper, we develop a new approach that allows the online inference of both the inducing point values and hyperparameters of the sparse multi-output GP model.

The rest of the paper is organized as follows. Section II describes the considered GP model. The proposed online regression and learning algorithm is presented in Section III. Experimental results using synthetic and real-world data are given in Section IV. Section V concludes the paper. We shall follow throughout this work the convention that bold lowercase and uppercase letters denote column vectors and matrices.

II. Sparse Multi-Output GP Model

Consider the sparse multi-output GP model shown in Fig. 1. It takes $\mathbf{x} \in \mathbb{R}^{D\times 1}$ as input, where D is the input dimensionality. The model has P outputs, each of which is a linear combination of Q latent functions $g_j(\mathbf{x})$, $j = 1, 2, ..., Q$. The ith model output $y_i(\mathbf{x})$ is equal to

$$y_i(\mathbf{x}) = \sum_{j=1}^{Q} w_{i,j} \cdot g_j(\mathbf{x}) + \epsilon_i,\ \ i = 1, 2, ..., P, \tag{1}$$

where $w_{i,j}$ are the mixing coefficients, ϵ_i is a zero-mean independent Gaussian noise of the i-th output channel with precision β_i, i.e., $\epsilon_i \sim \mathcal{N}(0, \beta_i^{-1})$. Note that the latent functions $g_j(\mathbf{x})$ operate on the *same* input $\mathbf{x}$ and as such, there are P input-output relationships to be modeled simultaneously. Besides, the latent functions are *shared* among the outputs, making the output variables correlated as in [26], [30]–[33]. In practical multi-output regression problems, output variables are rarely linearly dependent. As a result, with the considered GP model given in Fig. 1, at least $Q = P$ latent functions are needed to effectively capture the P input-output relationships. It is a common practice in multi-output GP regression to use multiple latent functions, e.g. [1], [26]–[33].

The latent functions $g_j(\mathbf{x})$ have independent GP priors denoted as

$$g_j(\mathbf{x}) \sim \mathcal{GP}(m_j(\mathbf{x}), k_j(\mathbf{x}, \mathbf{x}')), \tag{2}$$

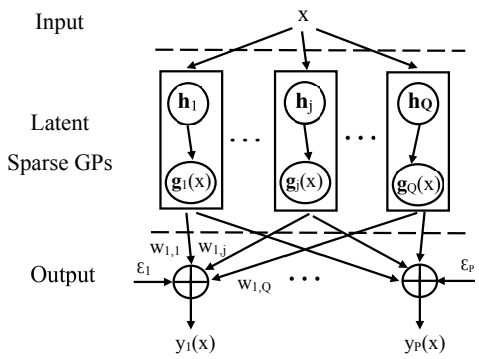

Fig. 1. The considered sparse multi-output GP model.

where $m_j(\mathbf{x})$ is the mean function and $k_j(\mathbf{x}, \mathbf{x}')$ represents the positive semi-definite covariance function. Among the available choices, the zero mean function $m_j(\mathbf{x}) = 0$ and stationary squared exponential (SE) kernel

$$k_j(\mathbf{x}, \mathbf{x}') = \alpha_j^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{\Lambda}_j^{-1}(\mathbf{x} - \mathbf{x}')\right) \quad (3)$$

are widely used. Here, α_j^2 is the variance of the latent function $g_j(\mathbf{x})$. The scaling matrix $\mathbf{\Lambda}_j$ is equal to $\mathbf{\Lambda}_j = \text{diag}(\lambda_{1,j}, \lambda_{2,j}, ..., \lambda_{D,j})$ whose diagonal elements $\lambda_{d,j}$, $d = 1, 2, ..., D$, are the characteristic length scales for the corresponding input dimensions. Other mean and covariance functions are also available. For example, the mean function can be the linear combination of a set of basis functions. There are other stationary kernels including the covariance functions of the Matérn class as well as non-stationary kernels such as the neural network covariance function (see [1]).

The parameters of the mean and covariance functions together with the mixing coefficients $w_{i,j}$ and noise precisions β_i are the hyperparameters of the considered multi-output GP model. With the zero mean function and SE kernel in (3), the model hyperparameter vector would be

$$\boldsymbol{\theta} = [\boldsymbol{\alpha}^T, \boldsymbol{\lambda}_1^T, \boldsymbol{\lambda}_2^T, ..., \boldsymbol{\lambda}_Q^T, \mathbf{w}_1^T, \mathbf{w}_2^T, ...\mathbf{w}_Q^T, \boldsymbol{\beta}^T]^T, \quad (4)$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_Q]^T$, $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_P]^T$, $\boldsymbol{\lambda}_j = [\lambda_{1,j}, \lambda_{2,j}, ..., \lambda_{D,j}]^T$ and $\mathbf{w}_j = [w_{1,j}, w_{2,j}, ..., w_{P,j}]^T$. The vectors $\boldsymbol{\lambda}_j$ and $\mathbf{w}_j$ collect the characteristic length scales and mixing coefficients for the j-th latent function $g_j(\mathbf{x})$. The model hyperparameter vector $\boldsymbol{\theta}$ under other choices of the mean and covariance functions can be defined similarly.

To achieve sparsity and efficiently deal with large scale problems, we introduce a set of inducing inputs into each latent GP. Let $\mathbf{Z}_j = [\mathbf{z}_{1,j}, \mathbf{z}_{2,j}, ..., \mathbf{z}_{M,j}]$ collect the locations of the M inducing points of the jth latent function, where $\mathbf{z}_{k,j}$, $k = 1, 2, ..., M$, are indeed the inputs for the inducing points $g_j(\mathbf{z}_{k,j})$ and thus, they are $D \times 1$ column vectors. Moreover, define $\mathbf{h}_j = [g_j(\mathbf{z}_{1,j}), g_j(\mathbf{z}_{2,j}), ..., g_j(\mathbf{z}_{M,j})]^T$ to collect $g_j(\mathbf{z}_{k,j})$. When every latent GP employs the zero mean function, SE kernel and M inducing points, from (4), the sparse multi-output GP model in consideration would have in total $P + (1 + P + D)Q + MQ$ unknowns.

Since $g_j(\mathbf{x})$ has a GP prior with mean function $m_j(\mathbf{x})$ and covariance function $k_j(\mathbf{x}, \mathbf{x}')$, the prior distribution of $\mathbf{h}_j$ would be

$$p_0(\mathbf{h}_j) = \mathcal{N}(\mathbf{m}_j(\mathbf{Z}_j), \mathbf{K}_j(\mathbf{Z}_j, \mathbf{Z}_j)). \quad (5)$$

The mean vector $\mathbf{m}_j(\mathbf{Z}_j)$ is defined as

$$\mathbf{m}_j(\mathbf{Z}_j) = [m_j(\mathbf{z}_{1,j}), m_j(\mathbf{z}_{2,j}), ..., m_j(\mathbf{z}_{M,j})]^T. \quad (6)$$

The elements of the covariance matrix $\mathbf{K}_j(\mathbf{Z}_j, \mathbf{Z}_j)$ are equal to

$$[\mathbf{K}_j(\mathbf{Z}_j, \mathbf{Z}_j)]_{k,l} = k_j(\mathbf{z}_{k,j}, \mathbf{z}_{l,j}), \ k, l = 1, 2, ..., M. \quad (7)$$

Here, each latent function is assumed to have the same number of inducing points. This assumption can be relaxed without affecting the validity of the online sparse multi-output GP training technique developed in the next section. It is also worthwhile to point out that the sparse multiple-output GP model considered in this work is slightly more general than the COGP from [26]. In particular, in COGP, each output variable has an *individual* latent GP. In the considered model, we can easily create an individual latent GP for the i-th model output by setting the i-th mixing coefficient of $g_j(\mathbf{x})$, $w_{i,j}$, to 1 and others to 0.

III. Online Regression and Learning

In this section, a MPF-based algorithm for online regression and learning of the sparse multi-output GP model described in Section II is proposed. It is able to update in a recursive manner the posterior over the inducing points and model hyperparameters using sequentially obtained data. The algorithm development begins with deriving the predictive distribution of the model outputs when the posterior over the inducing points is Gaussian and the hyperparameters are given. By applying the Kalman filter update [53] and variational inference, an approximate regression method for updating the posterior over the inducing points is then established. Finally, we present the MPF that incorporates the developed approximated regression method to achieve simultaneous inference of the inducing points and model hyperparameters.

A. Model Prediction

Denote the posterior over the inducing points of the jth latent sparse GP, $\mathbf{h}_j$, at previous time step as $p(\mathbf{h}_j|\mathcal{D}_{0:t-1})$. Here, $t = 1, 2, ...$, and $\mathcal{D}_{0:t-1}$ represent, respectively, the time index and the set containing all the training data received up to $t-1$. Assume $\mathbf{h}_j$ follows a Gaussian distribution such that

$$p(\mathbf{h}_j|\mathcal{D}_{0:t-1}) = \mathcal{N}(\boldsymbol{\mu}_{t-1}^{\mathbf{h}_j}, \mathbf{C}_{t-1}^{\mathbf{h}_j}), \ j = 1, 2, ..., Q, \quad (8)$$

where $\boldsymbol{\mu}_{t-1}^{\mathbf{h}_j}$ and $\mathbf{C}_{t-1}^{\mathbf{h}_j}$ are the associated mean and covariance[1]. It is further assumed that the inducing points from different sparse latent GPs are *conditionally independent* to each other. The above two assumptions come from that: 1) the inducing points of different latent GPs, $\mathbf{h}_j$, have independent

[1]For $t = 1$, we set $p(\mathbf{h}_j|\mathcal{D}_{0:t-1})$ to be the prior distribution of $\mathbf{h}_j$, $p_0(\mathbf{h}_j)$, given in (5).

Gaussian priors (see (5)) and 2) with the approximated regression method developed later in Section III.B, the posteriors of $\mathbf{h}_j$ at time t would remain to be Gaussian and conditionally independent if they are independently Gaussian distributed at time $t-1$.

Define the composite inducing point vector

$$\mathbf{h} = [\mathbf{h}_1^T, \mathbf{h}_2^T, ..., \mathbf{h}_Q^T]^T. \tag{9}$$

As a result, its posterior distribution at time step $t-1$ is

$$p(\mathbf{h}|\mathcal{D}_{0:t-1}) = \prod_{j=1}^{Q} p(\mathbf{h}_j|\mathcal{D}_{0:t-1}) = \mathcal{N}(\boldsymbol{\mu}_{t-1}^{\mathbf{h}}, \mathbf{C}_{t-1}^{\mathbf{h}}), \tag{10}$$

where

$$\boldsymbol{\mu}_{t-1}^{\mathbf{h}} = [(\boldsymbol{\mu}_{t-1}^{\mathbf{h}_1})^T, (\boldsymbol{\mu}_{t-1}^{\mathbf{h}_2})^T, ..., (\boldsymbol{\mu}_{t-1}^{\mathbf{h}_Q})^T]^T, \tag{11a}$$

$$\mathbf{C}_{t-1}^{\mathbf{h}} = \text{diag}(\mathbf{C}_{t-1}^{\mathbf{h}_1}, \mathbf{C}_{t-1}^{\mathbf{h}_2}, ..., \mathbf{C}_{t-1}^{\mathbf{h}_Q}). \tag{11b}$$

At the current time step t, we are interested in predicting the model outputs for N_t inputs given $p(\mathbf{h}|\mathcal{D}_{0:t-1})$ and the model hyperparameters in $\boldsymbol{\theta}$. For this purpose, first define $\mathbf{g}_{t,j} = [g_j(\mathbf{x}_{t,1}), g_j(\mathbf{x}_{t,2}), ..., g_j(\mathbf{x}_{t,N_t})]^T$ to include the values of the latent function $g_j(\mathbf{x})$ at the inputs $\mathbf{X}_t = [\mathbf{x}_{t,1}, \mathbf{x}_{t,2}, ..., \mathbf{x}_{t,N_t}]$. Stacking them yields $\mathbf{g}_t = [\mathbf{g}_{t,1}^T, \mathbf{g}_{t,2}^T, ..., \mathbf{g}_{t,Q}^T]^T$. Given the inducing points $\mathbf{h}$, the conditional distribution of $\mathbf{g}_t$ is

$$p(\mathbf{g}_t|\mathbf{h}) = \prod_{j=1}^{Q} p(\mathbf{g}_{t,j}|\mathbf{h}_j). \tag{12}$$

The above factorization comes from the fact that the latent functions $g_j(\mathbf{x})$ have independent GP priors (see Section II). Morever, because the joint prior of $\mathbf{g}_{t,j}$ and $\mathbf{h}_j$ is a finite representation of the GP specified in (2), we have

$$p(\mathbf{g}_{t,j}|\mathbf{h}_j) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_t^{\mathbf{g}_j}, \tilde{\mathbf{C}}_t^{\mathbf{g}_j}),\ j = 1, 2, ..., Q. \tag{13}$$

The mean vector $\tilde{\boldsymbol{\mu}}_t^{\mathbf{g}_j}$ and covariance matrix $\tilde{\mathbf{C}}_t^{\mathbf{g}_j}$ are, by the conditional of multivariate Gaussian distribution [1],

$$\tilde{\boldsymbol{\mu}}_t^{\mathbf{g}_j} = \mathbf{m}_j(\mathbf{X}_t) + \mathbf{P}_{t,j}(\mathbf{h}_j - \mathbf{m}_j(\mathbf{Z}_j)), \tag{14a}$$

$$\tilde{\mathbf{C}}_t^{\mathbf{g}_j} = \mathbf{K}_j(\mathbf{X}_t, \mathbf{X}_t) - \mathbf{P}_{t,j}\mathbf{K}_j(\mathbf{Z}_j, \mathbf{Z}_j)\mathbf{P}_{t,j}^T, \tag{14b}$$

where $\mathbf{m}_j(\mathbf{X}_t) = [m_j(\mathbf{x}_{t,1}), m_j(\mathbf{x}_{t,2}), ..., m_j(\mathbf{x}_{t,N_t})]^T$. The matrix $\mathbf{P}_{t,j}$ is defined as [1]

$$\mathbf{P}_{t,j} = \mathbf{K}_j(\mathbf{X}_t, \mathbf{Z}_j)\mathbf{K}_j^{-1}(\mathbf{Z}_j, \mathbf{Z}_j). \tag{15}$$

The definitions of $\mathbf{K}_j(\mathbf{X}_t, \mathbf{X}_t)$ and $\mathbf{K}_j(\mathbf{X}_t, \mathbf{Z}_j)$ are very similar to that of $\mathbf{K}_j(\mathbf{Z}_j, \mathbf{Z}_j)$ given in (7). Therefore, they are omitted here for brevity.

To derive the predictive distribution of the outputs of the considered sparse multi-output GP model at $\mathbf{X}_t$, we define the composite model output vector as

$$\mathbf{y}_t = [\mathbf{y}_{t,1}^T, \mathbf{y}_{t,2}^T, ..., \mathbf{y}_{t,N_t}^T]^T. \tag{16}$$

Here, $\mathbf{y}_{t,n} = [y_1(\mathbf{x}_{t,n}), y_2(\mathbf{x}_{t,n}), ..., y_P(\mathbf{x}_{t,n})]^T$ collects the P model outputs corresponding to a single input $\mathbf{x}_{t,n}$, $n = 1, 2, ..., N_t$. From (1), the conditional distribution of $\mathbf{y}_t$ given $\mathbf{g}_t$, the latent function values, can be expressed as

$$p(\mathbf{y}_t|\mathbf{g}_t) = \mathcal{N}(\mathbf{H}_t\mathbf{g}_t, \boldsymbol{\Lambda}_t). \tag{17}$$

The covariance $\boldsymbol{\Lambda}_t$ is $\boldsymbol{\Lambda}_t = \mathbf{I}_{N_t} \otimes \boldsymbol{\Lambda}$, where $\otimes$ denotes the Kronecker product and $\mathbf{I}_{N_t}$ is a $N_t \times N_t$ identity matrix. The matrix $\boldsymbol{\Lambda}$ is equal to $\boldsymbol{\Lambda} = \text{diag}(\beta_1^{-1}, \beta_2^{-1}, ..., \beta_P^{-1})$, as a result of the independence between measurement noises from different output channels. The $PN_t \times QN_t$ observation matrix $\mathbf{H}_t$ is given by

$$\mathbf{H}_t = \begin{bmatrix} \mathbf{w}_1 \otimes \mathbf{e}_1^T & \mathbf{w}_2 \otimes \mathbf{e}_1^T & ... & \mathbf{w}_Q \otimes \mathbf{e}_1^T \\ \mathbf{w}_1 \otimes \mathbf{e}_2^T & \mathbf{w}_2 \otimes \mathbf{e}_2^T & ... & \mathbf{w}_Q \otimes \mathbf{e}_2^T \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{w}_1 \otimes \mathbf{e}_{N_t}^T & \mathbf{w}_2 \otimes \mathbf{e}_{N_t}^T & ... & \mathbf{w}_Q \otimes \mathbf{e}_{N_t}^T, \end{bmatrix}, \tag{18}$$

where $\mathbf{e}_n$ is a $N_t \times 1$ column vector with its nth element being 1 and others being 0. The vector $\mathbf{w}_j = [w_{1,j}, w_{2,j}, ...w_{P,j}]^T$ contains the mixing coefficients of the jth latent sparse GP, $j = 1, 2, ..., Q$, as defined in Section II.

The conditional distribution of the model outputs $\mathbf{y}_t$ given the inducing points $\mathbf{h}$ is equal to

$$p(\mathbf{y}_t|\mathbf{h}) = \int p(\mathbf{y}_t|\mathbf{g}_t)p(\mathbf{g}_t|\mathbf{h})\text{d}\mathbf{g}_t. \tag{19}$$

Putting (12) and (17), and carrying out the integration yield

$$p(\mathbf{y}_t|\mathbf{h}) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_t^{\mathbf{y}}, \tilde{\mathbf{C}}_t^{\mathbf{y}}), \tag{20}$$

where

$$\tilde{\boldsymbol{\mu}}_t^{\mathbf{y}} = \mathbf{H}_t[(\tilde{\boldsymbol{\mu}}_t^{\mathbf{g}_1})^T, (\tilde{\boldsymbol{\mu}}_t^{\mathbf{g}_2})^T, ..., (\tilde{\boldsymbol{\mu}}_t^{\mathbf{g}_Q})^T]^T \tag{21a}$$

$$\tilde{\mathbf{C}}_t^{\mathbf{y}} = \mathbf{H}_t \cdot \text{diag}(\tilde{\mathbf{C}}_t^{\mathbf{g}_1}, \tilde{\mathbf{C}}_t^{\mathbf{g}_2}, ..., \tilde{\mathbf{C}}_t^{\mathbf{g}_Q}) \cdot \mathbf{H}_t^T + \boldsymbol{\Lambda}_t. \tag{21b}$$

The predictive distribution of the model outputs $\mathbf{y}_t$ can then be found by marginalizing out $\mathbf{h}$ via

$$p(\mathbf{y}_t|\mathcal{D}_{0:t-1}) = \int p(\mathbf{y}_t|\mathbf{h})p(\mathbf{h}|\mathcal{D}_{0:t-1})\text{d}\mathbf{h}, \tag{22}$$

where the approximation $p(\mathbf{y}_t|\mathbf{h}, \mathcal{D}_{0:t-1}) \approx p(\mathbf{y}_t|\mathbf{h})$ has been applied. The above approximation indicates that the current model outputs $\mathbf{y}_t$ and the training data in $\mathcal{D}_{0:t-1}$ are conditionally independent given the inducing points $\mathbf{h}$. As a result, the information from $\mathcal{D}_{0:t-1}$ is transmitted to $\mathbf{y}_t$ only through the posterior of $\mathbf{h}$, $p(\mathbf{h}|\mathcal{D}_{0:t-1})$. Because the number of inducing points in $\mathbf{h}$ is generally much smaller than that of the data points in the dataset $\mathcal{D}_{0:t-1}$, this is referred to as the inducing point assumption or sparse approximation (see [35]–[38], [50]). With sparse approximation, storing and manipulating directly the training data for inference as in standard GP [1] is no longer necessary, instead, we need only to utilize the posterior of the inducing points. In future works, it would be interesting to compare $p(\mathbf{y}_t|\mathcal{D}_{0:t-1})$ in (22) with the true predictive distribution of the model outputs $\mathbf{y}_t$, which can be derived from the joint distribution of $\mathbf{y}_t$ and $\mathcal{D}_{0:t-1}$. This could provide insights into the approximation error introduced by the use of the inducing point assumption.

Substituting (10) and (20) into (22) and following the computations in the prediction step of the standard Kalman filter yield [53]

$$p(\mathbf{y}_t|\mathcal{D}_{0:t-1}) = \mathcal{N}(\bar{\boldsymbol{\mu}}_t^{\mathbf{y}}, \bar{\mathbf{C}}_t^{\mathbf{y}}). \tag{23}$$

The predicted model outputs at inputs $\mathbf{X}_t$ are given by the mean vector

$$\bar{\boldsymbol{\mu}}_t^{\mathbf{y}} = \mathbf{H}_t\bar{\boldsymbol{\mu}}_t^{\mathbf{g}}. \tag{24}$$

The prediction uncertainty is quantified by the covariance $\bar{\mathbf{C}}_t^{\mathbf{y}}$ equal to

$$\bar{\mathbf{C}}_t^{\mathbf{y}} = \mathbf{H}_t \bar{\mathbf{C}}_t^{\mathbf{g}} \mathbf{H}_t^T + \mathbf{\Lambda}_t. \tag{25}$$

$\bar{\boldsymbol{\mu}}_t^{\mathbf{g}}$ and $\bar{\mathbf{C}}_t^{\mathbf{g}}$ are defined as

$$\bar{\boldsymbol{\mu}}_t^{\mathbf{g}} = [(\bar{\boldsymbol{\mu}}_t^{\mathbf{g}_1})^T, (\bar{\boldsymbol{\mu}}_t^{\mathbf{g}_2})^T, ..., (\bar{\boldsymbol{\mu}}_t^{\mathbf{g}_Q})^T]^T, \tag{26a}$$

$$\bar{\mathbf{C}}_t^{\mathbf{g}} = \text{diag}(\bar{\mathbf{C}}_t^{\mathbf{g}_1}, \bar{\mathbf{C}}_t^{\mathbf{g}_2}, ..., \bar{\mathbf{C}}_t^{\mathbf{g}_Q}), \tag{26b}$$

where for $j = 1, 2, ...Q$,

$$\bar{\boldsymbol{\mu}}_t^{\mathbf{g}_j} = \mathbf{m}_j(\mathbf{X}_t) + \mathbf{P}_{t,j}(\boldsymbol{\mu}_{t-1}^{\mathbf{h}_j} - \mathbf{m}_j(\mathbf{Z}_j)), \tag{27a}$$

$$\bar{\mathbf{C}}_t^{\mathbf{g}_j} = \tilde{\mathbf{C}}_t^{\mathbf{g}_j} + \mathbf{P}_{t,j} \mathbf{C}_{t-1}^{\mathbf{h}_j} \mathbf{P}_{t,j}^T. \tag{27b}$$

Two important observations can be obtained from the above development. First, according to (23), the model prediction follows a Gaussian distribution. This is because the posterior of the inducing points $\mathbf{h}$ is Gaussian, and from (14a) and (21a), both the latent function values $\mathbf{g}_t$ and model outputs $\mathbf{y}_t$ are indeed linearly related to $\mathbf{h}$. Second, note from (27) that computing the predicted latent function values $\mathbf{g}_{t,j}$ and their covariance requires knowledge pertaining to the jth latent sparse GP only. This is due to the assumption that the inducing points of different latent functions are conditionally independent (see (10)) and the latent functions have independent GP priors (see (12)). As a result, the model prediction can be found by first evaluating (27) *in parallel* at the latent GPs and then fusing the results using (24) and (25).

B. Approximate Model Regression

1) Exact Regression: With slight abuse of notations, assume that at time step t, a set of N_t training data points $\{(\mathbf{x}_{t,n}, \mathbf{y}_{t,n})\}_{n=1}^{N_t}$ is received, where $\mathbf{x}_{t,n}$ are the inputs but $\mathbf{y}_{t,n}$ are now the *noisy observations*. We are interested in deriving the updated posterior over the inducing points, denoted by $p(\mathbf{h}|\mathcal{D}_{0:t})$, using the posterior at the previous time step $p(\mathbf{h}|\mathcal{D}_{0:t-1})$ and newly obtained training data. Here, $\mathcal{D}_{0:t} = \mathcal{D}_{0:t-1} \cup \{\mathbf{x}_{t,n}, \mathbf{y}_{t,n}\}_{n=1}^{N_t}$. Again, let $\mathbf{X}_t = [\mathbf{x}_{t,1}, \mathbf{x}_{t,2}, ..., \mathbf{x}_{t,N_t}]$ collect the N_t inputs at time t.

The updated posterior is, according to the Bayesian theorem,

$$p(\mathbf{h}|\mathcal{D}_{0:t}) = \frac{p(\mathbf{y}_t|\mathbf{h}) p(\mathbf{h}|\mathcal{D}_{0:t-1})}{p(\mathbf{y}_t|\mathcal{D}_{0:t-1})}, \tag{28}$$

where $\mathbf{y}_t$ collects all the noisy observations at time step t in a way similar to (16). To obtain (28), the approximation $p(\mathbf{y}_t|\mathbf{h}, \mathcal{D}_{0:t-1}) \approx p(\mathbf{y}_t|\mathbf{h})$ due to the inducing point assumption has been applied (also see discussions under (22)).

To evaluate (28), we can put (10) and (20), and follow the measurement update step of the standard Kalman filter [53] to arrive at [2]

$$p(\mathbf{h}|\mathcal{D}_{0:t}) = \mathcal{N}(\boldsymbol{\mu}_t^{\mathbf{h}}, \mathbf{C}_t^{\mathbf{h}}), \tag{29}$$

where the mean vector $\boldsymbol{\mu}_t^{\mathbf{h}}$ and covariance $\mathbf{C}_t^{\mathbf{h}}$ are given by

$$\boldsymbol{\mu}_t^{\mathbf{h}} = \boldsymbol{\mu}_{t-1}^{\mathbf{h}} + \mathbf{C}_{t-1}^{\mathbf{h}} \mathbf{P}_t^T \mathbf{H}_t^T (\bar{\mathbf{C}}_t^{\mathbf{y}})^{-1} (\mathbf{y}_t - \bar{\boldsymbol{\mu}}_t^{\mathbf{y}}), \tag{30a}$$

$$\mathbf{C}_t^{\mathbf{h}} = ((\mathbf{C}_{t-1}^{\mathbf{h}})^{-1} + \mathbf{P}_t^T \mathbf{H}_t^T (\tilde{\mathbf{C}}_t^{\mathbf{y}})^{-1} \mathbf{H}_t \mathbf{P}_t)^{-1}. \tag{30b}$$

[2]Note that the inducing points $\mathbf{h}$ do not evolve stochastically over time. Besides, the measurements $\mathbf{y}_t$ are conditionally independent of the previously collected training data $\mathcal{D}_{0:t-1}$ given $\mathbf{h}$, due to the inducing point assumption. As a result, the posterior mean and covariance of $\mathbf{h}$ given in (30) can be found alternatively via applying the recursive least squares (RLS) technique [53].

Here $\bar{\boldsymbol{\mu}}_t^{\mathbf{y}}$, $\bar{\mathbf{C}}_t^{\mathbf{y}}$ and $\tilde{\mathbf{C}}_t^{\mathbf{y}}$ are given in (24), (25) and (21b). The matrix $\mathbf{P}_t$ is in the form

$$\mathbf{P}_t = \text{diag}(\mathbf{P}_{t,1}, \mathbf{P}_{t,2}, ..., \mathbf{P}_{t,Q}), \tag{31}$$

where $\mathbf{P}_{t,j}$ is defined in (15).

The *exact* regression (29) can be considered as a generalization of the online regression technique developed in [38] to the multi-output scenario. To gain more insights into the result, we first express $\mathbf{H}_t \mathbf{P}_t$ in the following block form

$$\mathbf{H}_t \mathbf{P}_t = [\mathbf{G}_{t,1}, \mathbf{G}_{t,2}, ..., \mathbf{G}_{t,Q}] \tag{32}$$

where from (18) and (31), $\mathbf{G}_{t,j}$ is equal to

$$\mathbf{G}_{t,j} = \begin{bmatrix} \mathbf{w}_j \otimes \mathbf{e}_1^T \\ \mathbf{w}_j \otimes \mathbf{e}_2^T \\ \vdots \\ \mathbf{w}_j \otimes \mathbf{e}_{N_t}^T \end{bmatrix} \mathbf{P}_{t,j}, \ j = 1, 2, ..., Q. \tag{33}$$

Putting (32) and substituting the definitions of $\boldsymbol{\mu}_{t-1}^{\mathbf{h}}$ and $\mathbf{C}_{t-1}^{\mathbf{h}}$in (11) into (30a), we can write the posterior mean $\boldsymbol{\mu}_t^{\mathbf{h}}$ at time step t as

$$\boldsymbol{\mu}_t^{\mathbf{h}} = [(\boldsymbol{\mu}_t^{\mathbf{h}_1})^T, (\boldsymbol{\mu}_t^{\mathbf{h}_2})^T, ..., (\boldsymbol{\mu}_t^{\mathbf{h}_Q})^T]^T, \tag{34}$$

where $\boldsymbol{\mu}_t^{\mathbf{h}_j}$ is the posterior mean of the inducing points of the jth latent sparse GP, and it is equal to

$$\boldsymbol{\mu}_t^{\mathbf{h}_j} = \boldsymbol{\mu}_{t-1}^{\mathbf{h}_j} + \mathbf{C}_{t-1}^{\mathbf{h}_j} \mathbf{G}_{t,j}^T (\bar{\mathbf{C}}_t^{\mathbf{y}})^{-1} (\mathbf{y}_t - \bar{\boldsymbol{\mu}}_t^{\mathbf{y}}). \tag{35}$$

The above result indicates that updating the posterior mean of $\mathbf{h}_j$ can be performed *individually* at the jth latent function as long as the term $(\bar{\mathbf{C}}_t^{\mathbf{y}})^{-1}(\mathbf{y}_t - \bar{\boldsymbol{\mu}}_t^{\mathbf{y}})$ is available.

We next evaluate the posterior covariance $\mathbf{C}_t^{\mathbf{h}}$. By substituting (32) and (11b) into (30b), we can verify that $\mathbf{C}_t^{\mathbf{h}}$ can be expressed in the block form given in (36) on the top of the next page. Here, $\mathbf{C}_t^{\mathbf{h}_j}$ is equal to, for $j = 1, 2, ..., Q$,

$$\begin{aligned} \mathbf{C}_t^{\mathbf{h}_j} &= \left((\mathbf{C}_{t-1}^{\mathbf{h}_j})^{-1} + \mathbf{G}_{t,j}^T (\tilde{\mathbf{C}}_t^{\mathbf{y}})^{-1} \mathbf{G}_{t,j} \right)^{-1} \\ &= \mathbf{C}_{t-1}^{\mathbf{h}_j} - \mathbf{C}_{t-1}^{\mathbf{h}_j} \mathbf{G}_{t,j}^T (\tilde{\mathbf{C}}_t^{\mathbf{y}} + \mathbf{G}_{t,j} \mathbf{C}_{t-1}^{\mathbf{h}_j} \mathbf{G}_{t,j}^T)^{-1} \mathbf{G}_{t,j} \mathbf{C}_{t-1}^{\mathbf{h}_j} \end{aligned} \tag{37}$$

where the matrix inversion lemma [54] has been used to establish the second equality. $\mathbf{C}_{t-1}^{\mathbf{h}_j}$ is the posterior covariance of $\mathbf{h}_j$ at the previous time step $t-1$ and $\tilde{\mathbf{C}}_t^{\mathbf{y}}$ is defined in (21b). It can be seen from (36) that $\mathbf{C}_t^{\mathbf{h}}$ is a full matrix, although the posterior covariance of $\mathbf{h}$ at the previous time step $t-1$, $\mathbf{C}_{t-1}^{\mathbf{h}}$, is block diagonal (see (11b)). In other words, with the exact regression (29), the inducing points from different latent functions would eventually become correlated.

Manipulating the $QM \times QM$ full posterior covariance matrix $\mathbf{C}_t^{\mathbf{h}}$ is cumbersome. Specifically, this would render the distributed computation of the covariances of the latent function predictions using (27b) infeasible, as it cannot utilize the off-diagonal blocks of the posterior covariance of the inducing points. Besides, by again applying the matrix inversion lemma [54], we can rewrite (30b) as, after putting (21b) and (25),

$$\mathbf{C}_t^{\mathbf{h}} = \mathbf{C}_{t-1}^{\mathbf{h}} - \mathbf{C}_{t-1}^{\mathbf{h}} \mathbf{P}_t^T \mathbf{H}_t^T (\bar{\mathbf{C}}_t^{\mathbf{y}})^{-1} \mathbf{H}_t \mathbf{P}_t \mathbf{C}_{t-1}^{\mathbf{h}}. \tag{38}$$

Even if $(\bar{\mathbf{C}}_t^{\mathbf{y}})^{-1}$ is readily available, evaluating (38) still has a complexity of $\mathcal{O}(Q^2 M^2)$ and storing the off-diagonal blocks of $\mathbf{C}_t^{\mathbf{h}}$ is needed.

$$\mathbf{C}_t^{\mathbf{h}} = \begin{bmatrix} (\mathbf{C}_t^{\mathbf{h}_1})^{-1} & \mathbf{G}_{t,1}^T(\tilde{\mathbf{C}}_t^{\mathbf{y}})^{-1}\mathbf{G}_{t,2} & \cdots & \mathbf{G}_{t,1}^T(\tilde{\mathbf{C}}_t^{\mathbf{y}})^{-1}\mathbf{G}_{t,Q} \\ \mathbf{G}_{t,2}^T(\tilde{\mathbf{C}}_t^{\mathbf{y}})^{-1}\mathbf{G}_{t,1} & (\mathbf{C}_t^{\mathbf{h}_2})^{-1} & \cdots & \mathbf{G}_{t,2}^T(\tilde{\mathbf{C}}_t^{\mathbf{y}})^{-1}\mathbf{G}_{t,Q} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}_{t,Q}^T(\tilde{\mathbf{C}}_t^{\mathbf{y}})^{-1}\mathbf{G}_{t,1} & \mathbf{G}_{t,Q}^T(\tilde{\mathbf{C}}_t^{\mathbf{y}})^{-1}\mathbf{G}_{t,2} & \cdots & (\mathbf{C}_t^{\mathbf{h}_Q})^{-1} \end{bmatrix}^{-1}. \tag{36}$$

2) Variational Approximation and Distributed Regression: We apply the variational inference to approximate the exact regression result in (29) using the following factorized distribution

$$q(\mathbf{h}|\mathcal{D}_{0:t}) = \prod_{j=1}^{Q} p(\mathbf{h}_j|\mathcal{D}_{0:t}) \tag{39}$$

such that the Kullback-Leibler (KL) divergence between $q(\mathbf{h}|\mathcal{D}_{0:t})$ and $p(\mathbf{h}|\mathcal{D}_{0:t})$

$$\text{KL}(q(\mathbf{h}|\mathcal{D}_{0:t})||p(\mathbf{h}|\mathcal{D}_{0:t})) = \int q(\mathbf{h}|\mathcal{D}_{0:t}) \log \frac{q(\mathbf{h}|\mathcal{D}_{0:t})}{p(\mathbf{h}|\mathcal{D}_{0:t})} \text{d}\mathbf{h} \tag{40}$$

is minimized. Mathematically, $q(\mathbf{h}|\mathcal{D}_{0:t})$ can also be found by maximizing the evidence lower bound (ELBO) of the log measurement likelihood $\log(p(\mathbf{y}_t|\mathcal{D}_{0:t-1}))$, which is given by [55]

$$L_t(\boldsymbol{\theta}) = \int q(\mathbf{h}|\mathcal{D}_{0:t}) \log \frac{p(\mathbf{y}_t|\mathbf{h})p(\mathbf{h}|\mathcal{D}_{0:t-1})}{q(\mathbf{h}|\mathcal{D}_{0:t})} \text{d}\mathbf{h}. \tag{41}$$

Here, the model hyperparameter vector $\boldsymbol{\theta}$ is included in the above definition to explicitly show that the ELBO $L_t(\boldsymbol{\theta})$ is indeed dependent on $\boldsymbol{\theta}$.

Note from (29) and (39) that we are approximating a multivariate Gaussian distribution using a factorized distribution. By generalizing the result in Chapter 10.1.2 of [56], we can show that the factorized distribution $q(\mathbf{h}|\mathcal{D}_{0:t})$ that minimizes the KL divergence in (40) and at the same time, maximizes the ELBO in (41) is a factorized Gaussian. In particular, the individual factor $p(\mathbf{h}_j|\mathcal{D}_{0:t})$ is equal to [56]

$$p(\mathbf{h}_j|\mathcal{D}_{0:t}) = \mathcal{N}(\boldsymbol{\mu}_t^{\mathbf{h}_j}, \mathbf{C}_t^{\mathbf{h}_j}),\ j = 1, 2, ..., Q. \tag{42}$$

The mean vector $\boldsymbol{\mu}_t^{\mathbf{h}_j}$ is equal to the one given in (35). In other words, the posterior mean of the inducing points $\mathbf{h}$ obtained via the exact regression (29) is correctly captured by the *approximate* posterior $q(\mathbf{h}|\mathcal{D}_{0:t})$. The covariance matrix $\mathbf{C}_t^{\mathbf{h}_j}$ is given in (37). Subtracting $\mathbf{C}_t^{\mathbf{h}_j}$ from the *marginal* posterior covariance of the inducing points $\mathbf{h}_j$, which can be obtained via applying the partitioned matrix inversion formula [54] to (36), would yield a positive semidefinite matrix. This indicates that the approximate posterior $q(\mathbf{h}|\mathcal{D}_{0:t})$ under-estimates the covariance of the inducing points.

Substituting (42) into (39) yields the desired approximated posterior over the inducing points that will be propagated to the next time step

$$q(\mathbf{h}|\mathcal{D}_{0:t}) = \mathcal{N}(\boldsymbol{\mu}_t^{\mathbf{h}}, \mathbf{C}_t^{\mathbf{h}}) = \prod_{j=1}^{Q} \mathcal{N}(\boldsymbol{\mu}_t^{\mathbf{h}_j}, \mathbf{C}_t^{\mathbf{h}_j}), \tag{43}$$

where $\boldsymbol{\mu}_t^{\mathbf{h}}$ is defined in (30a) and $\mathbf{C}_t^{\mathbf{h}}$ is re-defined here as

$$\mathbf{C}_t^{\mathbf{h}} = \text{diag}(\mathbf{C}_t^{\mathbf{h}_1}, \mathbf{C}_t^{\mathbf{h}_2}, ..., \mathbf{C}_t^{\mathbf{h}_Q}). \tag{44}$$

Note that with the developed approximate regression, the inducing points from different latent sparse GPs would *always* have independent Gaussian posteriors. Moreover, the posterior covariance of $\mathbf{h}_j$ can now be updated *in parallel* using (37) once the matrix $\tilde{\mathbf{C}}_t^{\mathbf{y}}$ becomes available to all the latent functions. The complexity is decreased from $\mathcal{O}(Q^2M^2)$ to $\mathcal{O}(QM^2)$, thanks to removing the dependence among $\mathbf{h}_j$.

Substituting (10), (20), (39) and (42) into (41), we note that all the distributions in the integral in (41) are Gaussian. Therefore, the ELBO $L_t(\boldsymbol{\theta})$ can be shown to be equal to, after applying the results from Appendix A of [1],

$$\begin{aligned} L_t(\boldsymbol{\theta}) = &-\frac{1}{2}\log|2\pi\tilde{\mathbf{C}}_t^{\mathbf{y}}| - \frac{1}{2}\sum_{j=1}^{Q}\log|\mathbf{C}_{t-1}^{\mathbf{h}_j}(\mathbf{C}_t^{\mathbf{h}_j})^{-1}| \\ &- \frac{1}{2}(\mathbf{y}_t - \mathbf{H}_t\boldsymbol{\mu}_t^{\mathbf{h}})^T(\tilde{\mathbf{C}}_t^{\mathbf{y}})^{-1}(\mathbf{y}_t - \mathbf{H}_t\boldsymbol{\mu}_t^{\mathbf{h}}) \\ &- \frac{1}{2}\text{tr}(\mathbf{P}_t^T\mathbf{H}_t^T(\tilde{\mathbf{C}}_t^{\mathbf{y}})^{-1}\mathbf{H}_t\mathbf{P}_t \cdot \text{diag}(\mathbf{C}_t^{\mathbf{h}_1}, ..., \mathbf{C}_t^{\mathbf{h}_Q})) \\ &- \frac{1}{2}\sum_{j=1}^{Q}\text{tr}((\mathbf{C}_{t-1}^{\mathbf{h}_j})^{-1}(\boldsymbol{\mu}_t^{\mathbf{h}_j} - \boldsymbol{\mu}_{t-1}^{\mathbf{h}_j})(\boldsymbol{\mu}_t^{\mathbf{h}_j} - \boldsymbol{\mu}_{t-1}^{\mathbf{h}_j})^T) \\ &- \frac{1}{2}\sum_{j=1}^{Q}\text{tr}((\mathbf{C}_{t-1}^{\mathbf{h}_j})^{-1}(\mathbf{C}_t^{\mathbf{h}_j} - \mathbf{C}_{t-1}^{\mathbf{h}_j})). \end{aligned} \tag{45}$$

It has been pointed out below (42) that the variational approximation in (39) tends to under-estimate the posterior covariance (also see e.g., [56]). This could lead to the *premature* convergence of the approximate posterior $q(\mathbf{h}|\mathcal{D}_{0:t})$. In other words, $q(\mathbf{h}|\mathcal{D}_{0:t})$ might no longer be updated after a few iterations, which may result in degraded model prediction performance because not all the training data are fully explored. The above drawback can be mitigated by applying the back-to-the prior (B2P) or uncertainty-injection (UI) technique [11]. In this work, by taking into accout the properties of the adopted variational approximation (see discussions under (42)), the UI forgetting is used. Specifically, it does not change the posterior mean $\boldsymbol{\mu}_t^{\mathbf{h}_j}$ but it scales up the posterior covariance as

$$\mathbf{C}_t^{\mathbf{h}_j} \leftarrow \frac{1}{\lambda}\mathbf{C}_t^{\mathbf{h}_j}. \tag{46}$$

The forgetting factor λ takes a value in the range $[0.95, 1]$.

The developed approximate regression technique for the sparse multi-output GP model is summarized in **Algorithm 1**. It performs the recursive updating of the posterior over the inducing points using the newly obtained training data $\{\mathbf{x}_{t,n}, \mathbf{y}_{t,n}\}_{n=1}^{N_t}$, given fixed model hyperparameters $\boldsymbol{\theta}$. It can be easily modified to handling missing data. In particular, we just need to remove the rows in $\mathbf{H}_t$ corresponding to the missing data points when executing the algorithm.

It is also straightforward to establish a distributed realization of **Algorithm 1**. Suppose there are Q end-point nodes and a central node. The prediction of the latent function values (lines 2-5 in **Algorithm 1**) can be carried out in parallel at the end-point nodes. The predicted latent function values and their covariance matrices, which are $N_t \times 1$ and $N_t \times N_t$, are fused at the central node (lines 6-9 in **Algorithm 1**). The obtained matrix $\tilde{\mathbf{C}}_t^{\mathbf{y}}$ and vector $(\bar{\mathbf{C}}_t^{\mathbf{y}})^{-1}(\mathbf{y}_t - \bar{\boldsymbol{\mu}}_t^{\mathbf{y}})$, which are $PN_t \times PN_t$ and $PN_t \times 1$, are sent back to the end-point nodes for approximate regression (lines 10-13 in **Algorithm 1**). The amount of data exchange mainly depends on the number of training data points N_t.

Algorithm 1 The approximate model regression algorithm

1: **procedure** ApproxRegression($\boldsymbol{\mu}_{t-1}^{\mathbf{h}}$,$\mathbf{C}_{t-1}^{\mathbf{h}}$,$\mathbf{X}_t$,$\mathbf{y}_t$,$\boldsymbol{\theta}$)
2: **for** $j = 1, 2, ..., Q$ **do** ▷ Latent GP prediction
3: Compute $\mathbf{P}_{t,j}$ and $\mathbf{G}_{t,j}$ using (15) and (33).
4: Compute $\bar{\boldsymbol{\mu}}_t^{\mathbf{g}_j}$, $\bar{\mathbf{C}}_t^{\mathbf{g}_j}$ and $\tilde{\mathbf{C}}_t^{\mathbf{g}_j}$ using (27) and (14b).
5: **end for**
6: Evaluate (24) and (25) to find $\bar{\boldsymbol{\mu}}_t^{\mathbf{y}}$ and $\bar{\mathbf{C}}_t^{\mathbf{y}}$.
7: Calculate $(\bar{\mathbf{C}}_t^{\mathbf{y}})^{-1}(\mathbf{y}_t - \bar{\boldsymbol{\mu}}_t^{\mathbf{y}})$.
8: Find $\tilde{\mathbf{C}}_t^{\mathbf{y}}$ using (21b).
9: Evaluate (45) to obtain the ELBO $L_t(\boldsymbol{\theta})$.
10: **for** $j = 1, 2, ..., Q$ **do** ▷ Approximate regression
11: Compute the posterior mean $\boldsymbol{\mu}_t^{\mathbf{h}_j}$ using (35).
12: Compute the scaled posterior covariance $\mathbf{C}_t^{\mathbf{h}_j}$ using (37) and (46).
13: **end for**
14: **return** $\boldsymbol{\mu}_t^{\mathbf{h}}$ given in (30a), $\mathbf{C}_t^{\mathbf{h}}$ defined in (44) and the ELBO $L_t(\boldsymbol{\theta})$.
15: **end procedure**

C. MPF-based Model Regression and Learning

The model prediction and approximate regression algorithms developed in the previous two subsections both assume that the model hyperparameters are fixed. Nevertheless, training the considered sparse multi-output GP model requires that besides the inducing point values, the model hyperparameters also need to be learned from sequentially arriving data. The inducing points are linearly related to the measurements $\mathbf{y}_t$ given the hyperparameters (see (21a) and the discussions at the end of Section III.A). However, the hyperparameters are nonlinearly related to $\mathbf{y}_t$, which makes analytically deriving their posterior intractable. The joint inference of the inducing points and hyperparameters is thus a Bayesian filtering problem with mixed linear/nonlinear state. We shall apply the MPF [46], [57], also called the Rao-Blackwellized particle filter (RBPF), to this filtering task to achieve online regression and learning of the sparse multi-output GP model.

The MPF-based regression and learning algorithm is developed by following the approach in [57]. Specifically, at the previous time step $t-1$, the joint posterior of the inducing points $\mathbf{h}$ and model hyperparameters $\boldsymbol{\theta}$ can be factorized as

$$p(\mathbf{h}, \boldsymbol{\theta}_{t-1}|\mathcal{D}_{0:t-1}) = p(\mathbf{h}|\boldsymbol{\theta}_{t-1}, \mathcal{D}_{0:t-1})p(\boldsymbol{\theta}_{t-1}|\mathcal{D}_{0:t-1}) \quad (47)$$

where $p(\mathbf{h}|\boldsymbol{\theta}_{t-1}, \mathcal{D}_{0:t-1})$ corresponds to the posterior of $\mathbf{h}$ in (10) evaluated at $\boldsymbol{\theta}_{t-1}$. Here, $\boldsymbol{\theta}_{t-1}$ should be interpreted as the *estimates* of the model hyperparameters. With the MPF, the joint posterior in (47) is approximated empirically using a set of K particles $\{\boldsymbol{\mu}_{t-1}^{\mathbf{h},k}, \mathbf{C}_{t-1}^{\mathbf{h},k}, \boldsymbol{\theta}_{t-1}^k, w_{t-1}^k\}_{k=1}^K$, which indeed correspond to K different sparse multi-output GP models, as

$$p(\mathbf{h}, \boldsymbol{\theta}_{t-1}|\mathcal{D}_{0:t-1}) \approx \sum_{k=1}^K w_{t-1}^k \mathcal{N}(\boldsymbol{\mu}_{t-1}^{\mathbf{h},k}, \mathbf{C}_{t-1}^{\mathbf{h},k})\delta(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_{t-1}^k) \quad (48)$$

where $p(\mathbf{h}|\boldsymbol{\theta}_{t-1}^k, \mathcal{D}_{0:t-1}) = \mathcal{N}(\boldsymbol{\mu}_{t-1}^{\mathbf{h},k}, \mathbf{C}_{t-1}^{\mathbf{h},k})$ according to (10) has been utilized, $\delta(\cdot)$ is the Dirac delta function and w_{t-1}^k are the particle weights satisfying $\sum_{k=1}^K w_{t-1}^k = 1$.

The temporal evolution of $\boldsymbol{\theta}_{t-1}$ follows

$$\boldsymbol{\theta}_t = a\boldsymbol{\theta}_{t-1} + (1-a)\bar{\boldsymbol{\theta}}_{t-1} + \mathbf{v}_{t-1} \quad (49)$$

which was originally proposed in [58]. This allows the exploration of the hyperparameter space using the kernel smoothing with shrinkage while yielding converged hyperparameter estimates [58]–[60].

Here, $a = (3b-1)/(2b)$, with $b \in (0.95, 0.99)$ and $\bar{\boldsymbol{\theta}}_{t-1} = \sum_{k=1}^K w_{t-1}^k \boldsymbol{\theta}_{t-1}^k$ is the empirical mean of $\boldsymbol{\theta}_{t-1}$. The vector $\mathbf{v}_{t-1}$ is drawn from $\mathcal{N}(\mathbf{0}, (1-a^2)\mathbf{V}_{t-1})$, where $\mathbf{V}_{t-1} = \sum_{k=1}^K w_{t-1}^k (\boldsymbol{\theta}_{t-1}^k - \bar{\boldsymbol{\theta}}_{t-1})(\boldsymbol{\theta}_{t-1}^k - \bar{\boldsymbol{\theta}}_{t-1})^T$ is the empirical covariance matrix of $\boldsymbol{\theta}_{t-1}$. From (49), we can write the conditional distribution of $\boldsymbol{\theta}_t$ given $\boldsymbol{\theta}_{t-1}$ as

$$p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}) = \mathcal{N}(a\boldsymbol{\theta}_{t-1} + (1-a)\bar{\boldsymbol{\theta}}_{t-1}, (1-a^2)\mathbf{V}_{t-1}). \quad (50)$$

The MPF algorithm updates the K particles at the current time step t using **Algorithm 2**. It can be seen that once a new set of training data $\{\mathbf{X}_t, \mathbf{y}_t\}$ is obtained, the MPF first updates the hyperparameter estimates for each particle (line 3 in **Algorithm 2**). Then, the MPF refines the posterior of the inducing points for each particle using the updated model hyperparameters and the approximate regression technique given in **Algorithm 1** (line 4 in **Algorithm 2**). In this way, the MPF achieves the joint inference of the inducing points and model hyperparameters. Note that **Algorithm 1** is run K times in each iteration of the MPF, because it employs the empirical approximation of the analytically intractable joint posterior of the inducing points and model hyperparameters (see (48)).

Algorithm 2 The MPF-based online regression and learning

1: **procedure** MPF($\{\boldsymbol{\mu}_{t-1}^{\mathbf{h},k}, \mathbf{C}_{t-1}^{\mathbf{h},k}, \boldsymbol{\theta}_{t-1}^k, w_{t-1}^k\}_{k=1}^K$,$\mathbf{X}_t$,$\mathbf{y}_t$)
2: **for** $k = 1, 2, ..., K$ **do**
3: Draw sample $\boldsymbol{\theta}_t^k$ from $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}^k)$ defined in (50).
4: Run ApproxRegression($\boldsymbol{\mu}_{t-1}^{\mathbf{h},k}, \mathbf{C}_{t-1}^{\mathbf{h},k}$,$\mathbf{X}_t$,$\mathbf{y}_t$,$\boldsymbol{\theta}_t^k$).
5: Weight updating using $w_t^k = \exp(L_t(\boldsymbol{\theta}_t^k))w_{t-1}^k$.
6: **end for**
7: Weight normalization using $w_t^k \leftarrow w_t^k / \sum_{k=1}^K w_t^k$.
8: Resample the particles and set $w_t^k = \frac{1}{K}$ if needed.
9: **return** $\{\boldsymbol{\mu}_t^{\mathbf{h},k}, \mathbf{C}_t^{\mathbf{h},k}, \boldsymbol{\theta}_t^k, w_t^k\}_{k=1}^K$.
10: **end procedure**

Note from line 5 of **Algorithm 2** that different from the standard MPF that uses the measurement likelihood $p(\mathbf{y}_t|\mathcal{D}_{0:t-1})$ to update the particle weights, its lower bound, the ELBO $L_t(\boldsymbol{\theta}_t^k)$, is utilized instead in the proposed MPF. This comes from adopting the variational approximation when

updating the posterior over the inducing points $\mathbf{h}$ (see (39)). With particle resampling (line 8 in **Algorithm 2**), the ELBO is indeed used in a similar way as in [61] for selecting the particles, or equivalently speaking, choosing different sparse multi-output GP models.

When implementing the MPF in the experiments, we employ the effective sample size (ESS)-based residual resampling [62]. When the online training stops, the MPF algorithm outputs the particle with the largest weight. This corresponds to a sparse multi-output model whose hyperparameters are given by $\boldsymbol{\theta}_t^k$ of the output particle, while the mean and covariance of its inducing points are given by $\boldsymbol{\mu}_t^{\mathbf{h},k}$ and $\mathbf{C}_t^{\mathbf{h},k}$ of the output particle [3]..

In the presence of Q end-point nodes and a central node, the MPF in **Algorithm 2** can be realized in a distributed manner as follows. Each end-point node performs the prediction and approximate regression of the jth latent function for *all* the K particles (see lines 2-5 and 10-13 in **Algorithm 1**). The central node fuses the results from the end-point nodes, and send back the updated hyperparameter estimates $\boldsymbol{\theta}_t^k$ as well as the mean and covariance of the inducing points if the particle resampling is performed.

In the developed MPF-based training algorithm, the most time-consuming part is the calculation of $\mathbf{K}_j^{-1}(\mathbf{Z}_j, \mathbf{Z}_j)$ in (15) after the update of the model hyperparameters and $(\mathbf{C}_{t-1}^{\mathbf{h}_j})^{-1}$ for evaluating the ELBO (45). This is valid when the number of training data points N_t is small such that $PN_t < M$, where M is the number of inducing points. Therefore, the proposed online training technique has a computational cost of $\mathcal{O}(M^3)$.

IV. Experiments

In this section, we evaluate the performance of the MPF-based algorithm proposed in Section III for the online regression and learning of the sparse multi-output GP model shown in Fig. 1. Two sets of experiments are conducted, one using synthetic data and the other using real data. In the experiments, all the latent functions are assumed to have zero-mean GP priors (i.e., $m_j(\mathbf{x}) = 0$) with stationary SE kernel (see (3)). The number of particles is fixed at $K = 20$. The UI forgetting factor is $\lambda = 0.99$. The performance metrics used are the root mean square error (RMSE) and negative log-likelihood (NLL).

Initializing the proposed MPF-based training algorithm requires determining K particles so that an empirically approximated joint distribution of the inducing points and model hyperparameters as in (48) can be established. For this purpose, we emulate the practical scenario by buffering a few training data sets that arrive first for algorithm initialization. For the synthetic data-based experiments, the first 3 training data set are stored while the real data-based experiments use the first 80 training data sets for initial configuration. The buffered data are fed to the stochastic gradient-based offline training algorithm originally developed for the COGP [26] to produce the desired K sparse multi-output GP models. We modify the COGP training algorithm so that it no longer uses individual GPs for the output variables. The obtained K particles are assigned a weight of $\frac{1}{K}$ and they are in general different from one another. This is because the COGP training algorithm sets the locations of the inducing points to be the inputs of *randomly* selected training data points. The above initialization scheme is chosen because no information on the initial distribution of the model hyperparameters are available. As a result, we cannot initialize the proposed MPF-based algorithm by sampling the initial distributions as in existing MPF literature (see e.g., [57]).

All the experiments are conducted using MATLAB that runs on a desktop with an Intel(R) Core(TM) i7-7700 3.60GHz CPU and 16GB RAM.

A. Synthetic Data

First, we consider simultaneously modeling the following two well-correlated bivariate functions [19]

$$f_1(x_1, x_2) = 3\cos(x_1) + 4\cos(2x_2) \tag{51a}$$

$$f_2(x_1, x_2) = 2\cos(x_1) + 3\cos(2x_2). \tag{51b}$$

For this purpose, we use the sparse multi-output GP model with $P = 2$ outputs and $Q = 2$ latent sparse functions. Each latent sparse GP has $M = 60$ inducing points, whose locations are randomly selected from the region χ corresponding to the Cartesian product $[-4.5, 4.5] \times [-4.5, 4.5]$.

We generate 60 training data sets first and each one of them is then fed to the proposed MPF-based technique at a time to train in an online manner the aforementioned multi-output GP model. Every training data set contains 30 input-output pairs. The inputs are selected randomly in the region χ. The outputs are produced by first adding to the true function values $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$ independent zero-mean Gaussian noises with standard deviations 0.5 and 0.4. Then, for the first output, the values corresponding to the inputs from the region $[-4, 0] \times [0, 4]$ are removed. For the second output, missing values are created by eliminating those whose inputs are from the region $[0, 4] \times [-4, 0]$. We perform 50 Monte Carlo ensemble runs.

In Fig. 2(a), we plot the predicted values for the function $f_1(x_1, x_2)$ with x_2 being fixed at 2 and the associated 99% confidence interval. The results are produced using the trained sparse multi-output GP model in a certain ensemble run. The true function values are included for comparison. Recall that for the function $f_1(x_1, x_2 = 2)$, its values in the region $x_1 \in [-4, 0]$ are in fact *unobserved*. But the obtained multi-output model is able to accurately reconstruct them, showing that the proposed online training algorithm can effectively exploit the correlation between the two bivariate functions to be modeled. To illustrate this point, the function $f_1(x_1, x_2)$ is modeled *independently* using a single-output GP with exact inference. The obtained regression results are shown in Fig. 2(b). It can be seen that the function values of $f_1(x_1, x_2 = 2)$ when $x_1 \in [-4, 0]$ are poorly recovered, which is somewhat expected as they are missing in the training data. Figs. 2(c) and 2(d) show the predicted values for the function $f_2(x_1, x_2)$ with x_1 set to 2. They are generated by our trained sparse multi-output GP model and an independent single-output GP

[3]It is worthwhile to point out that the outputted $\boldsymbol{\theta}_t^k$ is not necessarily the maximum a posteriori (MAP) estimate of the model hyperparameter vector $\boldsymbol{\theta}$ [63], [64]

targeting $f_2(x_1, x_2)$ only. The observations are very similar to those from Figs. 2(a) and 2(b). Specifically, the unobserved region $x_2 \in [-4, 0]$ is well reconstructed with the multi-output model trained in an online manner using the algorithm proposed in the previous section.

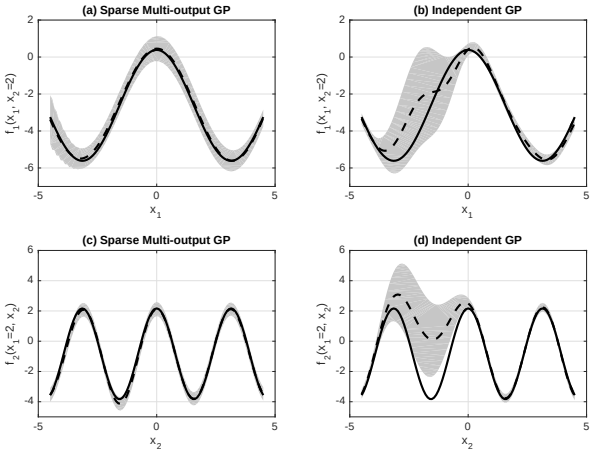

Fig. 2. Comparison of the true function values (solid line) with their predicted version (dashed line) and 99% confidence interval (shaded area) in a certain ensemble run. (a) Results for $f_1(x_1, x_2 = 2)$ from the sparse multi-output GP. (b) Results for $f_1(x_1, x_2 = 2)$ from the independent single-output GP. (c) Results for $f_2(x_1 = 2, x_2)$ from the sparse multi-output GP. (d) Results for $f_2(x_1 = 2, x_2)$ from the independent single-output GP.

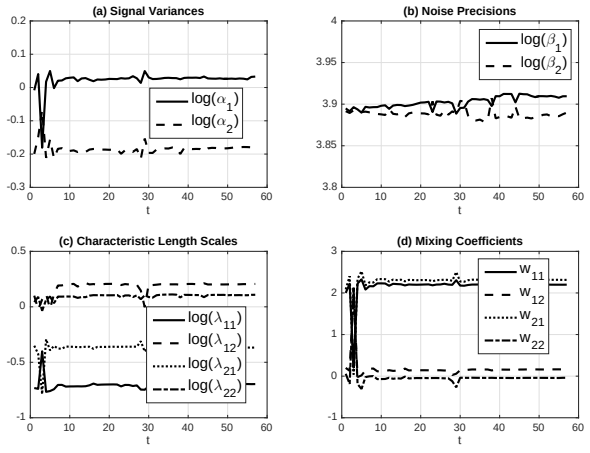

Fig. 3. Model hyperparameter learning over time in a certain ensemble run. (a) Estimation of the latent function variances $\log(a_1)$ and $\log(a_2)$. (b) Estimation of the noise precisions $\log(\beta_1)$ and $\log(\beta_2)$. (c) Estimation of the characteristic length scales $\log(\lambda_{1,1})$, $\log(\lambda_{1,2})$, $\log(\lambda_{2,1})$ and $\log(\lambda_{2,2})$. (d) Estimation of the mixing coefficients $w_{1,1}$, $w_{1,2}$, $w_{2,1}$ and $w_{2,2}$.

Fig. 3 shows the evolution of the model hyperparameter estimates during the ensemble run that produces the results in Fig. 2. At the beginning of the online model training process, the hyperparameter estimates exhibit large variations. This is because the initialized particles correspond to different multi-output GP models and they may yield various ELBOs over the same training data set (see (45)). This leads to different particle weights according to line 5 and 7 of **Algorithm 2**.

TABLE I
AVERAGE RMSE AND NLL FOR THE TWO FUNCTIONS IN (51) WITH UNOBSERVED REGIONS IN TRAINING DATA

	Sparse multi-output GP	Independent GP
RMSE for $f_1(x_1, x_2)$	0.25 ± 0.12	1.64 ± 0.63
RMSE for $f_2(x_1, x_2)$	0.23 ± 0.07	1.19 ± 0.45
NLL for $f_1(x_1, x_2)$	0.39 ± 1.07	4.22 ± 3.36
NLL for $f_2(x_1, x_2)$	0.27 ± 0.67	1.79 ± 1.41

TABLE II
AVERAGE RMSE AND NLL FOR THE TWO FUNCTIONS IN (51) WITHOUT UNOBSERVED REGIONS IN TRAINING DATA

	Sparse multi-output GP	Independent GP
RMSE for $f_1(x_1, x_2)$	0.13 ± 0.04	0.14 ± 0.03
RMSE for $f_2(x_1, x_2)$	0.09 ± 0.02	0.11 ± 0.02
NLL for $f_1(x_1, x_2)$	-0.56 ± 0.27	-0.35 ± 0.56
NLL for $f_2(x_1, x_2)$	-0.76 ± 0.11	-0.76 ± 0.27

As the proposed MPF-based technique outputs the particle with the largest weight, the hyperparameter estimates may vary greatly as they could come from different particles over time. After a few iterations, the MPF tends to converge and the particles would have similar hyperparameter estimates, due to the use of the hyperparameter evolution model (see (49)). Similar observations have been obtained in [51], [52], where the MPF-based training of the single-output GP model is considered.

In Table I, we show the RMSE and NLL when modeling the two bivariate functions in (51). The results are averaged over the 50 Monte Carlo ensemble runs. The inputs of the test data are the grid points of a 101×101 grid with uniformly spaced x_1 coordinates and x_2 coordinates in the interval $[-4.5, 4.5]$. We can see from Table I that in this experiment with missing function values, the trained sparse multi-output GP is superior to the independent GPs in terms of significantly smaller prediction RMSE and NLL. This comes again from the fact that independent single-output GPs fail to recover the function values in the unobserved regions (see e.g., Figs. 2(b) and 2(d)). On the other hand, the trained multi-output GP can effectively utilize the correlation between outputs to reconstruct both functions in (51) well.

Next, the impact of the number of inducing points used in the sparse multi-output GP model on the prediction performance is examined. For this purpose, the experiment that generates Table I is repeated with the number of inducing points M set to be $30, 40, 50, 60, 70$ and 80. We plot in Fig. 4 as a function of M the obtained average RMSEs and NLLs of the trained multi-output GP models in predicting the two function in (51). It can be seen that the prediction performance generally improves with M. This is somewhat expected because the use of more inducing points leads to better coverage of the input space by the multi-output GP model.

We repeat again the experiment that produces the results in Table I but no unobserved regions are assumed this time. The obtained average RMSE and NLL for modeling the two well-correlated functions in (51) are summarized in Table II. It can

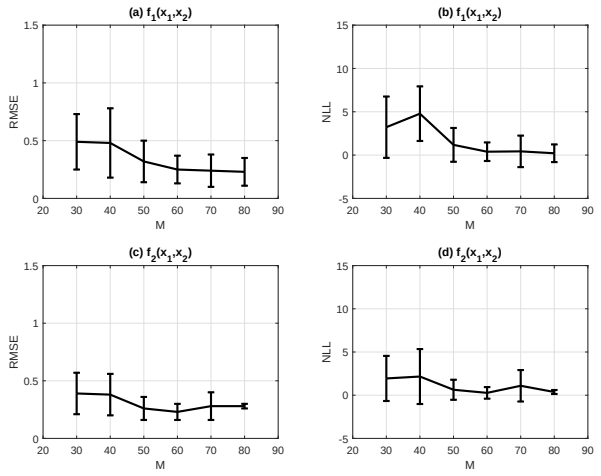

Fig. 4. Prediction RMSE and NLL for $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$ given in (51) from the trained sparse multi-output GP model as a function of the number of the inducing points M. (a) Averaged RMSE with the standard deviation for $f_1(x_1, x_2)$. (b) Averaged NLL with the standard deviation for $f_1(x_1, x_2)$. (c) Averaged RMSE with the standard deviation for $f_2(x_1, x_2)$. (d) Averaged NLL with the standard deviation for $f_2(x_1, x_2)$.

TABLE III
AVERAGE RMSE AND NLL FOR THE TWO FUNCTIONS IN (52) WITHOUT UNOBSERVED REGIONS IN TRAINING DATA

	Sparse multi-output GP	Independent GP
RMSE for $\tilde{f}_1(x_1, x_2)$	0.12 ± 0.05	0.14 ± 0.02
RMSE for $\tilde{f}_2(x_1, x_2)$	0.09 ± 0.04	0.11 ± 0.02
NLL for $\tilde{f}_1(x_1, x_2)$	-0.53 $\pm$ 0.40	-0.36 $\pm$ 0.36
NLL for $\tilde{f}_2(x_1, x_2)$	-0.74 $\pm$ 0.24	-0.74 $\pm$ 0.29

be seen that due to the absence of unobserved regions in the training data, single-output GPs exhibit significantly enhanced prediction performance. But they are still slightly inferior to the trained multi-output GP model, possibly because the latter can explore the correlation between the two functions to be modeled to further improve the prediction performance.

Next, we introduce phase shifts into $f_2(x_1, x_2)$ to decrease its correlation with $f_1(x_1, x_2)$. The resulting two bivariate functions are now given by [19]

$$\tilde{f}_1(x_1, x_2) = 3\cos(x_1) + 4\cos(2x_2), \tag{52a}$$

$$\tilde{f}_2(x_1, x_2) = 2\cos(x_1 + 1) + 3\cos(2x_2 + 1). \tag{52b}$$

The experiment that generates Table II is repeated. The average RMSE and NLL for modeling the two functions in (52) are summarized in Table III. It can be seen that for weakly correlated output variables and without unobserved regions in the training data, the sparse multi-output GP continues to perform slightly better than independent GPs.

B. Robot Inverse Dynamics Data

The robot inverse dynamics problem considered in [17], [23] is a typical multi-output regression problem in robotics, where the mapping from a 21-dimensional input space to a 7-dimensional output space needs to be modeled. The inputs include 7 joint positions, 7 joint velocities and 7 joint accelerations, while the outputs correspond to 7 joint torques. The original dataset contains 48,933 data points collected over a 7-degree-of-freedom anthropomorphic robot arm. We use 40,000 data points for training and 4,400 data points for testing.

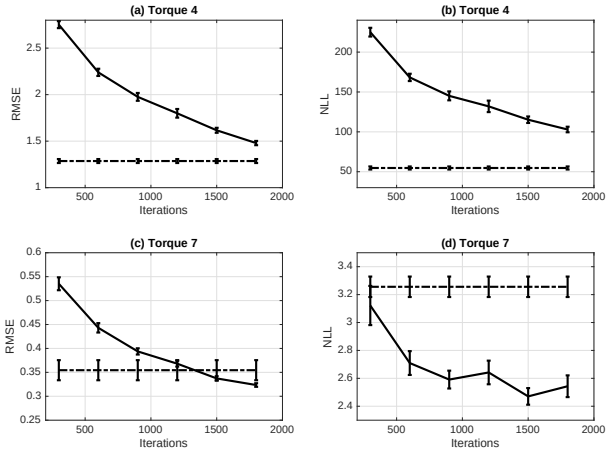

Fig. 5. Comparison of the prediction RMSE and NLL for torques 4 and 7 from the trained sparse multi-output GP model (solid line) and COGP (dashed line). (a) Averaged RMSE with the standard deviation for torque 4. (b) Averaged NLL with the standard deviation for torque 4. (c) Averaged RMSE with the standard deviation for torque 7. (d) Averaged NLL with the standard deviation for torque 7.

First, we train a sparse multi-output GP model with $P = 2$ outputs and $Q = 2$ latent sparse GPs using the proposed MPF-based technique in Section III to model torques 4 and 7 simultaneously. Each latent sparse GP has $M = 500$ inducing points whose locations are randomly selected from the first 80 training data sets. At each time step, a set of 100 input-output pairs that are randomly selected from the 40,000 training data points is fed to the MPF-based training algorithm. In this experiment, after initialization, it takes around 2 hours and 22 minutes on average for the proposed MPF-based algorithm to process 1,800 training data set (i.e., 4.73s per data set).

The prediction RMSE and NLL of the trained sparse multi-output GP are contrasted with those of COGP [26]. The reasons behind we choosing COGP only for comparison are as follows. First, the sparse multi-output GP model considered in this paper is similar to COGP (see the discussions at the end of Sections I and II). Second, it was shown [26], [45] that for the robot inverse dynamics problem, COGP offers an estimation performance close to other multi-output models such as the multi-task GP [34] and VDM-GPDS [45]. In other words, COGP can be considered as a representative of the multi-output GP models that are trained in an offline manner. For all the experiments in this subsection, the COGP is trained using the stochastic gradient-based optimization from [26] with 1000 iterations and each iteration utilizing 200 input-output pairs randomly chosen from the 40,000 training data points.

In this experiment, the realized COGP also has $P = 2$ outputs and $Q = 2$ shared latent sparse GPs, each of which has $M = 500$ inducing points. Different from [26], it no longer uses individual latent GPs for the two output variables.

As COGP is an offline method, the locations of the inducing points are randomly selected from the available 40,000 training data points and they are fixed during the training process. We perform 10 Monte Carlo ensemble runs.

The results are summarized in Fig. 5. We plot, as function of the number of iterations (i.e., the number of data collections obtained over time), the prediction RMSE and NLL for torques 4 and 7 from the trained sparse multi-output GP. For comparison, the results from COGP are also included. They are independent of the number of iterations, as COGP is trained in an offline manner. It can be seen that as the number of training data sets increases, the prediction performance of the trained sparse multi-output GP, especially the RMSE, would gradually approach that of COGP. This is somewhat expected, because with the sparse multi-output GP being exposed to more data points, a better prediction model can be obtained.

On the other hand, for the 4th torque, COGP still offers a smaller NLL than the sparse multi-output GP after 1,800 training data sets have been applied. This indicates that COGP is more *certain* about its predictions. The performance degradation might come from that the locations of the inducing points for COGP are selected from all the 40,000 training data points while for the sparse multi-output GP, they are chosen from the first 8,000 data points. We repeat the experiment to verify the above hypothesis. The newly obtained COGP has inducing points whose locations are from the first 8,000 data points. The average RMSE and NLL for the 4th torque now increase to 2.02 ± 0.12 and 134.77 ± 16.25, which are higher than those from the trained sparse multi-output GP after only 1,000 data sets have been used. Thus, it will be interesting to investigate in future works adapting the locations of the inducing points for further performance improvement.

In the second experiment, we study the effect of the number of latent functions Q on the prediction performance. For this purpose, the simulation that produces Fig. 5 is repeated but this time, the use of $Q = 3$ and $Q = 4$ latent GPs is considered. The obtained average prediction RMSE and NLL from the learned sparse multi-output GP model and COGP with $Q = 3$ and $Q = 4$ latent functions are given in Tables IV and V. In particular, the sparse multi-output GP model is trained using the proposed MPF-based technique and 1,800 data sets, each of which contains 100 input-output pairs. When $Q = 3$, it takes around 3 hours and 52 minutes for the proposed MPF-based algorithm to process 1,800 training data set (i.e., 7.73s per data set). The training time is increased to 5 hours and 43 minutes (i.e., 11.43s per data set) when $Q = 4$.

Comparing the results in Tables IV and V with those in Fig. 5 where only $Q = 2$ latent GPs are adopted, we note that using $Q = 3$ or 4 latent GPs leads to poorer predicton accuracy for both the sparse multi-output GP model and COGP. The performance degradation may be due to that using larger number of latent functions increases the model complexity, which could reduce the generalization capability of the trained multi-ouput models.

In the third experiment, we attempt to learn simultaneously torques 2, 3, 4 and 7, where torques 2 and 3 are negatively correlated while torques 4 and 7 are positively correlated [45]. For this purpose, we train a sparse multi-output GP model with $P = 4$ output variables and $Q = 4$ latent GPs using the proposed MPF-based algorithm. Each latent function has $M = 500$ inducing points and 1,800 data sets is fed to the training algorithm, each of which consists of 100 input-output pairs randomly selected from the available 40,000 training data points. For comparison, a COGP without individual latent GPs for the output variables is trained [26]. It also has $P = 4$ output variables and $Q = 4$ latent GPs, each of which has has $M = 500$ inducing points. We conduct five Monte Carlo ensemble runs. Simulations show that the average training time for the MPF-based algorithm is 5 hours and 57 minutes (i.e., 11.9s per data set).

TABLE IV
AVERAGE RMSE AND NLL FOR TORQUES 4 AND 7 WITH $Q = 3$ LATENT FUNCTIONS

	Sparse multi-output GP	COGP
RMSE for torque 4	3.17 ± 1.00	2.13 ± 0.11
RMSE for torque 7	0.63 ± 0.16	0.47 ± 0.02
NLL for torque 4	89.20 ± 27.70	119.78 ± 17.81
NLL for torque 7	3.17 ± 0.61	4.89 ± 0.71

TABLE V
AVERAGE RMSE AND NLL FOR TORQUES 4 AND 7 WITH $Q = 4$ LATENT FUNCTIONS

	Sparse multi-output GP	COGP
RMSE for torque 4	3.08 ± 0.43	2.20 ± 0.09
RMSE for torque 7	0.70 ± 0.12	0.49 ± 0.02
NLL for torque 4	84.55 ± 30.18	132.50 ± 17.80
NLL for torque 7	3.26 ± 0.78	5.52 ± 0.91

The obtained average prediction RMSE and NLL from the trained sparse multi-output GP model and COGP are summarized in Table VI. We can see that the learned COGP provides better prediction performance for torques 4 and 7 but the trained sparse multi-output GP model is superior in modeling torques 2 and 3. Moreover, comparing Tables V and VI reveals that the prediction accuracy for torques 4 and 7 improves, although the two experiments both use 4 latent functions. This may be because in this experiment, four torques are learned jointly with $Q = 4$ latent GPs while in the previous experiment, only torques 4 and 7 are modeled and the selected model might be too complicated to generalize well. On the other hand, comparing Tables VI with Fig. 5 indicates that modeling torques 4 and 7 together with torques 2 and 3 is inferior in terms of prediction accuracy to they being modeled

TABLE VI
AVERAGE RMSE AND NLL FOR TORQUES 2, 3, 4 AND 7

	Sparse multi-output GP	COGP
RMSE for torque 2	2.56 ± 0.11	3.04 ± 0.04
RMSE for torque 3	1.63 ± 0.09	1.77 ± 0.05
RMSE for torque 4	2.27 ± 0.14	1.92 ± 0.13
RMSE for torque 7	0.50 ± 0.04	0.46 ± 0.02
NLL for torque 2	210.13 ± 22.04	245.18 ± 4.87
NLL for torque 3	42.64 ± 2.71	58.85 ± 2.82
NLL for torque 4	77.99 ± 5.93	72.64 ± 8.81
NLL for torque 7	6.76 ± 0.10	4.66 ± 0.57

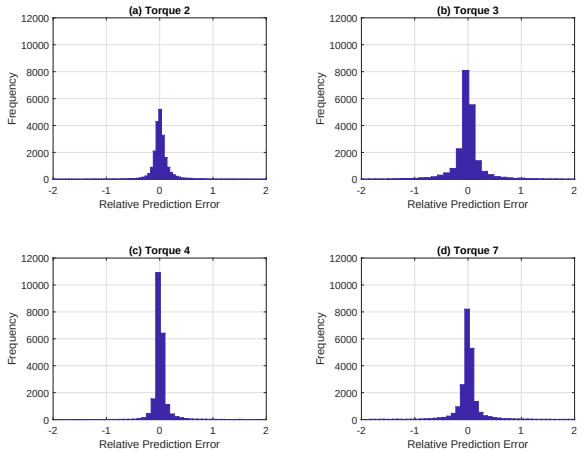

Fig. 6. Histograms of the relative prediction errors for torques 2, 3, 4 and 7 from the trained sparse multi-output GP model. (a) Histogram of the relative prediction errors for torque 2. (b) Histogram of the relative prediction errors for torque 3. (c) Histogram of the relative prediction errors for torque 4. (d) Histogram of the relative prediction error for torque 7.

separately using multi-output models with $Q = 2$ latent GPs. It would be interesting to examine in future works the effect of different model configurations on the prediction performance.

In Fig. 6, we plot the histograms of the *relative* prediction errors for torques 2, 3, 4 and 7 from the trained sparse multi-output GP model. The results from all the five Monte Carlo ensemble runs are included and as a result, each subfigure contains a total number of $5 \times 4,400$ relative prediction errors for a particular torque. It can be seen from Fig. 6 that the histograms of the relative prediction errors for the four torques in consideration are all bell-shaped. Besides, a large portion of the relative prediction errors are close to zero, which means that many prediction errors are small compared to the true torque values. In other words, the four torques are well predicted using the sparse multi-output GP model trained in an online manner via the proposed MPF-based algorithm.

V. Conclusions

This paper demonstrated the feasibility of performing online training of a sparse multi-output GP model using sequentially arriving data. The considered GP model produces multiple correlated outputs by linearly combining latent sparse GPs, each of which uses a separate set of inducing points to achieve sparse approximation. A novel MPF-based algorithm that can perform joint inference of the inducing point values and model hyperparameters was established. It incorporates a newly developed approximate regression approach that updates the posterior over the inducing points given the hyperparameters. The approximate regression method guarantees that the inducing points of different latent sparse GPs are always independent, which facilitates the *distributed implementation* of the proposed MPF-based training algorithm. Experiments using simulated and real data verified the effectiveness of the developed training algorithm. Specifically, the trained sparse multi-output GP can exploit the correlation between output variables to improve prediction accuracy in the presence of missing values. Moreover, with more iterations, its performance approaches that of the existing multi-output GP models obtained via offline training using all the training data.

TABLE VII
Symbols and Notations

Symbol	Explanation
D	model input dimension
P	number of model output variables
Q	number of latent GPs
M	number of inducing points for each latent GP
$\mathbf{z}_{k,j}$	$D \times 1$ location of inducing point $g(\mathbf{z}_{k,j})$ of latent GP j
$\mathbf{Z}_j$	$D \times M$ location matrix of inducing points of latent GP j, $\mathbf{Z}_j = [\mathbf{z}_{1,j}, \mathbf{z}_{2,j}, ..., \mathbf{z}_{M,j}]$
$\mathbf{h}_j$	$M \times 1$ inducing point vector of latent GP j, $\mathbf{h}_j = [g_j(\mathbf{z}_{1,j}), g_j(\mathbf{z}_{2,j}), ..., g_j(\mathbf{z}_{M,j})]^T$
$\mathbf{m}_j(\mathbf{Z}_j)$	$M \times 1$ prior mean vector of $\mathbf{h}_j$
$\mathbf{K}_j(\mathbf{Z}_j, \mathbf{Z}_j)$	$M \times M$ prior covariance of $\mathbf{h}_j$
$\mathbf{h}$	$MQ \times 1$ composite inducing point vector, $\mathbf{h} = [\mathbf{h}_1^T, \mathbf{h}_2^T, ..., \mathbf{h}_M^T]^T$
$\boldsymbol{\mu}_{t-1}^{\mathbf{h}_j}$	$M \times 1$ posterior mean vector of $\mathbf{h}_j$ at time $t-1$
$\mathbf{C}_{t-1}^{\mathbf{h}_j}$	$M \times M$ posterior covariance of $\mathbf{h}_j$ at time $t-1$
$\boldsymbol{\mu}_{t-1}^{\mathbf{h}}$	$MQ \times 1$ posterior mean vector of $\mathbf{h}$ at time $t-1$, $\boldsymbol{\mu}_{t-1}^{\mathbf{h}} = [(\boldsymbol{\mu}_{t-1}^{\mathbf{h}_1})^T, (\boldsymbol{\mu}_{t-1}^{\mathbf{h}_2})^T, ..., (\boldsymbol{\mu}_{t-1}^{\mathbf{h}_Q})^T]^T$
$\mathbf{C}_{t-1}^{\mathbf{h}}$	$MQ \times MQ$ posterior covariance of $\mathbf{h}$ at time $t-1$, $\mathbf{C}_{t-1}^{\mathbf{h}} = \text{diag}(\mathbf{C}_{t-1}^{\mathbf{h}_1}, \mathbf{C}_{t-1}^{\mathbf{h}_2}, ..., \mathbf{C}_{t-1}^{\mathbf{h}_Q})$
$\mathbf{x}_{t,n}$	nth input vector at time t, $D \times 1$
$\mathbf{X}_t$	$D \times N_t$ input matrix at time t, $\mathbf{X}_t = [\mathbf{x}_{t,1}, \mathbf{x}_{t,2}, ..., \mathbf{x}_{t,N_t}]$
$\mathbf{g}_{t,j}$	$N_t \times 1$ output vector of latent GP j at time t, $\mathbf{g}_{t,j} = [g_j(\mathbf{x}_{t,1}), g_j(\mathbf{x}_{t,2}), ..., g_j(\mathbf{x}_{t,N_t})]^T$
$\mathbf{g}_t$	$QN_t \times 1$ composite output vector of latent GPs at time t, $\mathbf{g}_t = [\mathbf{g}_{t,1}^T, \mathbf{g}_{t,2}^T, ..., \mathbf{g}_{t,Q}^T]^T$
$\tilde{\boldsymbol{\mu}}_t^{\mathbf{g}_j}$	$N_t \times 1$ conditional mean vector of $\mathbf{g}_{t,j}$ given $\mathbf{h}_j$
$\bar{\boldsymbol{\mu}}_t^{\mathbf{g}_j}$	$N_t \times 1$ predictive mean vector of $\mathbf{g}_{t,j}$
$\bar{\boldsymbol{\mu}}_t^{\mathbf{g}}$	$QN_t \times 1$ predictive mean vector of $\mathbf{g}_t$, $\bar{\boldsymbol{\mu}}_t^{\mathbf{g}} = [(\bar{\boldsymbol{\mu}}_t^{\mathbf{g}_1})^T, (\bar{\boldsymbol{\mu}}_t^{\mathbf{g}_2})^T, ..., (\bar{\boldsymbol{\mu}}_t^{\mathbf{g}_Q})^T]^T$
$\tilde{\mathbf{C}}_t^{\mathbf{g}_j}$	$N_t \times N_t$ conditional covariance of $\mathbf{g}_{t,j}$ given $\mathbf{h}_j$
$\bar{\mathbf{C}}_t^{\mathbf{g}_j}$	$N_t \times N_t$ predictive covariance of $\mathbf{g}_{t,j}$
$\bar{\mathbf{C}}_t^{\mathbf{g}}$	$QN_t \times QN_t$ predictive covariance of $\mathbf{g}_t$ $\bar{\mathbf{C}}_t^{\mathbf{g}} = \text{diag}(\bar{\mathbf{C}}_t^{\mathbf{g}_1}, \bar{\mathbf{C}}_t^{\mathbf{g}_2}, ..., \bar{\mathbf{C}}_t^{\mathbf{g}_Q})$
$\mathbf{m}_j(\mathbf{X}_t)$	$N_t \times 1$ prior mean vector of $\mathbf{g}_{t,j}$
$\mathbf{K}_j(\mathbf{X}_t, \mathbf{X}_t)$	$N_t \times N_t$ prior covariance of $\mathbf{g}_{t,j}$
$\mathbf{K}_j(\mathbf{X}_t, \mathbf{Z}_j)$	$N_t \times M$ prior cross covariance between $\mathbf{g}_{t,j}$ and $\mathbf{h}_j$
$\mathbf{P}_{t,j}$	$N_t \times M$ matrix, $\mathbf{P}_{t,j} = \mathbf{K}_j(\mathbf{X}_t, \mathbf{Z}_j)\mathbf{K}_j^{-1}(\mathbf{X}_t, \mathbf{X}_t)$
$\mathbf{P}_t$	$QN_t \times QM$ matrix, $\mathbf{P}_t = \text{diag}(\mathbf{P}_{t,1}, \mathbf{P}_{t,2}, ..., \mathbf{P}_{t,Q})$
$\mathbf{y}_{t,n}$	$P \times 1$ model output vector for input $\mathbf{x}_{t,n}$
$\mathbf{y}_t$	$PN_t \times 1$ composite model output vector at time t $\mathbf{y}_t = [\mathbf{y}_{t,1}, \mathbf{y}_{t,2}, ..., \mathbf{y}_{t,N_t}]^T$
$\mathbf{H}_t$	$PN_t \times QN_t$ model observation matrix at time t
$\boldsymbol{\Lambda}_t$	$PN_t \times PN_t$ measurement noise covariance at time t
$\tilde{\boldsymbol{\mu}}_t^{\mathbf{y}}$	$PN_t \times 1$ conditional mean vector of $\mathbf{y}_t$ given $\mathbf{h}$
$\tilde{\mathbf{C}}_t^{\mathbf{y}}$	$PN_t \times PN_t$ conditional covariance of $\mathbf{y}_t$ given $\mathbf{h}$
$\bar{\boldsymbol{\mu}}_t^{\mathbf{y}}$	$PN_t \times 1$ predictive mean vector of $\mathbf{y}_t$
$\bar{\mathbf{C}}_t^{\mathbf{y}}$	$PN_t \times PN_t$ predictive covariance of $\mathbf{y}_t$
$\mathbf{G}_{t,j}$	$PN_t \times M$ matrix

Appendix

The symbols and notations used are summarized in Table VII.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and anonymous reviewers for their comments that helped improve this paper. The authors acknowledge the support from the China Scholarship Council (CSC) for Le Yang and Ke Wang.

REFERENCES

[1] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[2] F. Pérez-Cruz, S. V. Vaerenbergh, J. Murillo-Fuentes, M. Lázaro-Gredilla, and I. Santamaría, "Gaussian processes for nonlinear signal processing," *IEEE Signal Process. Mag.*, vol. 40, pp. 40–50, July 2013.

[3] H. Wymeersch, S. Maranò, W. M. Gifford, and M. Z. Win, "A machine learning approach to ranging error mitigation for UWB localization," *IEEE Trans. Communications*, vol. 60, pp. 1719–1728, June 2012.

[4] B. Laufer-Goldshtein, R. Talmon, and S. Gannot, "Semi-supervised source localization on multiple manifolds with distributed microphones," *IEEE/ACM Trans. Audio, Speech and Language Process.*, vol. 25, pp. 1477–1491, April 2017.

[5] F. Yin, Y. Zhao, F. Gunnarsson, and F. Gustafsson, "Received-signal-strength threshold optimization using Gaussian processes," *IEEE Trans. Signal Process.*, vol. 65, pp. 2164–2177, April 2017.

[6] H. He and W.-C. Siu, "Single image super-resolution using Gaussian process regression," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 449–456.

[7] A. Ranganathan, M.-H. Yang, and J. Ho, "Online sparse Gaussian process regression and its applications," *IEEE Trans. Image Process.*, vol. 20, pp. 391–404, February 2011.

[8] Y. Saatçi, R. Turner, and C. E. Rasmussen, "Gaussian process change point models," in *Proc. Int. Conf. Machine Learning (ICML)*, 2010, pp. 927–934.

[9] C. Guestrin, A. Krause, and A. Singh, "Near-optimal sensor placements in Gaussian processes," in *Proc. Int. Conf. Machine Learning (ICML)*, 2005, pp. 265–272.

[10] A. Carron, M. Todescato, R. Carli, L. Schenato, and G. Pillonetto, "Multi-agents adaptive estimation and converage control using Gaussian regression," in *Proc. European Control Conf. (ECC)*, 2015, pp. 2490–2495.

[11] S. V. Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaría, "Kernel recursive least-squares tracker for time-varying regression," *IEEE Trans. Neural Networks and Learning Systems*, vol. 23, pp. 1313–1326, August 2012.

[12] L. S. Muppirisetty, T. Svensson, and H. Wymeersch, "Spatial wireless channel prediction under location uncertainty," *IEEE Trans. Wireless Communications*, vol. 15, pp. 1031–1044, February 2016.

[13] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models," *Autonomous Robots*, vol. 27, pp. 75–90, July 2009.

[14] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen, "Bayesian inference and learning in Gaussian process state-space models with particle MCMC," in *Proc. Adv. in Neural Information Process. Syst. (NIPS)*, 2013, pp. 3156–3164.

[15] Y. Wang, M. A. Brubaker, B. Chaib-draa, and R. Urtasun, "Bayesian filtering with online Gaussian process latent variable models," in *Proc. Conf. Uncertainty in Artificial Intelligence (UAI)*, 2013, pp. 849–857.

[16] M. Knotters, D. J. Brus, and J. O. Voshaar, "A comparison of kriging, co-kriging and kriging combined with regression for spatial interpolation of horizon depth with censored observations," *Geoderma*, vol. 67, pp. 227–246, August 1995.

[17] S. Vijayakumar and S. Schaal, "Locally weighted projection regression : An $O(n)$ algorithm for incremental real time learning in high dimensional space," in *Proc. Int. Conf. Machine Learning (ICML)*, 2000, pp. 1079–1086.

[18] M. A. Osborne, S. J. Roberts, A. Rogers, S. D. Ramchurn, and N. R. Jennings, "Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes," in *Proc. Int. Conf. Information Process. in Sensor Networks (IPSN)*, April 2008, pp. 109–120.

[19] B. Wang and T. Chen, "Gaussian process regression with multiple response variables," *Chemometrics and Intelligent Laboratory Systems*, vol. 142, pp. 159–165, March 2015.

[20] S. Sun, C. Zhang, and G. Yu, "A Bayesian network approach to traffic flow forecasting," *IEEE Trans. Intelligent Transportation Syst.*, vol. 7, pp. 124–132, March 2006.

[21] I. Bilionis, N. Zabaras, B. Konomi, and G. Lin, "Multi-output separable Gaussian process: Towards an efficient, fully Bayesian paradigm for uncertainty quantification," *J. of Computational Physics*, vol. 241, pp. 212–239, May 2013.

[22] B. Zhang, B. Konomi, H. Sang, G. Karagiannis, and G. Lin, "Full scale multi-output Gaussian process emulator with nonseparable auto-covariance functions," *J. of Computational Physics*, vol. 300, pp. 623–642, Nov. 2015.

[23] K. Chai, C. K. I. Williams, S. Klanke, and S. Vijayakumar, "Multi-task Gaussian process learning of robot inverse dynamics," in *Proc. Adv. in Neural Information Process. Syst. (NIPS)*, 2009, pp. 265–272.

[24] J. Yuan, K. Wang, T. Yu, and M. Fang, "Reliable multi-objective optimization of high-speed WEDM process based on Gaussian process regression," *Intl. J. Machine Tools and Manufacture*, vol. 48, pp. 47–60, January 2008.

[25] T. Chen, K. Hadinoto, W. J. Yan, and Y. F. Ma, "Efficient meta-modelling of complex process simulations with time-space-dependent outputs," *Computers and Chemical Engineering*, vol. 35, pp. 502–509, March 2011.

[26] T. V. Nguyen and E. V. Bonilla, "Collaborative multi-output Gaussian processes," in *Proc. Conf. Uncertainty in Artificial Intelligence (UAI)*, 2014, pp. 643–652.

[27] P. Boyle and M. Frean, "Dependent Gaussian processes," in *Proc. Adv. in Neural Information Process. Syst. (NIPS)*, 2005, pp. 217–224.

[28] M. Alvarez and N. Lawrence, "Sparse convolved Gaussian processes for multi-output regression," in *Proc. Adv. in Neural Information Process. Syst. (NIPS)*, 2009, pp. 57–64.

[29] ——, "Computationally efficient convolved multiple output Gaussian processes," *J. of Machine Learning Research*, vol. 12, pp. 1459–1500, May 2011.

[30] Y. W. Teh, M. Seeger, and M. I. Jordan, "Semiparametric latent factor models," in *Proc. Intl. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2005, pp. 330–340.

[31] B. M. Yu, J. Cunningham, G. Santhanam, S. Ryu, K. Shenoy, and M. Sahani, "Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity," in *Proc. Adv. in Neural Information Process. Syst. (NIPS)*, 2009, pp. 1881–1888.

[32] A. G. Wilson, D. A. Knowles, and Z. Ghahramani, "Gaussian process regression networks," in *Proc. Intl. Conf. Machine Learning (ICML)*, 2012, pp. 599–606.

[33] T. V. Nguyen and E. V. Bonilla, "Efficient variational inference for Gaussian process regression networks," in *Proc. Intl. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2013, pp. 472–480.

[34] E. Bonilla, K. M. Chai, and C. K. I. Williams, "Multi-task Gaussian process prediction," in *Proc. Adv. in Neural Information Process. Syst. (NIPS)*, 2008, pp. 153–160.

[35] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *J. of Machine Learning Research*, vol. 6, pp. 1939–1959, Dec. 2005.

[36] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Proc. Adv. in Neural Information Process. Syst. (NIPS)*, 2006, pp. 1257–1264.

[37] ——, "Local and global sparse Gaussian process approximations," in *Proc. Intl. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2007, pp. 524–531.

[38] M. Huber, "Recursive Gaussian process: On-line regression and learning," *Pattern Recognition Letters*, vol. 45, pp. 85–91, August 2014.

[39] M. Titsias, "Variational model selection for sparse Gaussian process regression," School of Computer Science, University of Manchester, Tech. Rep., 2009.

[40] ——, "Variational learning of inducing variables in sparse Gaussian processes," in *Proc. Intl. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2009, pp. 567–574.

[41] M. Titsias and N. Lawrence, "Bayesian Gaussian process latent variable model," in *Proc. Intl. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2010, pp. 844–851.

[42] J. Hensman, N. Fusi, and N. Lawrence, "Gaussian processes for big data," in *Proc. Conf. Uncertainty in Artificial Intelligence (UAI)*, 2013, pp. 282–290.

[43] Y. Gal, M. van der Wilk, and C. E. Rasmussen, "Distributed variational inference in sparse Gaussian process regression and latent variable models," in *Proc. Adv. in Neural Information Process. Syst. (NIPS)*, 2014, pp. 3257–3265.

[44] M. Bauer, M. van der Wilk, and C. E. Rasmussen, "Understanding probabilistic sparse Gaussian process approximations," in *Proc. Adv. in Neural Information Process. Syst. (NIPS)*, 2016, pp. 1533–1541.

[45] J. Zhao and S. Sun, "Variational dependent multi-output Gaussian process dynamic systems," *J. of Machine Learning Research*, vol. 17, pp. 1–36, Aug. 2016.
[46] T. Schön, F. Gustafusson, and P.-J. Nordlund, "Marginalized particle filters for mixed linear/nonlinear state-space models," *IEEE Tran. Signal Process.*, vol. 53, pp. 2279–2289, July 2005.
[47] L. Csato and M. Opper, "Sparse online Gaussian processes," *Neural Computation*, vol. 14, pp. 641–688, March 2002.
[48] D. Nguyen-Tuong, J. Peters, and M. Seeger, "Local Gaussian process regression for real time online model learning and control," in *Proc. Adv. in Neural Information Process. Syst. (NIPS)*, 2008, pp. 1193–1200.
[49] ——, "Model learning with local Gaussian process regression," *Advanced Robotics*, vol. 23, pp. 2015–2034, July 2009.
[50] H. Bijl, J. van Wingerden, T. B. Schön, and M. Verhaegen, "Online sparse Gaussian process regression using FITC and PITC approximations," in *Proc. Intl. Federation Automatic Control (IFAC)*, 2015, pp. 703–708.
[51] Y. Wang and B. Chaib-draa, "A marginalized particle Gaussian process regression," in *Proc. Adv. in Neural Information Process. Syst. (NIPS)*, 2012, pp. 1187–1195.
[52] ——, "An online Bayesian filtering framework for Gaussian process regression: Application to global surface temperature analysis," *Expert Systems with Applications*, vol. 67, pp. 285–295, Jan. 2017.
[53] S. M. Kay, *Fundamentals of Statistical Signal Processing, Estimation Theory*. Upper Saddle River, NJ, USA: Prentice Hall, 1993.
[54] L. L. Scharf, *Statistical Signal Processing, Detection, Estimation and Time Series Analysis*. Reading, MA: Addison-Wesley, 1991.
[55] M. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, pp. 183–233, Nov. 1999.
[56] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.
[57] P. Li, R. Goodball, and V. Kadirkamanathan, "Estimation of parameters in a linear state space model using a Rao-Blackwellised particle filter," *IEE Proceedings on Control Theory Applications*, vol. 151, pp. 727–738, Nov. 2004.
[58] J. Liu and M. West, "Combined parameter and state estimation in simulation-based filtering," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds. New York: Springer-Verlag, 2001, pp. 197–223.
[59] N. Kantas, A. Doucet, S. Singh, and J. Maciejowski, "An overview of sequential Monte Carlo methods for parameter estimation in general state-space models," in *Proc. IFAC Symposium on System Identification*, 2009, pp. 774–785.
[60] C. Nemeth, P. Fearnhead, and L. Mihaylova, "Sequential Monte Carlo methods for state and parameter estimation in abruptly changing environments," *IEEE Trans. Signal Process.*, vol. 62, pp. 1245–1255, March 2014.
[61] M. Beal and Z. Ghahramani, "The variational Bayesian EM algorithm for incomplete data: With application to scoring graphical model structures," in *Bayesian Statistics 7*, J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, and M. West, Eds. Oxford: Oxford University Press, 2003, pp. 453–463.
[62] T. Li, M. Bolic, and P. M. Djuric, "Resampling methods for particle filters: classification, implementation and strategies," *IEEE Signal Process. Magazine*, vol. 32, pp. 70–86, May 2015.
[63] O. Cappe, S. J. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proc. of the IEEE*, vol. 95, pp. 899–924, May 2007.
[64] S. Saha, Y. Boers, H. Driessen, P. Mandal, and A. Bagchi, "Particle based MAP state estimation: A comparison," in *Proc. Intl. Conf. Information Fusion (FUSION)*, 2009, pp. 278–283.