



<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *CAMSAP 2015, December 13–16, Cancún, Mexico*.

Citation for the original published paper:

Svensson, A., Dahlin, J., Schön, T B. (2015)

Marginalizing Gaussian process hyperparameters using sequential Monte Carlo.

In: *Proc. 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing* (pp. 477-480). Piscataway, NJ: IEEE

<https://doi.org/10.1109/CAMSAP.2015.7383840>

N.B. When citing this work, cite the original published paper.

Copyright 2015 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-265654>

Marginalizing Gaussian Process Hyperparameters Using Sequential Monte Carlo

Andreas Svensson*, Johan Dahlin**, Thomas B. Schön*
 *Department of Information Technology, Uppsala University, Sweden
 {andreas.svensson, thomas.schon}@it.uu.se
 **Division of Automatic Control, Linköping University, Sweden
 johan.dahlin@liu.se

Abstract—Gaussian process regression is a popular method for non-parametric probabilistic modeling of functions. The Gaussian process prior is characterized by so-called hyperparameters, which often have a large influence on the posterior model and can be difficult to tune. This work provides a method for numerical marginalization of the hyperparameters, relying on the rigorous framework of sequential Monte Carlo. Our method is well suited for online problems, and we demonstrate its ability to handle real-world problems with several dimensions and compare it to other marginalization methods. We also conclude that our proposed method is a competitive alternative to the commonly used point estimates maximizing the likelihood, both in terms of computational load and its ability to handle multimodal posteriors.

I. INTRODUCTION

The Gaussian process (GP) is a non-parametric probabilistic model that can be used to model an unknown nonlinear function $f(\cdot)$ from observed input data x and (noisy) output data $y = f(x)$. No explicit form of $f(\cdot)$ is assumed, but some assumptions on $f(\cdot)$ are encoded through the GP prior and a mean function $m_\theta(x)$, a covariance function $\kappa_\theta(x, x')$, and their so-called hyperparameters $\theta \in \Theta$. In mathematical terms, f is a priori modeled to be distributed as

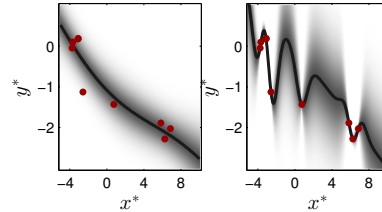
$$f(x) \sim \mathcal{GP}\left(m_\theta(x), \kappa_\theta(x, x')\right), \quad (1)$$

i.e., an infinite-dimensional Gaussian distribution. See [1] for a more general introduction to GPs.

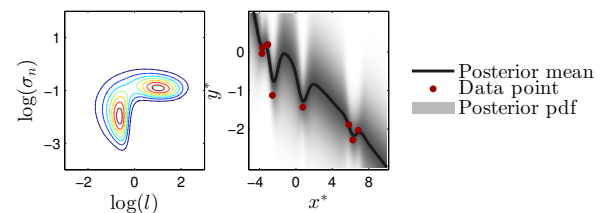
The posterior distribution over $f(\cdot)$ given data (y, x) is also a GP. This is due to the conjugacy property of the Gaussian distribution. The posterior is often greatly influenced by the choice of hyperparameters θ , which typically are unknown. We therefore propose a method to *marginalize* the hyperparameters in GPs. Marginalization can be seen as averaging over the range of hyperparameters supported by the data and by the prior; θ can be integrated out by treating it as a random variable with prior $p(\theta)$ and likelihood $p(y|x, \theta)$, giving rise to the posterior $p(\theta|y, x) \propto p(y|x, \theta)p(\theta)$. For example, the predictive distribution is computed by

$$p(y^*|x^*, y, x) = \int p(y^*|x^*, y, x, \theta)p(\theta|y, x)d\theta, \quad (2)$$

which unfortunately is analytically intractable. However, using a Monte Carlo method to obtain N (weighted) samples



(a) Gaussian process regression for the data set defined by the red dots, using two different point estimates for the hyperparameters, each corresponding to a local minimum in (b, left).



(b) Left: the (multimodal) hyperparameter posterior conditional on the 9 data points. Right: the posterior using the proposed method (which marginalizes the hyperparameters, and thus handles the multimodality).

Fig. 1. A small example illustrating the influence of the hyperparameters in the GP prior to the posterior estimate.

$\{w^{(i)}, \theta^{(i)}\}_{i=1}^N$ of the distribution $p(\theta|y, x)$, the predictive distribution (2) can be approximated by

$$\hat{p}(y^*|x^*, y, x) = \sum_{i=1}^N w^{(i)}p(y^*|x^*, y, x, \theta^{(i)}), \quad (3)$$

where the weights are normalized, i.e., $\sum_i w^{(i)} = 1$.

A common alternative to marginalization is to choose a point estimate of θ using an optimization procedure maximizing the likelihood $p(y|x, \theta)$ (sometimes referred to as empirical Bayes). This may be difficult if the likelihood is multimodal. See the small toy example in Figure 1 illustrating the robustness of marginalization compared to point estimates. There are also situations where point estimates are not sufficient, and marginalization is necessary, such as the change point detection problem in Section III-C.

Our contribution is a method for sampling from the hyperparameter posterior distribution $p(\theta|y, x)$, based on sequential Monte Carlo (SMC) samplers [2]. SMC samplers and their convergence properties are well studied [3].

Several methods have previously been proposed in the literature for marginalization of the GP hyperparameters: Bayesian

Monte Carlo (BMC) [4], slice sampling [5], Hamiltonian Monte Carlo [6, 7], and adaptive importance sampling (AIS) [8]. Particle learning which is closely related to SMC has been proposed by [9] for this purpose. The work by [9], however, is not targeting the hyperparameters directly, and makes (possibly restrictive) assumptions on conjugate priors and model structure.

In this paper, we compare our proposed method to some of these methods, and apply it to two real-data problems: the first demonstrates that marginalization does not have to be more computationally demanding than finding point estimates. The second example, which deals with a fault detection problem from industry, is possible only with an efficient method for marginalization. Our proposed method (and all examples) are available as Matlab code via the first authors homepage.

From the experiments, we conclude that the advantages of the proposed method are (i) robustness towards multimodal hyperparameter posteriors, (ii) simplified tuning (compared to some other alternatives), (iii) competitive computational load, and (iv) online updating of hyperparameters as the data record grows.

II. SAMPLING HYPERPARAMETERS USING SMC

For the numerical marginalization (3), we require N samples, known as *particles*, from the posterior. In this section, we discuss how to use a SMC sampler [2] to generate such a particle system $\{\theta^{(i)}, w^{(i)}\}_{i=1}^N$, where $w^{(i)}$ is the weight of particle $\theta^{(i)}$. The underlying idea is to construct a sequence of probability distributions $(\{\pi_0, \dots, \pi_P\})$, starting from the prior, and ending up in the posterior. The particles are then ‘guided’ through the sequence.

To construct a sequence $\{\pi_0, \dots, \pi_P\}$, we use the fact that $p(\theta|y, x)$ depends on the data (y, x) , by partitioning the data points into P disjoint batches $\{B_n\}_{n=1}^P$ and adding them sequentially as $\pi_n(\theta) \propto p(y_{B_{1:n}} | x_{B_{1:n}}, \theta)p(\theta)$.

To guide the particles through the smooth sequence $\{\pi_0, \dots, \pi_P\}$, we will iteratively apply the three steps weighting, resampling and propagation, akin to a particle filter.

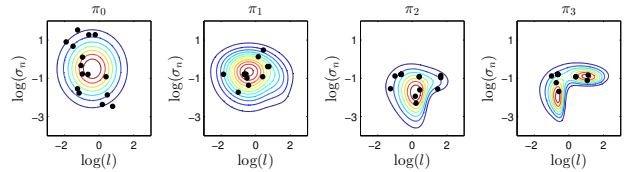
In the *weighting* step, the ‘usefulness’ of each particle is evaluated. To ensure convergence properties, the particles can be evaluated as [2, Section 3.3.2]

$$w_n^{(i)} = \frac{\pi_n(\theta_{n-1}^{(i)})}{\pi_{n-1}(\theta_{n-1}^{(i)})} w_{n-1}^{(i)}. \quad (4)$$

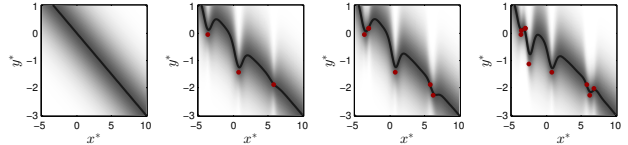
To avoid numerical problems, the particles have to be *resampled*. The idea is to duplicate particle with large weights, and discard particles with small weights.

To *propagate* the particles $\theta_{n-1}^{(i)}$ from π_{n-1} to π_n , a Metropolis-Hastings (MH) kernel $\mathcal{K} : \Theta \mapsto \Theta$ with invariant distribution π_n can be used. The procedure of propagating θ_{n-1} (a sample of π_{n-1}) to θ_n (a sample of π_n) by \mathcal{K} is as follows: (i) Sample a new particle θ' from a proposal $q(\cdot|\theta_{n-1})$, e.g., a random walk with variance h . (ii) Compensate for the discrepancy between π_n and q by setting $\theta_n = \theta'$ with probability

$$\alpha(\theta_n, \theta') = \min \left\{ 1, \frac{\pi_n(\theta')}{\pi_n(\theta_n)} \frac{q(\theta_n|\theta')}{q(\theta'|\theta_n)} \right\}, \quad (5)$$



(a) A transition from the prior $p(\theta)$ to the posterior $p(\theta|y, x)$ for the data in Figure 1b, obtained by adding 3 data points in each step to the likelihood. The particles are obtained from the SMC sampler.



(b) GP regression with marginalized hyperparameters from the corresponding posterior, obtained as a by-product from the particles depicted in (a). From left to right, 0 data points (i.e., the prior), 3 data points, 6 data points, and 9 data points. As we formulated the problem, only the rightmost figure is of interest. This illustrates however how this method can be used in online problem in a natural way.

Fig. 2. Illustration of the SMC sampler, as it evolves from the prior (no data) to the posterior (all data).

Algorithm 1 Hyperparameter posterior sampler

Input: Data (y, x) , GP prior, and prior $p(\theta)$.

Output: N samples $\{\theta^{(i)}\}_{i=1}^N$ from $p(\theta|y, x) \propto p(y|x, \theta)p(\theta)$.

All statements with superscript (i) are for $i = 1, \dots, N$.

- 1: Define $\pi_n(\theta) = p(y_{B_{1:n}} | x_{B_{1:n}}, \theta)p(\theta)$ by partitioning the data into P batches $\{B_n\}_{n=1}^P$.
 - 2: Sample $\theta_0^{(i)}$ from $p(\theta)$ ($= \pi_0(\theta)$).
 - 3: **for** $n = 1$ to P **do**
 - 4: Update weights according to (4).
 - 5: Resample $\{\theta_n^{(i)}, w_n^{(i)}\}_{i=1}^N$ if needed.
 - 6: **for** $k = 1$ to K **do**
 - 7: Propose $\theta'^{(i)}$ from $q(\theta'|\theta_n^{k-1, (i)})$.
 - 8: Set $\theta_n^{k, (i)} = \theta'^{(i)}$ with prob. $\alpha(\theta_n^{k-1, (i)}, \theta'^{(i)})$ (5).
 - 9: **end for**
 - 10: **end for**
-

and otherwise $\theta_n = \theta_{n-1}$. To improve the mixing, this procedure can be repeated K times. For this, we use the notation $\theta_{n-1} = \theta_n^0 \rightarrow \theta_n^1 \rightarrow \dots \rightarrow \theta_n^K = \theta_n$.

We now have an SMC sampler to obtain samples from the hyperparameter posterior, summarized in Algorithm 1 and illustrated by Figure 2. From the figure, the suitability to online applications is clear: If another data point is added to the data, the sequence can be extended to π_4 including the new data point, and only the transition from π_3 to π_4 has to be performed.

We make use of the adaptive SMC sampler by [10] in the numerical examples to adapt the proposal q automatically.

The computational cost of Algorithm 1 is in practice governed by the $2NPK$ evaluations of the likelihood $p(y|x, \theta)$. Hence, it is important to choose the number of samples N , SMC steps P , and MH-moves per SMC-step K sensibly. An idea of sensible numbers will be given along with the examples in the next section.

III. EXAMPLES AND RESULTS

We consider three examples for demonstrating our proposed approach. First, we consider a small simulated example, also comparing to alternative sampling methods, and thereafter two applications with real-world data. The first real-data example is a benchmark problem to compare the marginalization approach in Algorithm 1 to the point estimates obtained using optimization. In the third example, we illustrate how we can make use of our solution within a GP-based online change point detection algorithm. To this end, we require marginalization of the hyperparameters, so an efficient hyperparameter posterior sampler is indeed a key enabler for this. The online nature of the problem also fits well to the possibility to update the samples in Algorithm 1 online, as discussed in Section II.

A. Simulated example

We consider a small problem of 5 data points, and a covariance and mean function with 7 hyperparameters in total. We begin by considering the problem of marginalizing out 7 hyperparameters in a GP prior given 5 data points. Here, we are interested in comparing the performance of our SMC sampler (Algorithm 1) with some popular alternative methods; BMC [4], AIS [8], and (deterministic) gridding.

The results for 15 runs are presented in Figure 3; it is indeed good if the variance between consecutive runs of the same algorithm gives similar results. The variations between the runs decrease faster for Algorithm 1 than for the comparable methods. When the GP prior has few hyperparameters, we conclude that the AIS and gridding might be competitive methods. We have not managed to obtain competitive results with BMC for any problem size, but it should be noted that the computational load of BMC can be substantially decreased if the hyperparameter prior is independent between the dimensions.

The results for the conceptually different point estimates are also presented in Figure 3. The initialization point to the optimization algorithm is drawn from the prior: although it is a deterministic method, it is obviously very sensitive to the initialization.

B. Learning a robot arm model

We consider the problem of learning the inverse dynamics of a seven degrees-of-freedom SARCOS anthropomorphic robot arm [1, 11]. We use the same setup as [1, Section 2.5], i.e., a non-trivial setting involving 23 hyperparameters.

To handle the size of the data set (44 484 training and 4 449 test data points), we make use of a subset of: (i) datapoints and (ii) regressors as discussed by [1, Section 8.3.1]. To use our method, we sample the hyperparameters from the posterior with a subset of m data points. For comparability, we have also reproduced the results using point estimates from [1]. The results are reported in Table I. For Algorithm 1, $N = 15$, $P = 20$ and $K = 5$ was used. The priors to the logarithms of the length-scale and the signal variance are $\mathcal{N}(3, 3)$, and $\mathcal{N}(1, 1)$ for the noise variance.

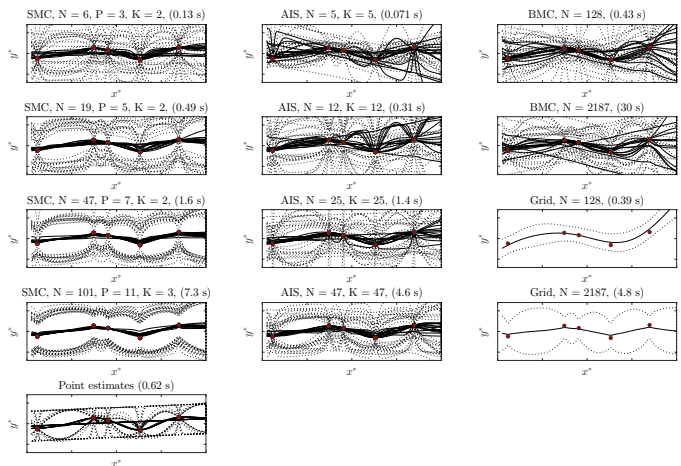


Fig. 3. Comparison between 15 runs of SMC (Algorithm 1), BMC, AIS, and gridding, as well optimized point estimates. The predictions (mean, solid, and 3 standard deviations, dashed) are shown, together with the red data points. The number of particles/samples/grid points is denoted by N , while K and P are algorithm specific tuning parameters. The mean computation time is also shown. All axis are equally scaled.

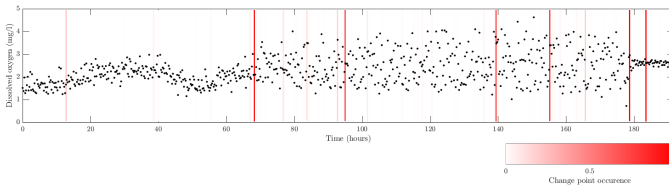
The quite ‘messy’ look in most of the plots indicates that the same method (with fixed settings) behaves differently on each run, which of course is an unwanted effect. However, the SMC sampler is not suffering from this problem for N, P, K large enough. This effect should also be expected for AIS and BMC, but apparently they need more samples/iterations (and thus computing time) than presented here before that effect can be seen.

TABLE I
RESULTS FOR THE SARCOS EXAMPLE IN SECTION III-B.

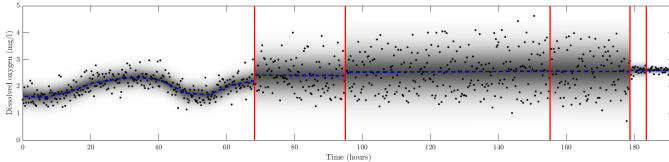
METHOD	m	SMSE ($\times 10^{-2}$)	MSLL	TIME (s)
SUBSET OF DATAPPOINTS				
POINT EST.	256	8.36 \pm 0.80	-1.38 \pm 0.04	6.8
SMC	256	8.10 \pm 1.32	-1.38 \pm 0.56	7.1
POINT EST.	512	6.36 \pm 1.13	-1.51 \pm 0.05	26.4
SMC	512	6.13 \pm 0.91	-1.49 \pm 0.04	22.3
POINT EST.	1024	4.31 \pm 0.16	-1.66 \pm 0.02	101
SMC	1024	4.54 \pm 0.33	-1.61 \pm 0.03	92.5
POINT EST.	2048	2.99 \pm 0.08	-1.78 \pm 0.03	423
SMC	2048	3.33 \pm 0.28	-1.69 \pm 0.06	405
SUBSET OF REGRESSORS				
POINT EST.	256	3.67 \pm 0.17	-1.63 \pm 0.02	6.8
SMC	256	3.55 \pm 0.28	-1.65 \pm 0.05	7.1
POINT EST.	512	2.77 \pm 0.44	-1.79 \pm 0.07	26.4
SMC	512	2.89 \pm 0.20	-1.77 \pm 0.03	22.3
POINT EST.	1024	2.03 \pm 0.11	-1.95 \pm 0.03	101
SMC	1024	2.00*	-1.95*	92.5

Table I presents the results in the same way as [1, Table 8.1]. SMSE is the standardized mean square error (i.e., mean square error normalized by the variance of the target), and MSLL is the mean standardized log loss; 0 if predicting using a Gaussian density with mean and variance of the training data, and negative if ‘better’. The time is referring to the time required to sample and optimize the hyperparameters, respectively (not including the test evaluation). Numerical problems were experienced for large m , therefore * indicates runs where no interval can be reported.

Table I indicates no significant difference between the performance of our method and point estimates. It is however worth also to note the computational load: As Algorithm 1 apparently makes an equally good job in finding relevant hyperparameters as the optimization, it is a confirmation that our proposed method is indeed a competitive alternative to point estimates even for large problems.



(a) Measurements of dissolved oxygen (in mg/l) in a bioreactor with a sampling period of 15 minutes. The indicated change points are marked in red. Especially as the algorithm is fully Bayesian, the outcome is one probability distribution per data sample. This is comprehensively illustrated as the occurrence of change points in ‘backwards simulations’ through these distributions. A more intensive red color is a more likely change point.



(b) Thresholding of Figure (a), with GP regression in each obtained segment. The different characteristics in different segments are possible due to marginalization of the hyperparameters.

Fig. 4. Results for the GP-based change point detection.

C. Fault detection of oxygen sensors

We now consider data from the wastewater treatment plant K ppalaverket, Sweden. An oxygen sensor measures the dissolved oxygen (in mg/l) in a bioreactor, but the sensor gets clogged because of suspended cleaning. The identification of such events is relevant to the control of wastewater treatment plants [12]. We apply the GP-based online change point detection algorithm by [7], where the hyperparameters are marginalized using our proposed method.

The GP-based change point detection presented by [7] can be summarized as follows: If data $y_{1:T}$ undergo a change at time r , it is of interest to (online) detect r , i.e., estimate $p(r|y_{1:t})$. The algorithmic idea is a recursive message passing scheme, updating the probability $p(r_t, y_{1:t})$, where $r_t \in \{1, \dots, t\}$ is the last change point at time t .

To make predictions using a GP model, the hyperparameters either have to be fixed across all data segments, or marginalized. As it is not relevant to use fixed hyperparameters, an efficient sampling algorithm is a key enabler in solving this problem. The consecutive predictions $p(y_t|r_{t-1}, y_{r_t:t-1})$ and $p(y_{t+1}|r_{t-1}, y_{r_t:t})$ are both needed for the algorithm, hence our approach fit this problem well, as discussed in Section II. We used $N = 25$ particles. On average, sampling the hyperparameters, i.e., one run of Algorithm 1, took 0.55 seconds on a standard desktop computer.

The results are presented in Figure 4a. The expected points, suspension and resuming of the cleaning, are indeed indicated. An interpretation of the result is obtained by converting the results to point estimates by thresholding, and plotting at the GP regression for each individual segment, see Figure 4b.

Note the data-driven nature of the algorithm, as no explicit model of the sensor was used at all. The tuning parameters are the covariance and mean functions, the prior of the change points and the hyperparameter priors.

IV. CONCLUSION

We have proposed and demonstrated an SMC-based method to marginalize hyperparameters in GP models. The observed benefits are robustness towards multimodal posteriors (Figure 1) and a competitive computational load (Section III-B), also compared to the commonly used point estimates of the hyperparameters. We have been able to cope with a hyperparameter space of dimension 23 (Section III-B), and also concluded a sound convergence behavior (Section III). Finally, the online update of the hyperparameters has been shown useful within the industry-relevant data-driven fault detection application (Section III-C). As a future direction, it would be interesting to apply our method to the challenging GP optimization problem of system identification [13].

ACKNOWLEDGMENTS

This work was supported by the project *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524) funded by the Swedish Research Council (VR). We would also like to thank Oscar Samuelsson and Dr. Jes s Zambrano for providing the sensor data in Section III-C.

REFERENCES

- [1] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Cambridge, MA: MIT Press, 2006.
- [2] P. Del Moral, A. Doucet, and A. Jasra, “Sequential Monte Carlo samplers,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 3, pp. 411–436, 2006.
- [3] N. Whiteley, “Sequential Monte Carlo samplers: error bounds and insensitivity to initial conditions,” *Stochastic Analysis and Applications*, vol. 30, no. 5, pp. 774–798, 2012.
- [4] 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 *Proceedings of the 7th international conference on information processing in sensor networks*, St. Louis, MO, USA, Apr. 2008, pp. 109–120.
- [5] D. K. Agarwal and A. E. Gelfand, “Slice sampling for simulation based fitting of spatial data models,” *Statistics and Computing*, vol. 15, no. 1, pp. 61–69, 2005.
- [6] R. M. Neal, “MCMC using Hamiltonian dynamics,” in *Handbook of Markov Chain Monte Carlo*, S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, Eds. Chapman & Hall/CRC Press, 2010.
- [7] Y. Saat ı, R. D. Turner, and C. E. Rasmussen, “Gaussian process change point models,” in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, Haifa, Israel, Jun. 2010, pp. 927–934.
- [8] D. Petelin, M. Ga perin, and V. Smidl, “Adaptive importance sampling for Bayesian inference in Gaussian process models,” in *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, Aug. 2014, pp. 5011–5015.
- [9] R. B. Gramacy and N. G. Polson, “Particle learning of Gaussian process models for sequential design and optimization,” *Journal of Computational and Graphical Statistics*, vol. 20, no. 1, pp. 102–118, 2011.
- [10] P. Fearnhead and B. M. Taylor, “An adaptive sequential Monte Carlo sampler,” *Bayesian Analysis*, vol. 8, no. 2, pp. 411–438, 2013.
- [11] S. Vijayakumar and S. Schaal, “Locally weighted projection regression: Incremental real time learning in high dimensional space,” in *Proceedings of the 7th International Conference on Machine Learning (ICML)*, Stanford, CA, USA, Jun. 2000, pp. 1079–1086.
- [12] G. Olsson, B. Carlsson, J. Comas, J. Copp, K. V. Gernaey, P. Ingildsen, U. Jeppsson, C. Kim, L. Rieger, I. Rodriguez-Roda, J.-P. Steyer, I. Tak acs, P. A. Vanrolleghem, A. Vargas Casillas, Z. Yuan, and L.  mand, “Instrumentation, control and automation in wastewater – from London 1973 to Narbonne 2013,” *Water Science and Technology*, vol. 69, no. 7, pp. 1373–1385, 2014.
- [13] J. Dahlin and F. Lindsten, “Particle filter-based Gaussian process optimization for parameter inference,” in *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, Aug. 2014, pp. 8675–8680.