Optimizing Long-term Predictions for Model-based Policy Search

Andreas Doerr[1,2], **Christian Daniel**[1], **Duy Nguyen-Tuong**[1], **Alonso Marco**[2], **Stefan Schaal**[2,3], **Marc Toussaint**[4], **Sebastian Trimpe**[2]

[1]Bosch Center for Artificial Intelligence, Renningen, Germany
{Andreas.Doerr3,Christian.Daniel,Duy.Nguyen-Tuong}@de.bosch.com
[2]Autonomous Motion Dept., Max Planck Institute for Intelligent Systems, Tübingen, Germany
{amarco,strimpe}@tue.mpg.de
[3]Computational Learning and Motor Control Lab, University of Southern California, USA
sschaal@usc.edu
[4]Machine Learning and Robotics Lab, University of Stuttgart, Germany
Marc.Toussaint@ipvs.uni-stuttgart.de

Abstract: We propose a novel long-term optimization criterion to improve the robustness of model-based reinforcement learning in real-world scenarios. Learning a dynamics model to derive a solution promises much greater data-efficiency and reusability compared to model-free alternatives. In practice, however, model-based RL suffers from various imperfections such as noisy input and output data, delays and unmeasured (latent) states. To achieve higher resilience against such effects, we propose to optimize a generative long-term prediction model directly with respect to the likelihood of observed trajectories as opposed to the common approach of optimizing a dynamics model for one-step-ahead predictions. We evaluate the proposed method on several artificial and real-world benchmark problems and compare it to PILCO, a model-based RL framework, in experiments on a manipulation robot. The results show that the proposed method is competitive compared to state-of-the-art model learning methods. In contrast to these more involved models, our model can directly be employed for policy search and outperforms a baseline method in the robot experiment.

Keywords: Model learning, model-based policy search, long-term predictions, Gaussian process dynamics model, learning control, reinforcement learning

1 Introduction

Recently, solutions to several important Reinforcement Learning (RL) problems have been presented such as learning directly from high-dimensional input space [1], generalizing between local solutions [2], safe exploration [3] or data-efficient learning [4]. While model-free approaches have been shown to be powerful learning methods when large amounts of training data are available, this prerequisite often does not hold for real-world problems. Model-based approaches, on the other hand, can be much more data efficient as they can learn solutions for complex tasks from mere seconds of system interaction time.

Despite the appealing theoretical benefits of model-based RL methods, many studies rely upon model-free methods since those make fewer assumptions. Model-based RL methods often assume, for example, i.i.d. noise free measurements of inputs and outputs, Markovian system behavior and a fully observable state space. Violation of these assumptions, however, is both frequent and critical to the learning process in practical applications. In the context of RL, a key requirement for a good model is a good *long-term* prediction of complete rollouts. In contrast, typical model learning approaches are based on one-step-ahead predictions, which tend to diverge when predicting long-term behavior due to the accumulation of small errors [5]. Furthermore, when using a model to infer a

1st Conference on Robot Learning (CoRL 2017), Mountain View, United States.

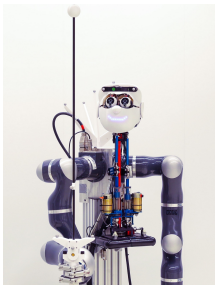


Figure 1: A humanoid upper-body robot learning to balance an inverted pendulum. For this task, improved data-efficiency and therefore faster policy learning is demonstrated based on the presented long-term optimization model-learning procedure.

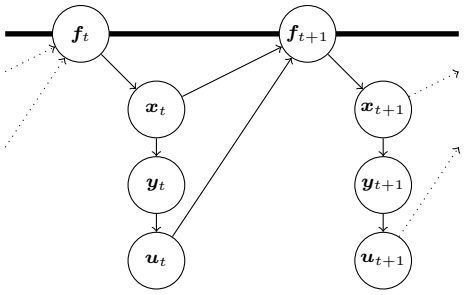


Figure 2: General system description. The solid black line indicates the latent states being jointly Gaussian under a Gaussian Process prior. This model can be extended to a latent autoregressive model by including historic states and inputs into the transition model.

policy, the model does not necessarily need to perform well for arbitrary inputs, but rather for typical inputs in the policy space.

We propose a novel model learning technique for Model-Based Policy Search (MBPS). In particular, the main contribution is the development of a new model optimization objective for learning latent autoregressive GP dynamics models by maximizing the likelihood of complete rollouts under a given policy. The proposed method is tailored, but not limited, to model learning for policy search. The effectiveness of our approach in both scenarios, model learning and policy search, is confirmed through a number of benchmark evaluations, as well as a comparison on a real robotic system.

2 Background and Related Work

In this work, we consider the control of an unknown dynamical system given as a State-Space Model (SSM) with unknown transition function $\boldsymbol{f}$ and observation function $\boldsymbol{g}$ as

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{\epsilon}_t^{(x)}, \quad \boldsymbol{y}_t = \boldsymbol{g}(\boldsymbol{x}_t) + \boldsymbol{\epsilon}_t^{(y)}, \tag{1}$$

with Gaussian process noise $\boldsymbol{\epsilon}_t^{(\mathrm{x})} \sim \mathcal{N}(0, \Sigma_{\mathrm{x}})$ and Gaussian sensor noise $\boldsymbol{\epsilon}_t^{(\mathrm{y})} \sim \mathcal{N}(0, \Sigma_{\mathrm{y}})$. The system has time discrete, continuously valued inputs $\boldsymbol{u}_t \in \mathbb{R}^{D_{\mathrm{u}}}$ and continuously valued observations $\boldsymbol{y}_t \in \mathbb{R}^{D_{\mathrm{y}}}$. Its internal, latent state $\boldsymbol{x}_t \in \mathbb{R}^{D_{\mathrm{x}}}$ can usually not be fully measured. Inputs to the system are either feedforward control signals or generated by a deterministic policy $\boldsymbol{u}_t = \pi(\boldsymbol{y}_t; \theta_\pi)$ parametrized by θ_π.

Since we usually do not have access to an analytical model of the real system, we need to learn the model from input/output data. From the i-th rollout of length T_i, we obtain a trajectory of input/output data $\tau_i = ((\boldsymbol{u}_0, \boldsymbol{y}_0), \ldots, (\boldsymbol{u}_{T_i}, \boldsymbol{y}_{T_i}))$ and possibly the corresponding feedback policy parameters θ_{π_i}. However, learning good models from experimental data is difficult for several reasons:

(i) Noisy outputs: Due to stochasticity in the process and in the sensors, measured data is corrupted by noise. Bayesian nonparametric Gaussian Process (GP) models [6] are a popular choice for dynamics models as they allow one to incorporate uncertainty about model and predictions [7]. Assuming a fully measurable system state and neglecting the observation noise, the one-step system dynamics could be modeled by a GP $\hat{\boldsymbol{f}}$ as

$$\boldsymbol{x}_{t+1} = \hat{\boldsymbol{f}}(\boldsymbol{x}_t, \boldsymbol{u}_t) + \epsilon_t. \tag{2}$$

Based on GP dynamics models, several methods have been proposed to propagate model uncertainty over multiple prediction steps using assumed density filtering (moment matching) [8, 9] or variational approximations [10]. However, the GP models are usually intrinsically optimized for one-step-ahead predictions. In contrast, we introduce a long-term prediction objective, but adopt the moment-matching approximation to propagate the uncertainty through rollouts. This approximation allows one to calculate predictive distributions and their gradients analytically, which is essential for fast policy learning.

(ii) Latent states: In most real-world scenarios, only a part of the relevant system state can be directly observed. To still capture the relevant information, which is required to predict the future system behavior, two classes of methods have been established:

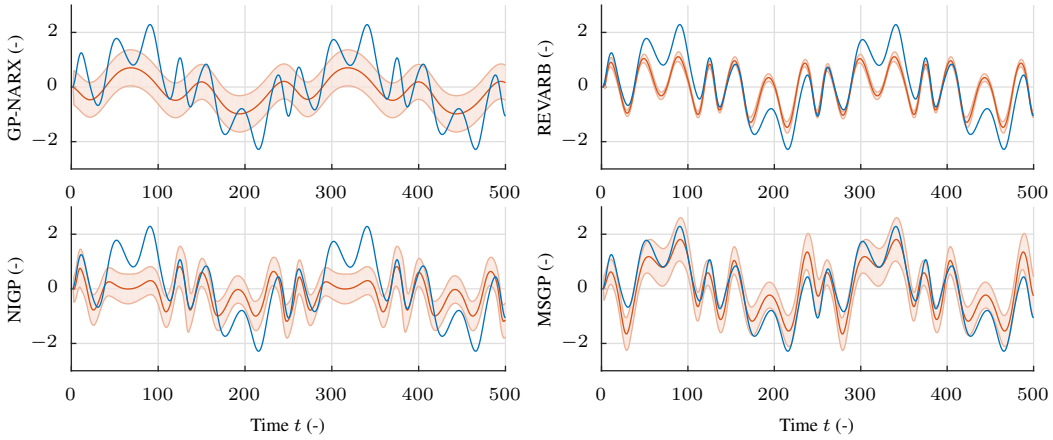


Figure 3: Comparison of the free simulation on the synthetic system 4 test dataset. From top to bottom in red (with +/- 2 standard deviation): GP-NARX, NIGP, REVARB and the proposed MSGP method. The true, latent state of the system is shown in blue.

The first class utilizes historical input and output measurements up to a horizon l_y and l_u respectively, to predict the next observation. Including previous measurements can resolve the issue of having only partial state measurements in many real systems. This leads to Nonlinear AutoRegressive models with eXogenous inputs (NARX),

$$\boldsymbol{y}_{t+1} = \hat{\boldsymbol{f}}(\boldsymbol{y}_t, \ldots, \boldsymbol{y}_{t-l_\mathrm{y}}, \boldsymbol{u}_t, \ldots, \boldsymbol{u}_{t-l_\mathrm{u}})\,. \tag{3}$$

NARX models based on GPs (GP-NARX) have been proposed for example by Murray-Smith et al. [11] and Kocijan et al. [12].

SSMs form the second class of methods, where an explicit representation of a Markovian system state is inferred. For a Bayesian treatment, GP-SSMs have been proposed by Frigola et al. [13], where both the transition model and the observation model can have GP priors. In Frigola and Rasmussen [14], data preprocessing and Bayesian modeling is combined for SSM. Inference in latent space proves to be a challenging task which is the major drawback of the existing SSM methods. Usually, an expectation-maximization (EM)-like method is employed to alternate between latent-state inference (E-Step) and model learning (M-step). While the E-step is usually based on smoothing methods, either using sampling [15], variational approximations [13] or moment matching approximations [16], the M-step usually consists in optimizing a variational lower bound on the trajectory likelihood. Learning the latent state representation proves to be a hard problem as it leads to a number of open parameters, which increases linearly in the dataset size. In recent work [17, 18], the fully parametrized latent state has been replaced by a recurrent recognition model, which is essentially a deep network that infers the latent state from future and past observations and therefore restricts the unlimited number of open latent space parameters to a limited number of deep network parameters.

By employing an autoregressive model, our method bypasses the challenging inference in latent space. At the same time, our method allows for analytically tractable optimization of the long-term trajectory likelihood similar to the inference in the SSM. In our experiments, we compare the performance of the proposed method to GP-NARX (and its PILCO variant).

(iii) Noisy inputs: The standard GP framework assumes noise free input measurements and outputs corrupted by constant-variance Gaussian noise. In time-series modeling, where each observation is corrupted by noise, the noise free input assumption as well as the i.i.d. assumption are violated. Neglecting the input noise results in model bias, especially in situations of low signal to noise ratios. An example for model-bias is shown in the top plot of Fig. 3, which illustrates the shortcomings of optimizing a model on noisy measurements using one-step ahead predictions (cf. GP-NARX results). When performing full rollouts of a trajectory on such a learned one-step dynamics, the model-bias leads to implausible predictions. In contrast, methods that explicitly deal with noisy inputs (NIGP) and optimize long-term predictions (REVARB, MSGP) can improve prediction performance significantly.

One line of research addressing noisy input measurements is treating them as deterministic inputs and inflates the corresponding output variance, which leads to state-dependent, heteroscedastic

GPs [19, 20]. In non-linear time series modeling, GP input and output share the same noise distribution, which can be exploited by more tailored methods as for example Noise Input Gaussian Processes (NIGP) [21].

In the deep learning community, powerful recurrent models have been published for sequential data [22]. However, these models are inherently deterministic. In order to utilize their long-term predictions for policy search or trajectory optimization, additional precautions are necessary to avoid regions of insufficient or inconclusive system knowledge. Additionally, training these models requires large amounts of data, which makes them less suitable for learning on physical systems with no prior knowledge. Recent work tries to alleviate these issues by introducing deep recurrent GPs [17] to leverage the data-efficiency of GP regression and to obtain uncertainty estimates. Similarly, Mishra et al. [23] utilize variational auto-encoders to obtain predictive distributions for long-term system behavior. Whilst being conceptually interesting, extensions of these methods for data-efficient learning from scratch are yet to be proposed.

3 Multi-Step Gaussian Processes for RL

In order to address the issues identified in Sec. 2, we propose an improved model learning technique aiming at increasing the performance of MBPS methods.

3.1 Main Idea

Our contribution is based on three core insights:

(i) Models employed for finite horizon policy optimization should capture the closed-loop, long-term behavior of the system given a feedback policy. When optimizing models for one-step-ahead predictions, the model learning process is oftentimes derailed by effects such as system noise, causing small errors to accumulate. If, however, the model is trained to predict full trajectories and accumulated errors are backpropagated into the predicted system states during model optimization, the resulting model becomes better at capturing the relevant long-term system dynamics.

(ii) When learning models with the purpose of policy search, we frequently have access to the actual policy that generated the data. Thus, by replacing the inputs to the model $\boldsymbol{u}_t$ by the inputs generated by the specific policy $\boldsymbol{u}_t = \pi(\boldsymbol{y}_t, \theta_\pi)$, we are able to optimize the model for predicting long-term system development based on how the policy would act for the predicted system behavior. In contrast to general system identification methods, which usually aim at learning a model over the entire space of arbitrary control input sequences, our method can learn a model tailored to the smaller control input manifold spanned by the class of considered feedback policies.

(iii) Model learning for long-term predictions and finite horizon policy search are strongly related problems. If approximations are necessary in the model evaluation for the subsequent policy search step, they should already be taken into account in the model learning step.

In the following subsections, we elaborate on how we implement these ideas to enable efficient model-based learning of policies on real systems with latent states and noisy observations from scratch in a Bayesian model learning framework, which we refer to as Multi-Step Gaussian Processes (MSGP).

3.2 Trajectory Likelihood Optimization

MSGP is built around a generative model for the distribution of observations $\boldsymbol{y}_{0:T}$ conditioned on the system's initial state $\boldsymbol{x}_0$ and *either* the sequence of actions $\boldsymbol{u}_{0:T}$ *or* the applied control policy, given by θ_π. Therefore, MSGP can learn from data observed in open-loop as well as closed-loop experiments. Ultimately, for a good long-term prediction model, all trajectories observed on the real system should be likely under these predictive distributions. For a set D_{ff} of trajectories based on a feedforward (ff) control signals $\boldsymbol{u}_{0:T_i}$, and a different set D_{fb} of trajectories originating from feedback (fb) policies $\theta_{\pi,j}$, the model loss is composed from

$$L_{\text{ff}}(\theta) = -\sum_{\tau_i \in D_{\text{ff}}} \log p(\boldsymbol{y}_{0:T_i} | \boldsymbol{u}_{0:T_i}, x_{0,i}) \quad \text{and} \quad L_{\text{fb}}(\theta) = -\sum_{\tau_j \in D_{\text{ff}}} \log p(\boldsymbol{y}_{0:T_j} | \theta_{\pi,j}, x_{0,j}) \,. \quad (4)$$

The generative MSGP model is parametrized by θ (made precise below). MSGP approximates (4) such that temporal correlations between the current timestep and all previous timesteps are maintained. Model errors are therefore accumulated over time and backpropagated during model optimization to obtain good long-term predictions.

The MSGP model is a specific version of the SSM in (1). The observation function g is given, without loss of generality, by the identity function (cf. [15]). The latent state transition model f is an autoregressive GP model. Thereby, we sidestep the challenging inference in a truly latent state space with unknown dimensionality and represent the missing information by taking into account historic state information. As in (1), the model includes Gaussian process and observation noise. A graphical model of the generative model is shown in Fig. 2.

In contrast to standard GP dynamics models, which are trained on one-step-ahead predictions given measured training data, the latent system state is not directly measurable. Following work by Turner et al. [16], the latent state GP is therefore parametrized by m inducing inputs $\bar{\boldsymbol{X}} = [\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_m]^T$ and targets $\bar{\boldsymbol{y}} = [\bar{\boldsymbol{y}}_1, \ldots, \bar{\boldsymbol{y}}_m]^T$. Inducing points are artificial GP inputs and targets which can be optimized to capture the system behavior. This approach is commonly employed for learning sparse GP representations [24, 25]. The GP inputs are of the autoregressive form in (3). The GP hyperparameters θ_{hyp}, inducing inputs $\bar{\boldsymbol{X}}$ and targets $\bar{\boldsymbol{y}}$, as well as process and observation noise variances are parameters of the generative model, which are jointly optimized. The parameter vector is therefore given by $\theta = (\theta_{\text{hyp}}, \bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_m, \bar{\boldsymbol{y}}_1, \ldots, \bar{\boldsymbol{y}}_m, \Sigma_{\text{x}}, \Sigma_{\text{y}})$.

Standard GP models are usually trained by maximizing the data log likelihood [6] given by

$$\log p(\boldsymbol{y}|\boldsymbol{X}) = -\frac{1}{2}\boldsymbol{y}^T(K + \sigma_{\text{n}}^2 I)^{-1}\boldsymbol{y} - \frac{1}{2}\log|K + \sigma_{\text{n}}^2 I| - \frac{n}{2}\log 2\pi\,, \tag{5}$$

where $\boldsymbol{y}$ and $\boldsymbol{X}$ are training data points obtained from measured system data with input dimensionality n. These standard models only account for Gaussian output noise given by σ_n. The GP covariance matrix K is obtained as $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j), \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in \bar{\boldsymbol{X}}$ for a given kernel function k. This objective automatically trades data-fit (first term) for model complexity (second term). In contrast, the long-term trajectory likelihood objectives in (4)) are not automatically penalizing the model complexity of the latent dynamics model. To avoid overfitting of the latent state model, we add the model complexity penalty given by

$$L_{\text{comp}}(\theta) = -\frac{1}{2}\log|K + \sigma_{\text{x}}^2 I|\,, \tag{6}$$

to the long-term likelihood term in (4). The full MSGP optimization objective is therefore given by

$$\theta^* = \arg\min_{\theta} L_{\text{ff}}(\theta) + L_{\text{fb}}(\theta) + L_{\text{comp}}(\theta) \tag{7}$$

3.3 Marginal Observation Distribution

In the following, we summarize the necessary computations to obtain the conditional observation distributions from (4). The approximations are inspired by the long-term prediction method utilized in the policy search framework PILCO [4]. Due to the Markovian state $\boldsymbol{x}_t$, the joint distribution of $p(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{y}, \boldsymbol{u})$ factorizes such that the marginal observation distribution is given by

$$p(\boldsymbol{y}) = \int p(\boldsymbol{x}_0)\prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{f}_t)p(\boldsymbol{f}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})p(\boldsymbol{u}_{t-1}|\boldsymbol{y}_{t-1}; \theta_\pi)\mathrm{d}\boldsymbol{u}\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{f}\,. \tag{8}$$

We assume a Gaussian distribution for the initial state $p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0, \Sigma_0)$.

Observation Model Based on the Gaussian approximation of the latent state $p(\boldsymbol{x}_t)$ in every prediction step the conditional observation distribution is obtained as

$$p(\boldsymbol{y}_t|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{x}_t, \Sigma_{\text{y}})\,. \tag{9}$$

Policy The policy $\boldsymbol{u}_t = \pi(\boldsymbol{y}_t; \theta_\pi)$ is evaluated on the current observation $\boldsymbol{y}_t$. To obtain the input to the dynamics model, we compute the joint distribution of the current state and the policy output $p(\boldsymbol{x}_t, \boldsymbol{u}_t)$ as given by

$$p(\boldsymbol{x}_t, \boldsymbol{u}_t) = \int p(\boldsymbol{u}_t|\boldsymbol{y}_t, \theta_\pi)p(\boldsymbol{y}_t|\boldsymbol{x}_t)p(\boldsymbol{x}_t)d\boldsymbol{y}_t\,. \tag{10}$$

This integral can be computed exactly for Gaussian observation models (cf. (9)) and linear policies. For non-linear policies (e.g. RBF networks or neural networks), we revert to a moment matching approximation of the joint probability distribution $p(\boldsymbol{x}_t, \boldsymbol{u}_t)$.

Dynamics Model Each latent state dimension is modeled independently by a single output GP. We employ the popular Squared Exponential (SE) kernel [6] given by

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_\mathrm{f}^2 \exp(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \Lambda (\boldsymbol{x}_i - \boldsymbol{x}_j)) \,, \tag{11}$$

with lengthscales $\Lambda = \mathrm{diag}(1/l_1^2, \dots 1/l_D^2)$ and signal variance σ_f^2. The predictive distribution for the next latent state f_{t+1} is then given by

$$p(f_{t+1}|\bar{\boldsymbol{x}}_t, \bar{\boldsymbol{X}}, \bar{\boldsymbol{y}}) \sim \mathcal{N}(\mu_{f_{t+1}}, \sigma_{f_{t+1}}^2) \,, \tag{12}$$

where the mean and the variance are given by

$$\mu_{f_{t+1}} = k(\bar{\boldsymbol{x}}_t)^T K^{-1} \bar{\boldsymbol{y}} \,, \quad \sigma_{f_{t+1}}^2 = k(\bar{\boldsymbol{x}}_t, \bar{\boldsymbol{x}}_t) - k(\bar{\boldsymbol{x}}_t)^T K^{-1} k(\bar{\boldsymbol{x}}_t) \,, \tag{13}$$

with $k(\bar{\boldsymbol{x}}_t) = [k(\bar{\boldsymbol{x}}_t, \bar{\boldsymbol{x}}_1), \dots, k(\bar{\boldsymbol{x}}_t, \bar{\boldsymbol{x}}_m)]^T$.

Propagation of Uncertainty The input to the dynamics model is jointly Gaussian. In general, the marginal distribution for propagating a Gaussian distributed input $p(\bar{\boldsymbol{x}}_t)$ through an arbitrary non-linear function $f(x)$, as given by

$$p(f_{t+1}) = \int p(f_{t+1}|\bar{\boldsymbol{x}}_t) p(\bar{\boldsymbol{x}}_t) d\bar{\boldsymbol{x}}_t \,, \tag{14}$$

is non-Gaussian. In order to obtain analytic gradients for the propagation step, we utilize Moment Matching (MM). The approximated predictive distribution for a Gaussian distributed input $\bar{\boldsymbol{x}}_t \sim \mathcal{N}(\boldsymbol{\mu}_{\bar{x}}, \boldsymbol{\Sigma}_{\bar{x}})$ is given by

$$p(f_{t+1}) = \mathcal{N}(\mu_{f_{t+1,\mathrm{MM}}}, \sigma_{f_{t+1,\mathrm{MM}}}) \,, \tag{15}$$

The expressions for mean and variance, and references to detailed derivations of the moment matching formalism can be found in the supplementary material. By applying the steps from (9) to (15) repeatedly over the prediction horizon, we obtain the full predictive distribution (4). Every prediction step is therefore dependent on all previous timesteps such that model errors are accumulated and consequently backpropagated during model optimization to improve the quality of long-term predictions. Due to the choice of a GP latent space dynamics model and the moment matching approximation for the predictive trajectory distribution, gradients for the loss with respect to the model parameters θ can be obtained analytically. We can then optimize the model parameters using standard gradient descent methods.

3.4 Relation between Model Learning and Policy Search

Both, MBPS and model learning can be expressed as

$$\theta_* = \arg\min_\theta J(p(\boldsymbol{y}_{0:T} \mid \boldsymbol{x}_0, \theta)) \,. \tag{16}$$

In MSGP, the loss function J used for model learning is the negative log-likelihood of the observed trajectories and the model's open parameters θ are given by the GP parameters. In contrast, for policy search, the loss J could be a quadratic penalty for deviations from a desired trajectory and the model's parameters θ are given by the policy parameters. In both problems, an optimization based on the predictive distribution $p(\boldsymbol{y}_{0:T})$ as given by (8) has to be carried out. Since (8) is generally not analytically tractable, approximations are made in model learning and policy search. Ideally, the approximations made in the subsequent policy search step should be incorporated in the model learning part already. MSGP is therefore tailored for the Probabilistic Inference for Learning COntrol (PILCO) method [4], since MSGP employs the same approximations to the predictive distribution (4) as utilized in PILCO. Since we changed the GP training procedure but not the GP-model structure itself, we end up with a standard GP, which can be directly employed within the PILCO framework.

4 Experimental Evaluation

The experimental evaluation of MSGP is twofold: (i) Evaluation of the model learning capabilities in comparison to existing system identification methods. (ii) Comparison of the MSGP policy search performance to a state-of-the-art method.

Table 1: Comparison of model learning methods on synthetic (first 5 rows) and real-world (last 5 rows) benchmark examples. The RMSE result is given for the free simulation on the noise free test dataset.

Task	GP-NARX	GP-NARX PILCO	NIGP	REVARB 1	REVARB 2	MSGP
Synthetic system 1	0.2265	0.3511	0.2453	0.2129	**0.1717**	0.3805
Synthetic system 2	0.3535	0.3467	0.4100	**0.3395**	0.3726	0.3424
Synthetic system 3	0.1572	0.1487	0.1950	0.3657	0.1695	**0.1341**
Synthetic system 4	0.5711	0.6016	0.7814	0.6978	0.4554	**0.3287**
Synthetic system 5	0.0164	0.0150	0.0101	0.1193	**0.0035**	0.0143
Actuator	0.6376	1.3767	0.6483	**0.4328**	0.5522	0.7124
Balancing	0.0773	0.0881	0.0776	0.1360	0.0732	**0.0599**
Drives	0.6888	0.7024	0.7525	0.7463	0.5620	**0.4217**
Furnace	1.1996	1.2012	**1.1949**	1.3434	1.9569	1.2013
Dryer	0.2856	0.2956	0.2802	0.8764	**0.1286**	0.1523

4.1 Model Learning Benchmark

For evaluating the system identification performance, the proposed MSGP method is compared to state-of-the-art model learning methods from the literature: (i) GP-NARX [12], (ii) PILCO's GP-NARX [4], (iii) NIGP [21], and (iv) REVARB [17], on five synthetic and five real-world datasets. To compare the quality of the learned system dynamics models, we compute the free simulation for each model. Details about the benchmark datasets and the setup of the individual benchmark methods can be found in the supplementary material.

Synthetic Benchmark Datasets The resulting RMSEs are stated for each method and dataset in Tab. 1 The results show that learning methods based on long-term predictions (REVARB, MSGP) tend to outperform the one-step-ahead models (GP-NARX, GP-NARX-PILCO, NIGP). Overall, the results show that the performance of MSGP is comparable to the one of REVARB methods, however, none of the methods outperforms the others in all cases. A visualization of the free running prediction results on the synthetic system 4 dataset is shown Fig. 3. The model bias introduced by the input noise is clearly visible for the standard GP-NARX model (top plot); the true dynamics is not captured by this model. In contrast, NIGP, REVARB and MSGP capture the dynamics much better. However, REVARB is clearly underestimating the uncertainty about the underlying system and both NIGP and REVARB cannot properly fit the high and low peaks, i.e. the learned lengthscales are too short to generalize well on the brink of the state space covered by training data.

Real-World Benchmark Datasets The results in Tab. 1 show that (like for the synthetic benchmarks) the methods based on long-term predictions (MSGP and REVARB) generally outperform the methods based on one-step-ahead predictions. While there is still no clear overall winner, MSGP seems to perform slightly better for the real-world data sets than for the synthetic ones.

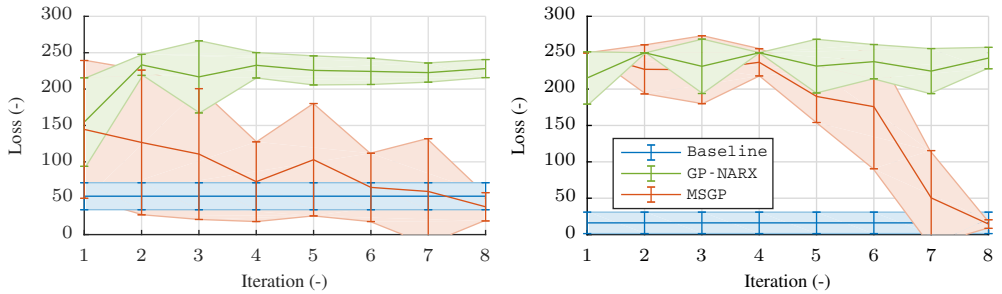
4.2 Policy Search

To demonstrate the policy learning properties of the proposed MSGP method combined with PILCO, we investigate the problem of learning to balance an inverted pendulum, which is a well-known benchmark scenario in control and reinforcement learning [26, 27].

4.2.1 Experimental Setup

We employ a humanoid upper-body robot platform as shown in Fig. 1. The system outputs considered for balancing are the pendulum and end-effector positions, and the input is given by the end-effector acceleration. The policy is composed from a PID feedback controller on the end-effector and a PD controller on the pendulum angle, similar to the experimental setup in Doerr et al. [28].

Despite the simplicity of the task of balancing an inverted pendulum, the robotic arm (cf. Fig. 1) introduces all real-world challenges like noise, delays, and latent states as discussed in Sec. 2. Given the MSGP framework, we can learn in this scenario from scratch without the need for manual tuning and injection of prior knowledge.



(a) Predicted rollout loss. (b) Actual rollout loss.

Figure 4: Iterative learning to balance the inverted pendulum without prior knowledge based on the GP-NARX model as used in standard PILCO (green) and the proposed MSGP model (red). As baseline, the performance of an optimal policy (blue) based on a fixed GP-NARX model is shown. In contrast to the iterative learning procedures, this baseline model was trained on a much larger dataset that incorporates random, instable and stable rollouts. MSGP improves the long-term model predictions such that policy search with no prior domain knowledge is feasible. The graphs represent the outcome of five independent learning runs, depicted as the average (solid line) with +/- 1 standard deviations as shaded areas.

4.2.2 Experimental Results

We compare policy search results based on the proposed MSGP model to standard PILCO using a GP-NARX model. Both methods are initiated with data from four rollouts based on smooth random signals. The distribution of policy costs over learning iterations (mean +/- one standard deviation) as obtained from five independent learning experiments is visualized in Fig. 4. The left plot shows the predicted cost, whereas the right plot shows the actual cost experienced for the rollout of the policy on the real system.

On our robotic setup, due to the challenges discussed in Sec. 2, neither a standard GP nor a GP-NARX dynamics model are sufficient to learn a good predictive model or a stabilizing policy (cf. GP-NARX results in Fig. 4). In contrast, the MSGP model optimization is able to learn a successful model and policy from scratch without requiring prior system knowledge. Stable balancing is achieved after six to eight policy iterations, which corresponds to roughly 30 seconds of interaction with the real system.

To demonstrate the general ability of the GP-NARX model to represent the system's latent dynamics, we report in Fig. 4 (blue) also the baseline policy obtained offline from a much larger, separately recorded dataset. For this, policy search was performed on a GP-NARX model that was trained on a larger dataset including multiple random, unstable, and stable rollouts. The baseline model is clearly sufficient to find a stabilizing policy. This indicates that progress in the iterative PILCO procedure is not limited by the expressivity of the GP-NARX model, but by the model optimization procedure itself. MSGP-based PILCO is able to robustly learn a stabilizing policy iteratively and with a much smaller dataset than the baseline.

5 Conclusion & Future Work

In this work, we introduced MSGP, a lightweight model learning technique to improve long-term predictions, which is applicable to model-based policy search frameworks in a straightforward way. The methodology can be interpreted as optimizing a recurrent, latent GP-NARX dynamics model by maximizing the likelihood of observed system trajectories. In contrast to classic system identification methods, we explicitly optimize the model with respect to its applicability in policy search. Therefore, long-term predictions take into account the dependency on the employed policy and the errors introduced by the approximations made in the policy search step.

Benchmark results are given for artificial and real-world system identification tasks, comparing MSGP to several state-of-the-art non-linear system identification methods, including recurrent (deep) GPs. While MSGP is competitive to state-of-the-art methods for system identification, it clearly outperforms existing approaches on the robot policy search task. Using MSGP, iterative learning of a challenging robotic task from scratch is possible, in contrast to the default PILCO implementation.

Acknowledgments

This research was supported in part by the Max Planck Society, National Science Foundation grants IIS-1205249, IIS-1017134, EECS-0926052, the Office of Naval Research, and the Okawa Foundation.

References

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[2] S. Levine and V. Koltun. Guided policy search. In *International Conference on Machine Learning (ICML)*, 2013.

[3] F. Berkenkamp, A. P. Schoellig, and A. Krause. Safe controller optimization for quadrotors with Gaussian processes. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496, 2016.

[4] M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, pages 465–472, 2011.

[5] S. A. Billings. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[6] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*. MIT Press, 2005.

[7] V. Peterka. Bayesian system identification. *Automatica*, 17(1):41–53, 1981.

[8] J. Quinonero-Candela, A. Girard, and C. E. Rasmussen. Prediction at an uncertain input for Gaussian processes and relevance vector machines-application to multiple-step ahead time-series forecasting. Technical report, Danish Technical University, 2002.

[9] A. Girard, C. E. Rasmussen, J. Quinonero-Candela, R. Murray-Smith, O. Winther, and J. Larsen. Multiple-step ahead prediction for non linear dynamic systemsa Gaussian process treatment with propagation of the uncertainty. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 529–536, 2003.

[10] A. C. Damianou and N. D. Lawrence. Deep Gaussian processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 207–215, 2013.

[11] R. Murray-Smith, T. A. Johansen, and R. Shorten. On transient dynamics, off-equilibrium behaviour and identification in blended multiple model structures. In *European Control Conference (ECC)*, pages 3569–3574. IEEE, 1999.

[12] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith. Dynamic systems identification with Gaussian processes. *Mathematical and Computer Modelling of Dynamical Systems*, 11(4): 411–424, 2005.

[13] R. Frigola, Y. Chen, and C. E. Rasmussen. Variational Gaussian process state-space models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3680–3688, 2014.

[14] R. Frigola and C. E. Rasmussen. Integrated pre-processing for bayesian nonlinear system identification with Gaussian processes. In *52nd IEEE Conference on Decision and Control (CDC)*, pages 5371–5376, 2013.

[15] 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 *Advances in Neural Information Processing Systems (NIPS)*, pages 3156–3164, 2013.

[16] R. Turner, M. Deisenroth, and C. Rasmussen. State-space inference and learning with Gaussian processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 868–875, 2010.

[17] C. L. C. Mattos, Z. Dai, A. Damianou, J. Forth, G. A. Barreto, and N. D. Lawrence. Recurrent Gaussian processes. *arXiv preprint arXiv:1511.06644*, 2015.

[18] S. Eleftheriadis, T. F. Nicholson, M. P. Deisenroth, and J. Hensman. Identification of Gaussian process state space models. *arXiv preprint arXiv:1705.10888*, 2017.

[19] P. W. Goldberg, C. K. Williams, and C. M. Bishop. Regression with input-dependent noise: A Gaussian process treatment. In *Advances in Neural Information Processing Systems (NIPS)*, volume 10, pages 493–499, 1997.

[20] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic Gaussian process regression. In *Proceedings of the 24th International Conference on Machine learning (ICML)*, pages 393–400. ACM, 2007.

[21] A. McHutchon and C. E. Rasmussen. Gaussian process training with input noise. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1341–1349, 2011.

[22] S. Hochreiter and J. Schmidhuber. LSTM can solve hard long time lag problems. In *Advances in Neural Information Processing Systems (NIPS)*, pages 473–479, 1997.

[23] N. Mishra, P. Abbeel, and I. Mordatch. Prediction and control with temporal segment models. *arXiv preprint arXiv:1703.04070*, 2017.

[24] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS)*, volume 18, page 1257, 2006.

[25] J. Quinonero-Candela, C. E. Rasmussen, and C. K. Williams. Approximation methods for Gaussian process regression. *Large-scale kernel machines*, pages 203–224, 2007.

[26] C. W. Anderson. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, 9(3):31–37, 1989.

[27] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe. Automatic LQR tuning based on Gaussian process global optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[28] A. Doerr, D. Nguyen-Tuong, A. Marco, S. Schaal, and S. Trimpe. Model-based policy search for automatic tuning of multivariate PID controllers. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[29] M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 5, pages 567–574, 2009.

[30] C. L. C. Mattos, Z. Dai, A. Damianou, J. Forth, G. A. Barreto, and N. D. Lawrence. REVARB implementation for RGP. https://github.com/zhenwendai/RGP, 2015. [Online; accessed 30-Jan-2017].

[31] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks*, 1(1):4–27, 1990.

[32] C. L. C. Mattos, A. Damianou, G. A. Barreto, and N. D. Lawrence. Latent autoregressive Gaussian processes models for robust system identification. *IFAC-PapersOnLine*, 49(7):1121–1126, 2016.

[33] M. Nrgaard. Hydraulic actuator dataset. http://www.iau.dtu.dk/nnbook/systems.html, 2000. [Online; accessed 30-Jan-2017].

[34] D. Moor. DaISy: Database for the identification of systems. http://homes.esat.kuleuven.be/~smc/daisy/, 2017. [Online; accessed 30-Jan-2017].

[35] T. Wigren. Input-output data sets for development and benchmarking in nonlinear identification. Technical report, Department of Information Technology, Uppsala University, 2010.

A Supplementary Material

In this supplementary material, technical details about the models and methods are given.

A.1 Model Learning Benchmark

The proposed model learning method MSGP is compared to several state-of-the-art methods. Details about the model configuration and the employed benchmark datasets can be found in the following sections.

A.1.1 Benchmark Methods

Each method utilizes 100 inducing inputs and moment matching is applied for long-term predictions.

(i) GP-NARX [12]: The system dynamics is modeled as a function of the history of observations and inputs. A squared exponential kernel with automatic relevance determination is employed and hyperparameters are optimized based on the maximum likelihood objective.

(ii) PILCO's GP-NARX [4]: PILCO adds penalty terms for large signal to noise ratios $\sigma_\mathrm{f}^2/\sigma_\mathrm{n}^2$ and extreme length-scales l_i^2 to the log marginal data likelihood to avoid numerical instabilities when optimizing the GP hyperparameters. Especially on real datasets, the standard GP hyperparameter optimization tends to underestimate the noise, which is prevented by this heuristic.

(iii) NIGP [21]: This method explicitly accounts for uncertainty in the input by treating input points as deterministic and inflating the corresponding output uncertainty. This leads to state-dependent noise proportional to the local gradient of the GP posterior mean. For time-series data, both input and output noise are the same, which can be exploited by this framework.

(iv) REVARB [17] Recurrent Variational Bayes (REVARB) is a recent proposition to optimize the lower bound to the log-marginal likelihood $\log p(\boldsymbol{y})$ using variational techniques. This framework is based on the variational sparse GP framework [29], but allows for computation of (time-)recurrent GP structures and deep GP structures (stacking multiple GP-layers in each time-step). A Python-implementation is available online [30]. For our benchmark, we run REVARB using one (REVARB1) respectively two (REVARB2) hidden layers, where each layer is provided with 100 inducing inputs. We closely follow the original setup as presented by [17], performing 50 initial optimization steps based on fixed variances and up to 10000 steps based on variable variances. Unlike for the other benchmark methods, the autoregressive history of REVARB implicitly becomes longer when introducing additional hidden layers.

(v) MSGP: The proposed MSGP model learning method is implemented in Python using *autograd* to automatically derive the loss gradients. The latent GP-NARX model is initialized using a sparse GP model trained on the observations $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$. We optimized the GP hyperparameters and targets by gradient descent using the Adam optimizer.

A.1.2 Benchmark Datasets

The benchmarks datasets are composed of popular system identification datasets from related work [31, 12, 32]. For all of these problems, both the inputs and outputs are one-dimensional $D_u = D_y = 1$. However, the system's true state is higher dimensional such that an autoregressive history or an explicit latent state representation is required to capture the relevant dynamics. The number of historic inputs and outputs for the autoregressive methods is fixed a-priori for each dataset.

Synthetic Datasets The synthetic benchmarks are taken from existing literature and are described by difference equations. The first four synthetic systems have been presented in [31], whereas the fifth originates from [12]. They have been used as benchmarks to assess robust system identification in the presence of outliers in [32]. The systems' training and test data as well as observation noise are summarized in Tab. 2.

Real-World Datasets The real-world benchmark datasets are composed from typical system identification problems from technical systems like hydraulic actuators, furnaces, hair dryers or electrical

Table 2: Artificial datasets used in the benchmark experiments (cf. Mattos et al. [32]).

Task	ODE	Training	Test	Noise
Synthetic system 1	$y_t = \frac{y_{t-1}y_{t-2}(y_{t-1}+2.5)}{1+y_{t-1}^2+y_{t-2}^2} + u_{t-1}$	$u_t = \mathrm{U}(-2,2)$ 300 samples	$u_t = \sin(2\pi t/25)$ 100 samples	$\mathcal{N}(0, 0.29)$
Synthetic system 2	$y_t = \frac{y_{t-1}}{1+y_{t-1}^2} + u_{t-1}^3$	$u_t = \mathrm{U}(-2,2)$ 300 samples	$u_t = \sin(2\pi t/25) + \sin(2\pi t/10)$ 100 samples	$\mathcal{N}(0, 0.65)$
Synthetic system 3	$y_t = 0.8y_{t-1} + (u_{t-1} - 0.8)u_{t-1}(u_{t-1} + 0.5)$	$u_t = \mathrm{U}(-1,1)$ 300 samples	$u_t = \sin(2\pi t/25)$ 100 samples	$\mathcal{N}(0, 0.07)$
Synthetic system 4	$y_t = 0.3y_{t-1} + 0.6y_{t-2} + 0.3\sin(3\pi u_{t-1}) + 0.1\sin(5\pi u_{t-1})$	$u_t = \mathrm{U}(-1,1)$ 500 samples	$u_t = \sin(2\pi t/250)$ 500 samples	$\mathcal{N}(0, 0.18)$
Synthetic system 5	$y_t = y_{t-1} - 0.5\tanh(y_{t-1} + u_{t-1}^3)$	$u_t = \mathcal{N}(u_t\|0,1)$ $-1 \leq u_t \leq 1$ 150 samples	$u_t = \mathcal{N}(u_t\|0,1)$ $-1 \leq u_t \leq 1$ 100 samples	$\mathcal{N}(0, 0.0025)$

Table 3: Summary of the real-world dataset characteristics. For each dataset, the lengths of the training and test trajectories are given together with the number of historic states employed for the NARX dynamics models.

Task	Training	Test	History
Hydraulic actuator [33]	512	512	$l_y = l_u = 10$
Ball balancing [34]	500	500	$l_y = l_u = 10$
Electrical drives [35]	250	250	$l_y = l_u = 10$
Gas furnace [34]	148	148	$l_y = l_u = 3$
Hair dryer [34]	500	500	$l_y = l_u = 2$

motors. References to the individual datasets, training and test trajectory length and the utilized history for the GP-NARX models are summarized in Tab. 3.

A.2 Moment Matching

In the moment matching approximation, an intractable distribution is approximated by a Gaussian distribution having its mean and variance. In order to propagate a Gaussian input through the nonlinear GP dynamics model, the following integral has to be solved

$$p(f_{t+1}) = \int p(f_{t+1}|\bar{\boldsymbol{x}}_t)p(\bar{\boldsymbol{x}}_t)d\bar{\boldsymbol{x}}_t\,. \tag{17}$$

First and second moment of the predictive distribution can be calculated in closed form if a Gaussian-like kernel is employed as derived in [8]. The approximated predictive distribution for a Gaussian distributed input $\bar{\boldsymbol{x}}_t \sim \mathcal{N}(\boldsymbol{\mu}_{\bar{x}}, \boldsymbol{\Sigma}_{\bar{x}})$ is given by

$$p(f_{t+1}) = \mathcal{N}(\mu_{f_{t+1,\mathrm{MM}}}, \sigma_{f_{t+1,\mathrm{MM}}})\,, \tag{18}$$

where the mean is calculated as

$$\mu_{f_{t+1,MM}} = \boldsymbol{q}\,\boldsymbol{\beta}^T\,, \tag{19}$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_m]^T = \boldsymbol{K}^{-1}\bar{\boldsymbol{y}}$ and

$$q_i = |\Lambda^{-1}\boldsymbol{\Sigma}_{\bar{x}} + \boldsymbol{I}|^{-1/2}\exp(-\frac{1}{2}(\boldsymbol{\mu}_{\bar{x}} - \bar{\boldsymbol{x}}_i)^T(\boldsymbol{\Sigma}_{\bar{x}} + \Lambda)^{-1}(\boldsymbol{\mu}_{\bar{x}} - \bar{\boldsymbol{x}}_i))\,. \tag{20}$$

The variance is obtained as

$$\sigma_{f_{t+1,\mathrm{MM}}} = \sigma_{f_{t+1}}^2 + \mathrm{Tr}(\boldsymbol{K}^{-1}(\boldsymbol{k}\boldsymbol{k}^T - \boldsymbol{L})) + \mathrm{Tr}(\boldsymbol{\beta}\boldsymbol{\beta}^T(\boldsymbol{L} - \boldsymbol{q}\boldsymbol{q}^T))\,, \tag{21}$$

and $\boldsymbol{L}$ is given by

$$L_{ij} = k(\bar{\boldsymbol{x}}_t, \bar{\boldsymbol{x}}_i)k(\bar{\boldsymbol{x}}_t, \bar{\boldsymbol{x}}_j)|2\Lambda^{-1}\boldsymbol{\Sigma}_{\bar{x}} + \boldsymbol{I}|^{-1/2} \exp(2(\boldsymbol{\mu}_{\bar{x}} - \bar{\boldsymbol{x}}_d)^T\Lambda^{-1}(2\Lambda^{-1} + \boldsymbol{\Sigma}_{\bar{x}}^{-1})^{-1}\Lambda^{-1}(\boldsymbol{\mu}_{\bar{x}} - \bar{\boldsymbol{x}}_d))\,, \tag{22}$$

where $\bar{\boldsymbol{x}}_d = 0.5(\bar{\boldsymbol{x}}_i + \bar{\boldsymbol{x}}_j)$. A detailed derivation of the moment matching approximation for GPs based on the SE kernel can be found in [8].