

Safety Guarantees for Planning Based on Iterative Gaussian Processes

Kyriakos Polymenakos, Luca Laurenti, Andrea Patane, Jan-Peter Calliess,
Luca Cardelli, Marta Kwiatkowska, Alessandro Abate & Stephen Roberts
Departments of Computer Science and Engineering Science
University of Oxford

ABSTRACT

Gaussian Processes (GPs) are widely employed in control and learning because of their principled treatment of uncertainty. However, tracking uncertainty for iterative, multi-step predictions in general leads to an analytically intractable problem. While approximation methods exist, they do not come with guarantees, making it difficult to estimate their reliability and to trust their predictions. In this work, we derive formal probability error bounds for iterative prediction and planning with GPs. Building on GP properties, we bound the probability that random trajectories lie in specific regions around the predicted values. Namely, given a tolerance $\epsilon > 0$, we compute regions around the predicted trajectory values, such that GP trajectories are guaranteed to lie inside them with probability at least $1 - \epsilon$. We verify experimentally that our method tracks the predictive uncertainty correctly, even when current approximation techniques fail. Furthermore, we show how the proposed bounds can be employed within a safe reinforcement learning framework to verify the safety of candidate control policies, guiding the synthesis of provably safe controllers.

KEYWORDS

Machine learning; reinforcement learning; single- and multi-agent planning and scheduling

1 INTRODUCTION

Gaussian processes (GPs) have been extensively used for modelling due to the variety of suitable properties they possess: they are probabilistic models, providing uncertainty estimates on their predictions; they are non-parametric, effectively adjusting the model complexity to the data, and finally they are usually data-efficient [22]. In plenty of scenarios (e.g. planning, forecasting, and time-series modelling) one needs to make several, possibly correlated, predictions at once (the second prediction is made before the first one can be evaluated versus a ground truth, and so on). For this we can discern two options: either train multiple models, each one predicting at different time-scales, or use a single model, that iteratively computes predictions that get in turn fed back as input to the model in the next step. We refer to the latter as *iterative planning*.

Of particular interest for the iterative planning scenario is the model-based reinforcement learning setting, where a GP model is used to evaluate a candidate control policy on the system. The

evaluation requires the model to provide predictions for the system's state over multiple time-steps under the proposed policy. It is important in these cases to have a realistic assessment of the error on the predictions, as this allows quantification of the risk of costly system failures, like collisions with obstacles or financial losses, and analysis of safety-critical applications. In such settings, we require predictions that are not only accurate on average, but also provide robust, (probabilistically) guaranteed worst-case accuracy.

Unfortunately, as GP models output probability distributions, iterative planning poses the problem of prediction over successive *noisy inputs* (i.e. with a distribution placed over the input space). This leads to an analytically intractable problem for such non-linear input-output mappings. While several approximation techniques have been proposed [14, 23], to the best of our knowledge, none of them provides guarantees, in the form of formal error bounds on their estimations, making it difficult to estimate reliability and trust predictions in application scenarios.

In this work we provide a probabilistic bound for iterative planning with GPs and develop a framework for its explicit computation. Given a user-defined tolerance $\epsilon > 0$, our method works by computing probabilistic bounds at each prediction step and propagating them over multiple time-steps in the form of intervals. The GP trajectories are guaranteed to lie inside these intervals at each time step with probability at least $1 - \epsilon$. In practice, this allows us to perform long-term predictions for the system trajectory with the prediction provably staying within known bounds with a specified probability. We further show how the bound can be implemented within a reinforcement learning scenario, in order to train control policies with guaranteed safe behaviour. We provide an algorithmic framework for the explicit computation of every value involved in the bound calculation, directly and efficiently from data, so that the bound can be explicitly computed independently of the form of the learned GP.

On a set of case studies, including the continuous mountain car problem [20] and an inverted pendulum scenario [11], we show how our method can correctly quantify prediction errors accounting for the model uncertainty. Finally, we illustrate how our bound can be successfully employed to verify both open loop and feedback policies and therefore to guide the synthesis of provably safe controllers.

In summary, the paper makes the following main contributions:

- We develop a formal bound, for iterative prediction settings, on the probability that the trajectories of a GP lie inside a specific region. We provide explicit computational techniques for calculating the bound.

- We implement the bound in a reinforcement learning scenario. We show how it can be applied to guide safe policy search.
- We provide experimental validations of our method, highlighting cases in which a competitive state-of-the-art method fails to properly propagate the GP uncertainty. We provide case of studies on certification of open-loop and feedback policies.

2 RELATED WORK

Iterative planning is an extensively studied problem across various model types [1, 6, 15, 16, 25].

In particular, GP multi-step-ahead prediction is generally achieved using heuristic approximations [14]. One of the most used approaches is Moment Matching (MM) which computes a Gaussian approximation over the (non-Gaussian) output distribution of a GP over a noisy input [4, 14]. The uncertainty estimated in this fashion can then be leveraged to learn control policies in frameworks such as PILCO [7, 9]. During the last few years, various extensions of PILCO have been proposed [8, 10, 18, 19]. For example, in [12] GPs have been replaced with neural networks, while in [21] an emphasis on safety is given. However, building on Gaussian approximations, all the cited approaches inherently fail to take into account multimodal behaviour and tend to underestimate uncertainty. As such, the synthesised policies are not guaranteed to be safe. Our method on the other hand comes with probabilistic guarantees that allow us to compute the subregions of the input space in which the trajectories of the analysed GP are bound to lie with high probability. As such it provides formal, guaranteed bounds on the GP trajectories and makes no particular assumptions on the GP model, enabling its use in safe reinforcement learning scenarios [13].

Numerical approximations exist for multi-step-ahead predictions [23] where the output distribution is directly approximated by using quadrature formulas and, in principle, worst-case scenario error bounds could be computed using existing techniques for numerical quadrature [24]. However, the analysis that leads to the bounds proposed in [24] is focused on stability, with the assumption that trajectories monotonically decrease the distance to a target state, and the authors explicitly exclude trajectories that move away from the target state before eventual convergence and stabilisation. In [23], where more general tasks are solved, no formal bounds are provided. Our algorithm provides valid bounds for the general case.

Interestingly, [17] focus on bounding the modelling error, that is the difference between the underlying system dynamics and the learnt GP model, which is a complementary problem to the one tackled in this work. In order to do so they assume that the underlying function describing the system dynamics, that is approximated by the GP, has a bounded reproducing kernel Hilbert space norm. Using that they show that the underlying function is Lipschitz continuous, but the resulting Lipschitz constants over the trajectories of the process are still very difficult to compute in practice. To propagate uncertainty for multiple time-steps they employ moment matching, and finally obtain an iterative bound around the predicted trajectories, similar in form to the bound we are proposing. However, while we assume a GP with given hyperparameters and quantify the effect of the process's inherent stochasticity, their

work quantifies the effect of the mismatch between the GP and the underlying system being modelled.

Formal and probabilistic guarantees for GPs have been discussed in [5] and [3] for regression and classification with GPs, respectively. Albeit formal, these methods cannot be directly applied to multi-step-ahead predictions scenarios as they are designed for GPs over single input points. Whereas, our method, by propagating probabilistic bounds through each time step is applicable to multi-step ahead prediction scenarios and can be used in reinforcement learning settings for guiding the search toward safe control policies.

3 BOUNDS FOR MULTI-STEP AHEAD PREDICTIONS WITH GAUSSIAN PROCESSES

Given an input space $U \subset \mathbb{R}^m$ and a time horizon $[0, H]$, for $t \in \{0, \dots, H-1\} \subset \mathbb{N}$ we consider a stochastic dynamical system

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, u_t), \quad u_t \in U, \quad (1)$$

where we assume that for $\mathbf{x} \in X \subset \mathbb{R}^n$, $\mathbf{f}(\mathbf{x}, u_t) \sim \mathcal{N}(\mu_{\mathbf{x}}^{\mathbf{f}}, \Sigma_{\mathbf{x}, \mathbf{x}}^{\mathbf{f}})$ that is, $\mathbf{f}(\mathbf{x}, u_t)$ is normally distributed with mean vector $\mu_{\mathbf{x}}^{\mathbf{f}}$ and covariance matrix $\Sigma_{\mathbf{x}, \mathbf{x}}^{\mathbf{f}}$. Mean and variance of $\mathbf{f}^i(\mathbf{x}, u_t)$, the i -th component of $\mathbf{f}(\mathbf{x}, u_t)$, are denoted with $\mu_{\mathbf{x}}^{\mathbf{f}, i}$ and $\Sigma_{\mathbf{x}, \mathbf{x}}^{\mathbf{f}, (i, i)}$. Intuitively, \mathbf{x}_t is a discrete-time stochastic process, whose time evolution depend on an input signal taking values in U . A parametric memory-less and deterministic policy $\pi^\theta : X \rightarrow U$ with parameters θ is a function that assigns a control input given the current state. By iterating Equation 1, we have that, for $t > 0$, \mathbf{x}_t is a random variable as it is the output of process \mathbf{f} . As such multi-step ahead predictions poses the problem of predictions over noisy inputs.

3.1 Prediction over noisy inputs

For a given $\mathbf{x} \in X$, $u \in U$ we have that $\mathbf{f}(\mathbf{x}, u)$ is a Gaussian random variable. However, if \mathbf{x}_t is a random variable itself (which is the case of prediction over noisy inputs), then $\mathbf{f}(\mathbf{x}_t, u)$ is generally not Gaussian anymore and its distribution is in general analytically intractable. In particular, we have that

$$\mathbf{f}(\mathbf{x}_t, u) \sim \int p(\mathbf{x}_{t+1} | \mathbf{x}, u) p(\mathbf{x}_t = \mathbf{x}) d\mathbf{x},$$

where $p(\mathbf{x}_{t+1} | \mathbf{x}, u)$ is the (normal) distribution of $\mathbf{f}(\mathbf{x}, u)$ and $p(\mathbf{x}_t = \mathbf{x})$ is the distribution of \mathbf{x}_t . As a consequence, the predictive distribution for \mathbf{x}_{t+1} is not Gaussian and approximations are required [14].

In this paper, given \mathbf{x}_t , we consider a predictor $\hat{\mathbf{x}}_t$ for \mathbf{x}_t , such that

$$\hat{\mathbf{x}}_t = g(\hat{\mathbf{x}}_{t-1}, u_{t-1}), \quad (2)$$

where $g(\hat{\mathbf{x}}_{t-1}, u_{t-1})$ is a deterministic function. That is, $\hat{\mathbf{x}}_t$ is a deterministic process that predicts the value of \mathbf{x}_t . For instance, we could have $\hat{\mathbf{x}}_t$ equals to the mean of \mathbf{x}_t , as estimated with moment matching techniques [14], but any other deterministic function will work for the results presented in this paper.

In what follows, in Theorem 3.1 we compute a probabilistic bound on the error between $\hat{\mathbf{x}}_t$ and \mathbf{x}_t . The bound is recursive, as

¹For simplicity we drop the dependence on u_t in both mean vector and covariance matrix.

the uncertainty needs to be propagated over multiple prediction steps. Then, in Corollary 3.3 we show that, given an $\epsilon > 0$, this bound can be used to build a tube around \hat{x}_t such that at each time step the trajectories of \mathbf{x}_t are guaranteed to be within the tube with probability at least $1 - \epsilon$. For any safe region $S \subset X$ we can hence produce certificates on whether GP trajectories will lie inside that region with high probability or not.

3.2 Bounds for Multi-Step Ahead Predictions

Consider the random variable on the error at time t , i.e. $\mathbf{e}_t = |\mathbf{x}_t - \hat{x}_t|_1$ and a constant $K_t > 0$. In Theorem 3.1 we compute $P(\mathbf{e}_t > K_t)$, that is the probability that the error between \mathbf{x}_t and \hat{x}_t is greater than K_t .

THEOREM 3.1. *For any $K > 0$ and $\mathbf{x}^* \in X$, let $I_{\mathbf{x}^*}^K = \{\mathbf{x} \in X : |\mathbf{x}^* - \mathbf{x}|_1 \leq K\}$. Assume $\mathbf{x}_0 \sim \mathcal{N}(\mu_0, \Sigma_{0,0})$. Then, for arbitrary constants $K_{t+1}, K_t > 0$, it holds that*

$$P(\mathbf{e}_{t+1} > K_{t+1}) \leq P\left(\sup_{\mathbf{x} \in I_{\hat{x}_t}^{K_t}} |\hat{x}_{t+1} - \mathbf{f}(\mathbf{x}, u_t)|_1 > K_{t+1}\right) P(\mathbf{e}_t \leq K_t) + P(\mathbf{e}_t > K_t),$$

with

$$P(\mathbf{e}_0 > K_0) = 1 - \int_{I_{\mathbf{x}_0}^{K_0}} \mathcal{N}(z | \mu_0, \Sigma_{0,0}) dz$$

for any $K_0 > 0$, where μ_0 and $\Sigma_{0,0}$ are mean and covariance matrix of \mathbf{x}_0 .

The proof of the above theorem is reported in Section 6. The resulting bound in Theorem 3.1 is recursive. Hence, in order to estimate the prediction error at time t , we need to compute the prediction error at the previous time steps, which is propagated over time through the bound. The recursion terminates as the distribution for \mathbf{x}_0 , that is the initial condition, is given. Intuitively K_t is a parametric cutoff threshold for the error at time t , and the resulting error bound at time $t + 1$, that is \mathbf{e}_{t+1} , is the sum of the contribution given by assuming that $\mathbf{e}_t \leq K_t$ and by the contribution when assuming $\mathbf{e}_t > K_t$ (and remains valid for any value of K_t).

Note that the bound in Theorem 3.1 requires the computation of $P(\sup_{\mathbf{x} \in I_{\hat{x}_t}^{K_t}} |g(\hat{x}_t, u_t) - \mathbf{f}(\mathbf{x}, u_t)|_1 > K_{t+1})$ that is, the probability that the supremum of a stochastic process is greater than a given threshold. This is in general a difficult problem [2]. However, $\mathbf{f}(\mathbf{x}, u_t)$, is a Gaussian process and $g(\hat{x}_t, u_t)$ a constant. Therefore, we can use the result from [5], where bounds for the supremum of a GP have been derived. These are extended to the current setup in the following proposition.

PROPOSITION 3.2. *Let $\mu(\mathbf{x}, \hat{x}_t) = g(\hat{x}_t, u_t) - \mu_{\mathbf{x}}^{\mathbf{f}}$. Assume $I_{\hat{x}_t}^{K_t}$ is a hyper-cube with sides of length $D > 0$. For $i \in \{1, \dots, n\}$ let*

$$\bar{\eta}^i = \frac{K_{t+1} - \sup_{\mathbf{x} \in I_{\hat{x}_t}^{K_t}} |\mu(\mathbf{x}, \hat{x}_t)|_1}{n} - \frac{12 \int_0^{\lambda^i} \sqrt{\ln \left(\left(\frac{\sqrt{N} L_{\hat{x}_t}^i D}{z} + 1 \right)^n \right)} dz,}{n}$$

with $\lambda^i = \frac{1}{2} \sup_{\mathbf{x}_1, \mathbf{x}_2 \in I_{\hat{x}_t}^{K_t}} d_{\hat{x}}^{(i)}(\mathbf{x}_1, \mathbf{x}_2)$ and n being the dimension of the state space. For each $i \in \{1, \dots, n\}$ assume $\bar{\eta}^i > 0$. Then, it holds that

$$P\left(\sup_{\mathbf{x} \in I_{\hat{x}_t}^{K_t}} |g(\hat{x}_t, u_t) - \mathbf{f}(\mathbf{x}, u_t)|_1 > K_{t+1}\right) \leq 2 \sum_{i=1}^n e^{-\frac{(\bar{\eta}^i)^2}{2\xi^{(i)}}},$$

where $\xi^{(i)} = \sup_{\mathbf{x} \in I_{\hat{x}_t}^{K_t}} \Sigma_{\mathbf{x}, \mathbf{x}}^{\mathbf{f}, (i, i)}$,

$$d_{\hat{x}}^{(i)}(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\mathbb{E}[(\mathbf{f}^i(\mathbf{x}_2, u_t) - \mu_{\mathbf{x}_2}^{\mathbf{f}, i} - (\mathbf{f}^i(\mathbf{x}_1, u_t) - \mu_{\mathbf{x}_1}^{\mathbf{f}, i}))^2]}$$

and $L_{\hat{x}_t}^i$ is a local Lipschitz constant for $d_{\hat{x}}^{(i)}$.

By using the upper bound of Proposition 3.2 in Theorem 3.1 we can propagate the bound through time for any value of $K_t > 0$, $t = 0, \dots, H$. This give us the degree of freedom necessary to iteratively select, given K_t , the values for K_{t+1} that meet an a-priori specified error $\epsilon > 0$. To do this it suffice to evaluate the one-step bound resulting from the combination of Proposition 3.2 and Theorem 3.1, and choose the smallest value of K_{t+1} such that $P(\mathbf{e}_{t+1} > K_{t+1}) < \epsilon$.

COROLLARY 3.3. *(of Theorem 3.1) For any $\epsilon > 0$ pick the smallest K_0, \dots, K_H such that for any $t \in \{0, \dots, H\}$ we have that $P(\mathbf{e}_t > K_t) < \epsilon$. Then, this implies that*

$$\forall t \in \{0, \dots, H\}, \quad P(\mathbf{x}_t \in I_{\hat{x}_t}^{K_t}) > 1 - \epsilon.$$

As a result we can compute a sequence of subsets $I_{\hat{x}_t}^{K_t}$ of the state space such that the GP trajectories are bounded to stay inside them with probability at least $1 - \epsilon$ at each time step. Given a safe region $S \subseteq X$ we can hence produce a certificate on the GP trajectories lying inside S with probability at least $1 - \epsilon$ by checking the intersection between the $I_{\hat{x}_t}^{K_t}$ and S .

3.2.1 Bounds in Bayesian Learning Settings. The bound in Proposition 3.2 requires the computation of $\sup_{\mathbf{x} \in I_{\hat{x}_t}^{K_t}} |\mu^i(\mathbf{x}, \hat{x}_t)|_1$, $\xi^{(i)}$, $L_{\hat{x}_t}^i$ and λ_1 , which are related to the extrema of the mean and variance of the GP \mathbf{f} in $I_{\hat{x}_t}^{K_t}$ and to a Lipschitz constant on $d_{\hat{x}}^{(i)}$. In a Bayesian learning setting, these can be computed by relying on the methods discussed in [5] and applying them to the GP of Equation 1. We here briefly review and adapt to the current settings the methods for the bounding of $\sup_{\mathbf{x} \in I_{\hat{x}_t}^{K_t}} |\mu(\mathbf{x}, \hat{x}_t)|_1$, $\xi^{(i)}$, while we refer to [5]

for a detailed explanation of how to compute $L_{\hat{x}_t}^i$ and λ_1 .

Let $k^i(\cdot, \cdot)$ be the GP kernel function for the i -th output dimension, $\mathbf{x} \in I_{\hat{x}_t}^{K_t}$ be a test point and $\mathcal{D} = \{\mathbf{x}_j, y_j \mid j = 1, \dots, M\}$ a training data set. Then the mean and variance of the Gaussian process \mathbf{f} conditioned on the training data is given by the set of equations [22]:

$$\mu_{\mathbf{x}}^{\mathbf{f}, i} = k(\mathbf{x}, \mathcal{D}) k^i(\mathcal{D}, \mathcal{D})^{-1} \mathbf{y} \quad (3)$$

$$\Sigma_{\mathbf{x}, \mathbf{x}}^{\mathbf{f}, (i, i)} = k(\mathbf{x}, \mathbf{x}) - k^i(\mathbf{x}, \mathcal{D}) k^i(\mathcal{D}, \mathcal{D})^{-1} k^i(\mathbf{x}, \mathcal{D})^T \quad (4)$$

where $\mathbf{y} = [y_1, \dots, y_M]$. Assuming continuity and differentiability of the kernel function $k(\cdot, \cdot)$, it is possible to find linear upper and lower bounds on the covariance between a test point and a point in the training dataset. In the case of squared exponential kernel it suffice to see that the covariance between a test point \mathbf{x} and a

training point x_j can be written as a differentiable, convex function of the uni-dimensional auxiliary variable $z_j = \|x^* - x_j\|$. As such, by inspection of the derivatives it is possible to find linear coefficients a_j^L, b_j^L, a_j^U and b_j^U such that²:

$$a_j^L + b_j^L \|x - x_j\| \leq k^i(x, x_j) \leq a_j^U + b_j^U \|x - x_j\| \quad \forall x \in I_{x_t}^{K_t} \quad (5)$$

These bounds can be propagated through the inference formula for \mathbf{f} by performing the matrix multiplication involved in Equations 3 and 4. The resulting equation for the mean and variance are respectively linear and quadratic on the auxiliary variable $z_j = \|x^* - x_j\|$, and can hence be optimised analytically by inspection of the derivatives. This can then be further refined using a branch and bound optimisation approach over $I_{x_t}^{K_t}$.

This can be straightforwardly generalised to take into account the extra input dimensions coming from a deterministic control strategy $\pi(x) = u$, without increasing the size of the branch and bound search space, that is without significantly change the computational time. To do so it suffices to solve the optimisation problems $u_j^L = \min_{x \in I_{x_t}^{K_t}} \pi_j(x)$ and $u_j^U = \max_{x \in I_{x_t}^{K_t}} \pi_j(x)$ for $j = 1, \dots, m$, that is computing maximum and minimum of the control allowed in the current state space sub-region. Notice that for policies function generally implemented (e.g. linear or sum of radial basis functions) this can be computed analytically and in constant time [9]. The bounds can then be used by treating u and x symmetrically.

3.2.2 Computational Complexity of Bound. Computation of the bound involves the calculation of Equation 5 for each point in the training set \mathcal{D} , and the computation of the inference formulas 3 and 4 on the resulting bounds. This is $\mathcal{O}(M)$ for the mean function and $\mathcal{O}(M^2)$ for the variance (as the latter is quadratic), where $M = |\mathcal{D}|$ is the number of training samples used. Refining the bounds with a branch and bound approach has a worst-case cost that is exponential in the dimension of the variable x , that is n . Bounding of $L_{x_t}^i$ and λ_1 is done in constant time. This is iterated for any output dimension of the GP. After branch and bound converged, computation of the optimal value for K_{t+1} is linear on the number of candidate values explored, as it involves the computation of the integral in Proposition 3.2 with known constants. Finally the procedure is identically repeated for each time step t .

3.3 Extending PILCO with Safety Guarantees

In this section, we incorporate the bounds proposed above in a safe, model-based policy search algorithm, which extends the Safe PILCO framework [21].

PILCO's goal is to control an unknown dynamical system throughout a task, by efficiently optimising the parameters θ of a feedback control policy π^θ , implemented originally as a linear controller or a sum of radial basis functions. In [21], safety considerations are added, with the introduction of constraints, that demand the system to stay in a safe subset of the state space $S \subseteq X$ with high probability. To achieve that, a composite objective function is introduced that combines safety and performance as $J^\pi(\theta) = R^\pi(\theta) - wQ^\pi(\theta)$, with R quantifying performance, Q

quantifying the probability of constraint violations, and $w > 0$ the weight parameter that regulates the trade-off. The objective function is adaptively tuned, by changing w , based on a safety check, that estimates the probability of constraint violation, before a policy is implemented. However, since this probability estimation is based on moment matching, it is vulnerable to the underestimation of uncertainty we highlighted above. Therefore, we'll use the results of 3.3 to verify that the proposed policies are indeed safe, before implementing them on the system.

The qualitative steps of the procedure we consider in this paper are presented in Algorithm 1. In Steps 1 – 2 we collect data by generating random policies, which are then used in Step 3 to train a GP model. Then, in Steps 4 – 13 we iteratively train candidate policies, evaluate their safety and refine policies and models until a sufficiently well performing safe-policy is synthesized.

In more detail, Step 5 employs the safe PILCO framework to synthesise a candidate policy (optimising its parameters θ) by maximising the composite objective function $J^\pi(\theta)$. Note that the objective function component related to safety, Q , can take various forms as long as it is correlated with the probability of violating the constraints. In this work we use the probability of constraint violations as estimated by moment matching, in accordance with the conclusions of [21]. In principle, the results of Corollary 3.3 could also be used in this step to replace moment matching. We do not do that because Step 5 requires the policies to be evaluated multiple times, depending on the optimiser's budget (in current experiments 50-100 times). Hence, the computational burden required to recompute the bounds so many times may make policy optimisation very costly computationally. Furthermore, note that in safe PILCO moment matching allows one to obtain analytic gradients of the objective function with respect to the parameters θ of the policy π^θ , which would not be available using Corollary 3.3.

In Step 6, Q , the probability that a given policy violates the constraints, is upper-bounded using Corollary 3.3. That is, for the policy resulting from Step 5, we a posteriori verify that, at each time-step, the probability of the system's state being out outside the safe set of states $Pr(x_t \notin S)$ is at most ϵ . If the verification succeeds the policy is implemented on the system, otherwise implementation is prohibited, and the policy optimisation of Step 5 is repeated with an adapted objective function. Hence, we can guarantee that only policies that are safe with high probability are implemented on the system throughout training and as a consequence, Algorithm 1 guarantees that exclusively safe policies are synthesised at the end of a successful algorithm run.

It's worth noting that it is possible for the the algorithm to fail to propose a safe policy. This is a legitimate outcome, since, firstly, a safe policy might not exist. Alternatively, such a result might indicate a lack of informative data, leading to higher predictive uncertainty and hence wide bounds that overlap with constraints. Finally the gradient-based optimisation employed by PILCO is local, and it cannot in general guarantee to find a globally optimal solution.

4 EXPERIMENTS

In this section we apply the methods presented above to various GPs with SQE kernel trained from data. In all the experiments we use

²By definition of convex function, a lower bound is given by any tangent to the function (computed through derivative calculations) and an upperbound is given by connecting the extrema of the function in $I_{x_t}^{K_t}$.

Algorithm 1 High level algorithm description. Notation: θ denotes the policy parameters, J the objective function, R, Q the performance and safety components of the objective function respectively and w the weight given to safety.

```

1: Initialise policy parameters  $\theta$ 
2: Interact with the system, collect data
3: Train GP model on the data
4: repeat
5:   Policy optimisation, maximising  $J^\pi(\theta) = R^\pi(\theta) + wQ^\pi(\theta)$ 
6:   Calculate bounds for chosen  $\epsilon$  from Theorem 3.1
7:   if Bounded trajectory satisfies constraints with probability
       at least  $1 - \epsilon$  then
8:     Interact with the system, collect data
9:     Retrain GP model on the new data set
10:  else
11:    Return to policy optimisation with an adapted objective
       function that has higher  $w$ 
12:  end if
13: until task learned (satisfactory performance achieved or lim-
       ited computational time/number of iterations reached.)

```

the bound from Theorem 3.1 with the L1 norm, that is with $d = 1$. First we explicitly compare our formal, guaranteed bounds with the probability estimation obtained by Moment Matching (MM) in two iterative prediction scenarios (that is, when no control is involved). We then investigate in the Mountain Car application [20] the behaviour of our methodology for certification of a given control policy. Finally we show how to compute safe-policies by using the full methodology discussed in Algorithm 1 in an inverted pendulum application³ [11]. GPs are trained with the GPML package, using maximum marginal likelihood for hyperparameter selection. Trained policies are obtained from PILCO, as linear controllers squashed through a sin wave to constrain the input magnitude. All the experiments were run in a MacBook Pro (Early 2015) with a DDR3 8 GB RAM @1867 MHz, and an Intel Core i7 processor @3.1 GHz.

4.1 Iterative Prediction

We analyse the behaviour of our method in a one-dimensional synthetic dataset where the system dynamics are distributed as a Gaussian at each time step. Further we assume that the initial state of the system is Gaussian, that is $\mathbf{x}_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$, with mean and variance given by $\mu_0 = 0$ and $\Sigma_0 = 0.01$. We compute predictions and bound the trajectory for an horizon of $H = 10$ time steps. We use $\epsilon = 0.05$, that is we require bounds holding with probability at least 95% and compare with the results obtained by MM. Namely, we compare our bounds with plus/minus two standard deviation estimated by MM. Notice that when MM is exact (i.e. when the system dynamics are effectively Gaussian at each time-step), then this would as well correspond to bounds at 95% probability. Results for this analysis are given in Figure 1, where our bound is depicted with a thick red solid line, and MM results are represented by the green shaded area. Further, we extract 100 trajectories from the GP, which are depicted with thin colored lines, in order to

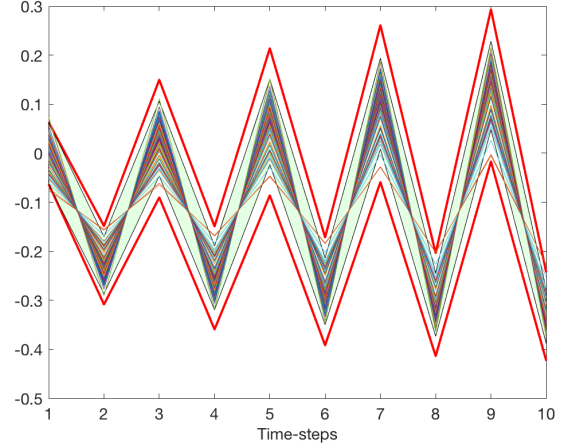


Figure 1: A set of 100 trajectories sampled from a GP (thin coloured line). The green shaded area corresponds to plus/minus two standard deviations of the moment matching prediction, and the thicker red lines delimit the area with 95% probability according to Theorem 3.1.

provide statistical validation for the results. Notice that the latter are almost entirely within the MM shaded area. In fact, since the system dynamics are fully Gaussian at each time step, that is \mathbf{x}_t is Gaussian for each t , then the approximation made by MM is almost exact and well behaved. Notice that our method successfully bounds the sampled trajectories at each time step.

Notice that MM succeeds in bounding the GP trajectories because the Gaussian approximation performed by MM is well suited for the example above. However, as soon as this does not hold anymore, then the results obtained with MM fail to bound the actual GP trajectories. As an example of this, consider a system with dynamics given by:

$$h(x) = \begin{cases} \text{sign}(x)x^4, & \text{if } |x| < 1 \\ x, & \text{otherwise.} \end{cases} \quad (6)$$

We train a GP on data sampled from the dynamic described by this system. This is further depicted in Figure 2. With the function being non-linear, we have that \mathbf{x}_t is non-Gaussian for $t > 0$, which implies that application of MM will introduce unaccounted approximation errors. Furthermore, the specific dynamics chosen are such that MM variance prediction will inevitably shrink, leading to a systematic underestimation of the actual region in which GP trajectories are located. In fact when the initial position of the trajectory, x_0 , is greater than 1, then the trajectory will constantly be at x_0 . As such, assuming $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$, one can choose a Σ_0 big enough such that the GP trajectories will be outside any tube parallel to the x -axis with probability greater than ϵ . However after finitely-many time steps MM variance will wrongly shrink to values very close to zero, hence failing to account for the majority of the probability mass of the GP.

Empirical results for this system using $\epsilon = 0.05$ are plotted in Figure 3, for values of initial variance Σ_0 going from 0.1 to 0.6. Notice that the empirical results agree with the discussion

³An implementation can be found at: [link blanked during peer review](#).

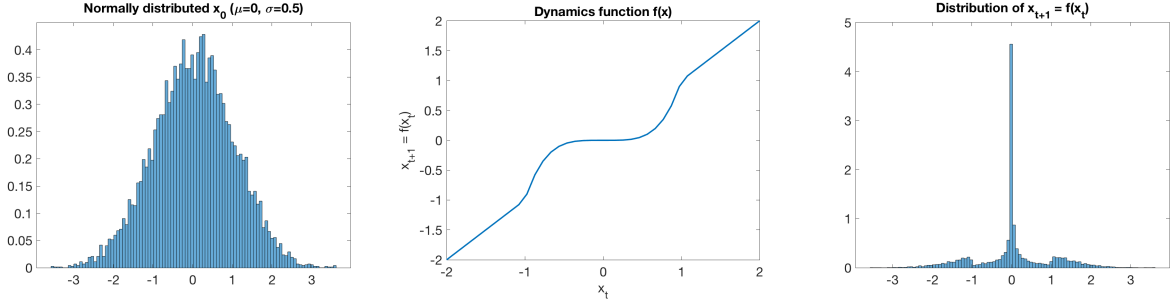


Figure 2: Initial state distribution, system dynamics and state distribution after a time-step for the system described by the set of Equations 6. Histograms show empirical results for 10000 trajectories. On the left is the normally distributed initial state, which passes through the nonlinear dynamics function in the middle, leading to the distribution on the right.

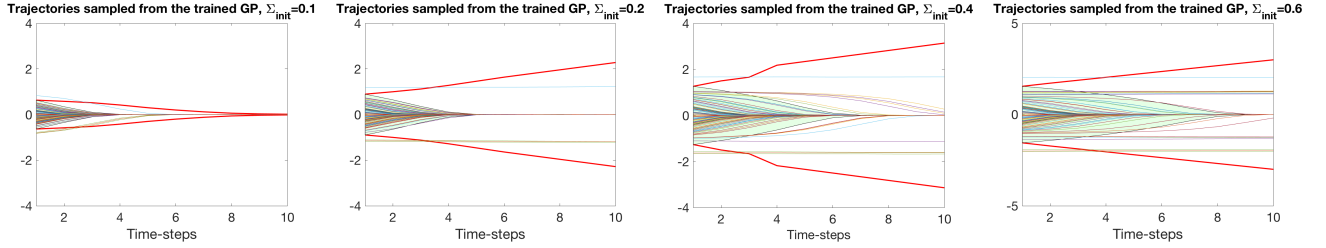


Figure 3: As the initial variance increases, more trajectories, having an initial state $|x_0| > 1$, do not converge to 0. Moment matching fails to account for this fact (green shaded area showing two standard deviations). Our bound (red line) grows appropriately. Thinner colored lines represent 100 sampled trajectories from the GP.

above. If the initial variance is small enough, then the overwhelming majority of GP trajectories will converge to zero. However, as the initial variance grows, more and more trajectories diverge. MM fails to account for this behavior, and the variance predicted by MM will fail to mirror the actual dynamics of the GP under analysis. Notice that, being guaranteed to provide correct results, our method is able to successfully bound (up to $1 - \epsilon$ probability) the actual trajectory of the GP, independently of the initial variance. In fact, our method does not rely on any particular assumption, and is able to provide worst-case scenario analysis independently of the general shape of the GP under investigation.

4.2 Open-loop control for Mountain Car

In this Section we show how our method can be used to certify a control input for a dynamical system. The environment we are considering is a version of the continuous mountain car problem [20]. Briefly, a car is supposed to go up a hill to its right, with a goal state on top of the hill as shown in Figure 4a. Because it does not have enough power to climb the hill directly, it has to go up a hill to the left first to gather momentum. The state space has two dimensions (position and velocity of the car), and the control input is one dimensional and corresponds to a force applied to the car.

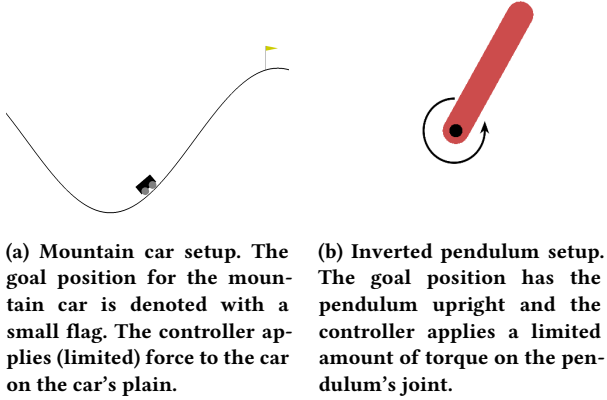
As previously, we train a GP on data generated from the environment, in this case following a random policy. We assume we have access to an initial normal distribution for the starting state and we want to evaluate a proposed sequence of actions. Specifically, we

want to perform predictions about the sequence of states (position and velocity) of the car, and to provide high probability bounds for these predictions.

The trained GP model has a 3-dimensional input space, as it takes (x_t, u_t) pairs as inputs, corresponding to the two state-space variables and the control input, and 2-dimensional outputs, that correspond to x_{t+1} . The two output dimensions correspond to two independent GPs, each one predicting a state variable. However, the predictions of each model are based on the previous predictions of *both* models. In more detail, assume a state $x_t \in X \subset \mathbb{R}^2$, where both components of x_t are bounded. These form a tuple $[x_t^1, x_t^2, u]$, where $x_t^1 \in [lb_1, ub_1]$, and $x_t^2 \in [lb_2, ub_2]$, and the exact value of u is known (as we are verifying an arbitrary, fixed control policy). This tuple is the input to the two GP models, with one of them providing the predicted position x_{t+1}^1 , with its new lower and upper bound, and the other one providing the same quantities for the velocity x_{t+1}^2 .

We train the GP model on a dataset of 500 randomly selected actions from the mountain car system dynamics. Now, for a proposed sequence of actions, we can bound the predicted trajectories, using our methods with $\epsilon = 0.1$ (that is bound with 90% probability). Results from a typical run are presented in Table 1. Drawing 1000 trajectories from the mountain car system we verify that empirically more than 90% (91.6%) of them stay within the bounded area around the predictions obtained by our bound and in fact, when

⁴The renderings come from <https://gym.openai.com/>

Figure 4: Renderings of the two environments.⁴

t	Control u	x^1	x^2	Bound x^1	Bound x^2
1	1.85	-0.50	0.00	0.020	0.020
2	-0.97	-0.38	0.53	0.030	0.080
3	1.39	-0.37	-0.49	0.055	0.125
4	0.17	-0.53	-0.20	0.105	0.220
5	-1.95	-0.57	-0.02	0.130	0.405
6	-	-0.87	-0.05	0.225	0.595

Table 1: Predictions along with 90% probability bounds for a sequence of 5 actions applied to the mountain car. Columns x^1 and x^2 report the average value of position and velocity of the car at each time step. Columns Bound x^1 and Bound x^2 report the dimension of the interval around x^1 and x^2 containing at least 90% of the trajectories as computed by Theorem 3.1

treating each dimension separately, they both respect the bounds for more than 95% of trajectories drawn.

4.3 Safe control for inverted Pendulum

Here we present an example of our bound being used in combination with Safe PILCO [21], to learn a provably safe closed-loop control policy for a version of the inverted pendulum scenario [11] following Algorithm 1. We assume the bottom end of the pendulum is fixed, and a controller that can apply torque to that joint, which is enough to counterbalance gravity, as long as the pendulum's angle is close to the upright position (up to $\sim 11.5^\circ$ in our case). The pendulum is initialised close to the goal position, (initial standard deviation is 5°), and the controller's task is to bring, and keep, the pendulum upright. See Figure 4b for a schematic representation. This is of course a task better suited for a feedback controller rather than a predetermined sequence of actions. We use the PILCO implementations to learn a linear controller squashed through a sin function. We apply our bound with $\epsilon = 0.1$ to obtain predictions on the pendulum position.

t	Bound x^1	Bound x^2
1	0.132	0.017
2	0.172	0.1131
3	0.174	0.038
4	0.161	0.024
5	0.149	0.032
6	0.146	0.027
7	0.140	0.103
8	0.140	0.035
9	0.140	0.182
10	0.149	0.039

Table 2: 90% probability bounds for the predicted state variables, angle (x^1) and angular velocity (x^2), of the inverted pendulum. Columns Bound x^1 and Bound x^2 report the dimension of the interval around 0 containing at least 90% of the trajectories as computed by Theorem 3.1. The bounds guarantee that with high probability the controller guides the system to a bounded area around the goal position.

We collect data by applying random policies to the system and then we run PILCO for a small number of iterations (3) until a reasonable policy is obtained. Finally, we calculate the formal bounds on the possible generated trajectories to certify whether the policy is safe. In this setting, intuitive constraints can be posed on the angle of the pendulum, at the critical angles where the controller can no longer overpower gravity ($0.2 \text{ radians} \approx 11.5^\circ$).

In Table 2 we present the resulting predictions and bounds from a run of the proposed algorithm showing how the synthesized feedback policy keeps the system around the origin with high probability.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we derived a new formal probabilistic bound for iterated predictions with a GP model, and we showed how the resulting bound can be embedded in the PILCO algorithm, leading to a framework with theoretical guarantees on the propagation of predictive uncertainty. Our approach does not make any further assumptions on the properties of the GP, other than knowledge of the kernel hyperparameters, generally learnt through maximum marginal likelihood, and every intermediate quantity used is calculated directly from the data. The experimental results we report show that our method is able to correctly propagate uncertainty even when existing heuristic approaches fail. Furthermore, they showcase how our method can be used for provably safe controller synthesis on GPs, which constitute a very rich class of stochastic models. In future work, we want to quantify the modelling error (i.e., the error performed in learning the ground truth in the GP training) and its effect on the proposed bounds, as well as scaling up our approach

to settings of higher complexity, by carefully managing the added computational cost.

6 PROOFS

Proof of Theorem 3.1. In order to prove Theorem 1 we first have to consider the following Lemma

LEMMA 6.1. *Let $\mathbf{f}(x)$ be a stochastic process. Consider measurable sets A and B . Then, it holds that*

$$P(\mathbf{f}(y) \in A | y \in B) \leq P(\sup_{y \in B} \mathbf{f}(y) \in A).$$

PROOF. (Sketch) To prove Lemma 6.1 it is enough to note that for each realization of \mathbf{f} , $y \in B$, and measurable g we have that $g(\mathbf{f}(y)) \leq \sup_{y^* \in B} g(\mathbf{f}(y^*))$. Hence, we can conclude by taking the expectation. \square

Now the following calculations follow

$$\begin{aligned} & P(\mathbf{e}_{t+1} > K_{t+1}) \\ & \text{(By Definition of } \mathbf{e}_t) \\ &= P(|g(\hat{\mathbf{x}}_t, \mathbf{u}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)|_1 > K_{t+1}) \\ & \text{(By Marginalising with the events } \mathbf{e}_t > K_t, \mathbf{e}_t \leq K_t) \\ &\leq P(|g(\hat{\mathbf{x}}_t, \mathbf{u}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)|_1 > K_{t+1} | \mathbf{e}_t \leq K_t) P(\mathbf{e}_t \leq K_t) \\ & \quad + P(\mathbf{e}_t > K_t) \\ & \text{(By Lemma 6.1)} \\ &\leq P(\sup_{\mathbf{x} \in I_{\hat{\mathbf{x}}_t}^{K_t}} |g(\hat{\mathbf{x}}_t, \mathbf{u}_t) - \mathbf{f}(\mathbf{x}, \mathbf{u}_t)|_1 > K_{t+1}) P(\mathbf{e}_t \leq K_t) \\ & \quad + P(\mathbf{e}_t > K_t) \\ & \text{(By the fact that } P(\mathbf{e}_t \leq K_t) = 1 - P(\mathbf{e}_t > K_t)) \\ &= P(\sup_{\mathbf{x} \in I_{\hat{\mathbf{x}}_t}^{K_t}} |g(\hat{\mathbf{x}}_t, \mathbf{u}_t) - \mathbf{f}(\mathbf{x}, \mathbf{u}_t)|_1 > K_{t+1}) (1 - P(\mathbf{e}_t > K_t)) \\ & \quad + P(\mathbf{e}_t > K_t). \end{aligned}$$

ACKNOWLEDGMENTS

This work has been partially supported by the EU’s Horizon 2020 program under the Marie Skłodowska-Curie grant No 722022, EP-SRC AIMS CDT grant EP/L015987/1, and Schlumberger.

REFERENCES

- [1] Alessandro Abate. 2017. Formal verification of complex systems: model-based and data-driven methods. In *Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design*. ACM, 91–93.
- [2] Robert J Adler and Jonathan E Taylor. 2009. *Random fields and geometry*. Springer Science & Business Media.
- [3] Arno Blaas, Luca Laurenti, Andrea Patane, Luca Cardelli, Marta Kwiatkowska, and Stephen Roberts. 2019. Robustness Quantification for Classification with Gaussian Processes. *arXiv preprint arXiv:1905.11876* (2019).
- [4] Joaquin Quinonero Candela, Agathe Girard, Jan Larsen, and Carl Edward Rasmussen. 2003. Propagation of uncertainty in Bayesian kernel models-application to multiple-step ahead forecasting. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings (ICASSP’03)*, Vol. 2. IEEE, II–701.
- [5] Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, and Andrea Patane. 2019. Robustness guarantees for Bayesian inference with Gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7759–7768.
- [6] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*. 4754–4765.
- [7] Marc Peter Deisenroth. 2010. *Efficient reinforcement learning using Gaussian processes*. Ph.D. Dissertation. Karlsruhe Institute of Technology.
- [8] Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox. 2014. Multi-task policy search for robotics. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3876–3881.
- [9] Marc Peter Deisenroth and Carl Edward Rasmussen. 2011. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *In Proceedings of the International Conference on Machine Learning*.
- [10] Marc Peter Deisenroth, Carl E. Rasmussen, and Dieter Fox. 2011. Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning. In *Robotics: Science and Systems*.
- [11] Kenji Doya. 2000. Reinforcement learning in continuous time and space. *Neural computation* 12, 1 (2000), 219–245.
- [12] Yarin Gal, Rowan Thomas McAllister, and Carl Edward Rasmussen. 2016. Improving PILCO with Bayesian Neural Network Dynamics Models. In *Data-Efficient Machine Learning workshop*, Vol. 951. 2016.
- [13] Javier Garcia and Fernando Fernández. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research* 16 (2015), 1437–1480. <http://jmlr.org/papers/v16/garcia15a.html>
- [14] Agathe Girard, Carl Edward Rasmussen, Joaquin Quinonero Candela, and Roderrick Murray-Smith. 2003. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *Advances in neural information processing systems*. 545–552.
- [15] Michael Green and David JN Limebeer. 2012. *Linear robust control*. Courier Corporation.
- [16] Gregory Kahn, Adam Villafior, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. 2017. Uncertainty-Aware Reinforcement Learning for Collision Avoidance. [abs/1702.01182](https://arxiv.org/abs/1702.01182) (2017). [arXiv:1702.01182](https://arxiv.org/abs/1702.01182) [http://arxiv.org/abs/1702.01182](https://arxiv.org/abs/1702.01182)
- [17] Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. 2018. Learning-based Model Predictive Control for Safe Exploration and Reinforcement Learning. *CoRR* [abs/1803.08287](https://arxiv.org/abs/1803.08287) (2018). [arXiv:1803.08287](https://arxiv.org/abs/1803.08287) [http://arxiv.org/abs/1803.08287](https://arxiv.org/abs/1803.08287)
- [18] Andras Gabor Kupcsik, Marc Peter Deisenroth, Jan Peters, and Gerhard Neumann. 2013. Data-efficient generalization of robot skills with contextual policy search. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.
- [19] Rowan McAllister and Carl Edward Rasmussen. 2017. Data-Efficient Reinforcement Learning in Continuous State-Action Gaussian-POMDPs. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 2040–2049.
- [20] Andrew William Moore. 1990. *Efficient Memory-based Learning for Robot Control*. Ph.D. Dissertation. University of Cambridge.
- [21] Kyriakos Polymenakos, Alessandro Abate, and Stephen Roberts. 2019. Safe Policy Search Using Gaussian Process Models. In *Proceedings of the 18th International Conference on Autonomous Agents and Multi Agent Systems*. IFAAMS, 1565–1573.
- [22] C. E. Rasmussen and C. K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. (2006).
- [23] J. Vinogradskaya, B. Bischoff, J. Achterhold, T. Koller, and J. Peters. 2018. Numerical Quadrature for Probabilistic Policy Search. *IEEE Transactions on Pattern Analysis & Machine Intelligence* (2018).
- [24] Julia Vinogradskaya, Bastian Bischoff, Duy Nguyen-Tuong, Anne Romer, Henner Schmidt, and Jan Peters. 2016. Stability of controllers for Gaussian process forward models. In *International Conference on Machine Learning*. 545–554.
- [25] Tung-Long Vuong and Kenneth Tran. 2019. Uncertainty-aware Model-based Policy Optimization. *arXiv preprint arXiv:1906.10717* (2019).