Approximate Value Iteration Based on Numerical Quadrature

Julia Vinogradska[1,2], Bastian Bischoff[1], Jan Peters[2,3]

***Abstract*—Learning control policies has become an appealing alternative to the derivation of control laws based on classic control theory. Value iteration approaches have proven an outstanding flexibility, while maintaining high data efficiency when combined with probabilistic models to eliminate model bias. However, a major difficulty for these methods is that the state and action spaces must typically be discretized and often the value function update is analytically intractable. In this paper, we propose a projection based approximate value iteration approach, that employs numerical quadrature for the value function update step. It can handle continuous state and action spaces and noisy measurements of the system dynamics while learning globally optimal control from scratch. In addition, the proposed approximation technique allows for upper bounds on the approximation error, which can be used to guarantee convergence of the proposed approach to an optimal policy under some assumptions. Empirical evaluations on the mountain benchmark problem show the efficiency of the proposed approach and support our theoretical results.**

***Index Terms*—Optimization and Optimal Control, Probability and Statistical Methods**

I. INTRODUCTION

LEARNING control has become a viable approach in both the machine learning and control community. Many successful applications impressively demonstrate the advantages of learning control [1], [2], [3], [4], [5], [6], [7]. In contrast to classical control methods, learning control does not presuppose a detailed understanding of the underlying dynamics but tries to infer the required information from data. Thus, relatively little expert knowledge about the system dynamics is required and fewer assumptions, such as a parametric form and parameter estimates, must be made.

In real-world applications of learning control, the system is typically subject to wear and tear, measurements are often costly and time consuming as learning proceeds in real time time on a real system. Thus, it is highly desirable to minimize the system interaction time required for learning. As a consequence, approaches that explicitly learn a dynamics model are often preferred, as model-free methods can require a prohibitive amount of system interactions [8], [9], [10], [11]. However, one drawback of model-based methods is that modeling errors can derail learning, as the inherently approximate and frequently highly erroneous model is implicitly assumed to approximate the real dynamics sufficiently well [12]. Thus, solutions to the approximate control problem might result in policies that do not solve the control task for the true system dynamics. This model bias can have severe consequences especially when little data is available and, thus, the model predictions are highly uncertain. Hence, employing Gaussian processes (GPs) as forward models for learning control is particularly appealing as they incorporate uncertainty about the system dynamics estimate. GPs infer a distribution over all plausible models given the observed data instead of compromising on an approximate model and, thus, avoid severe modeling errors. Another major advantage of GP forward models is that stability analyses for such closed-loop control systems are available [13], [14], including automatic tools [14] which do not require any expert knowledge.

A well-studied class of approaches to policy learning are value function based methods. These methods compute the value function, that maps each state to the expected long-term reward when starting in this state and following an optimal policy. The optimal policy can then be inferred by greedily optimizing the value function. Variations of this approach include iterative computation of the value function (value iteration) and alternating between computation of the value function for a fixed policy and improving this policy based on the value function (policy iteration), see e.g., [15], [16]. For finite state and action spaces, these approaches solve the optimal control task exactly and yield globally optimal policies. However, for continuous state and action spaces analytical solutions are only known for linear dynamics with quadratic cost and Gaussian noise [17].

In this paper, we introduce AVINQ (Approximate Value Iteration based on Numerical Quadrature), a probabilistic value iteration approach that approximates the value function iterates by projection onto a linear space of learned features. The dynamics is modeled as a GP and learned in an episodic setting. To enable projection of the (analytically intractable) value function iterates, numerical quadrature is employed. The proposed approach can learn globally optimal control from scratch, i.e., a policy that is optimal given all observations of the system dynamics. Employing GPs as forward dynamics models, AVINQ accurately handles the model uncertainty and is highly data efficient. It requires no manual effort or problem specific knowledge. Furthermore, as the quadrature error can be upper bounded, under some assumptions it can be shown

Manuscript received: September, 10, 2017; Revised December, 01, 2017; Accepted January, 15, 2018.

This paper was recommended for publication by Editor Dezhen Song upon evaluation of the Associate Editor and the Reviewer's comments.

[1]Julia Vinogradska and Bastian Bischoff are with Bosch Center for Artificial Intelligence (BCAI), Robert Bosch GmbH, Renningen, Germany `julia.vinogradska@de.bosch.com`, `bastian.bischoff@de.bosch.com`

[2]Julia Vinogradska and Jan Peters are with Intelligent Autonomous Systems Lab (IAS), Technische Universität Darmstadt, Darmstadt, Germany `mail@jan-peters.net`

[3]Jan Peters is with Max Planck Institute for Intelligent Systems, Tübingen, Germany

Digital Object Identifier (DOI): see top of this page.

that AVINQ converges to a globally optimal policy.

The paper will be organized as follows: first, we briefly review related work. In Sections II-B and II-D, we specify the considered problem and give an overview of value iteration approaches. Section III introduces the proposed algorithm, which is evaluated on multiple benchmark tasks in Section IV. A conclusion summarizes and discusses the provided results (Section V).

II. PRELIMINARIES

We begin with a brief review of related work. Subsequently, we give a formal description of the problem we consider in this paper. Finally, we recap some fundamental concepts of reinforcement learning, especially the class of value function based approaches.

A. Related Work

The problem of deriving control laws when uncertainty is present has been considered in classic control theory for many years. In controller design, uncertainty may arise from modeling inaccuracies, the presence of (external) disturbances and the lack of data, as some system parameters are not known in advance but only during operation. Thus, a controller must be designed for a family of systems that is specified, e.g., by bounds for model parameters or for nonparametric uncertainties as bounds for the operator norm of the unknown dynamics. Robust control [18], [19] designs a single controller that is provably stable for all systems in the specified family – often at cost of overall controller performance. Adaptive control [20], [21] instead adjusts control parameters online in order to achieve prespecified performance, which can be computationally demanding. These methods rely on parametric dynamics models, which must be specified by an expert for each problem. In addition, both schemes require stability analysis of the dynamics system, e.g., via manually designed Lyapunov functions, which can be extremely challenging for complex, nonlinear systems.

Nonparametric system dynamics models are very appealing due to their high flexibility. They learn a dynamics model from data instead of relying on expert knowledge to pick a sufficiently accurate parametric form suitable for the dynamics. Nonparametric regression methods that learn the system dynamics from data have been considered, e.g., in [1], [7], [22], [2], [12]. In [12], locally weighted Bayesian regression has been employed to model the system dynamics and uncertainty was treated as noise. To learn a policy, stochastic dynamic programming was applied on the discretized state space. The approaches [1], [7], [22], [2] model the forward dynamics as a Gaussian process.

Gaussian processes as forward models allow to incorporate uncertainty about the system dynamics without the need to discretize the state space. GPs have also been employed to model the system dynamics in [1], [23], [24], [25], [26], [22]. The approaches PILCO [1] and GPREPS [22] are policy search approaches that rely on GPs as forward dynamics models. The PILCO algorithm [1] employs GPs as forward dynamics models and conducts a search in the policy space, performing gradient ascent on the expected reward in an episodic setting. However, policy search approaches assume a parametric form of the policy to be given and typically optimize locally around a trajectory. Obtaining globally optimal policies with policy search is challenging and involves manual effort, e.g., choosing representative trajectories to be optimized simultaneously.

In contrast to policy search methods, value function based approaches do not impose limitations on the policy and learn globally optimal policies. Model-free approaches based on a value function [27], [28], [29] are a viable choice for small, finite state spaces or when the system is easily accessible and system interactions are not costly. For small, finite state and action spaces, tabular methods that store a complete list of values for all states and actions are very well suited. However, even for finite, but large state spaces the memory requirements of tabular methods are often infeasible. Furthermore, for continuous state and/or action spaces no list of values can possibly be made. Thus, for most problems approximation of the value function is inevitable. For example, Neural Fitted Q Iteration [27] trains a neural network to fit the Q function. This approach can handle large state and action spaces, but it requires many system interactions to train the neural network. Another class of commonly employed approximations are linear representations of the value function. These approaches approximate the value function by a linear combination of basis functions (or *features*) and offer easily interpretable representations while maintaining a high flexibility.

Many methods to automatically select good basis functions have been proposed [30], [31]. For example, [30] clusters states of similar behavior/similar Bellman residual and is applicable to finite state and action spaces. The methods proposed in [31], [32] greedily select optimal basis functions to add from a prespecified feature dictionary. However, these methods involve manual effort for choosing a suitable feature dictionary and typically require many basis functions as the features are not adjusted to the particular function.

In real-world scenarios, e.g., control of robotic systems, the required number of samples renders model-free value function based methods impractical. Value function based approaches that learn a GP dynamics model were presented in [25], [26], [23]. The GPRL approach [23] assumes the dynamics to be given as a GP and computes the value function at a fixed set of support points, that are used to train a noise-free GP to represent the value function. This approach is extended to learn the system dynamics and automatically select support points for the value function by the PVI algorithm [26]. GPDP [25] also employs GPs to represent the value function for either a given dynamics model or a learned GP forward dynamics. However, the approximation error of these approaches cannot be estimated straightforwardly as they rely on moment matching [33] as approximate inference which does not provide upper bounds for the error. Furthermore, the learning success is highly dependent on many hyperparameters as, e.g., the support points or the hyperparameters of the employed selection criteria, that must be hand-tuned. To the best of our knowledge, no approach to learn a globally optimal policy from scratch with high data efficiency, that does not require manual effort and allows for convergence analysis has been considered to date.

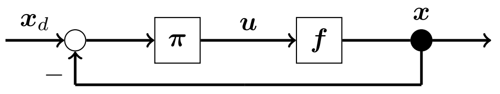


Fig. 1: A closed-loop control structure with controller $\boldsymbol{\pi}$, system dynamics $\boldsymbol{f}$ and target state $\boldsymbol{x}^d$. We model the system dynamics $\boldsymbol{f}$ as a Gaussian process and compute the optimal policy $\boldsymbol{\pi}$ from the systems optimal value function V^*. The proposed algorithm learns $\boldsymbol{f}$ and an approximation of V^* from interactions with the real system.

B. Problem Statement

In this paper, we aim to learn to control a previously unknown dynamics system. We consider discrete-time dynamics

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{\varepsilon} \tag{1}$$

with $\boldsymbol{x}_t, \boldsymbol{x}_{t+1} \in \mathbb{R}^D$, $\boldsymbol{u}_t \in \mathbb{R}^F$, unknown transition dynamics $\boldsymbol{f}$ and i.i.d. Gaussian measurement noise $\boldsymbol{\varepsilon} \in \mathcal{N}(0, \Sigma_{\boldsymbol{\varepsilon}})$ with diagonal noise variance $\Sigma_{\boldsymbol{\varepsilon}}$. Figure 1 shows such a closed-loop control setting. Given a reward function $r\colon \boldsymbol{x}_t \mapsto r(\boldsymbol{x}_t) \in \mathbb{R}$, the goal is to find a policy $\boldsymbol{\pi}\colon \boldsymbol{x} \mapsto \boldsymbol{\pi}(\boldsymbol{x})$ that maximizes the expected reward up to time horizon T, when choosing $\boldsymbol{u}_t := \boldsymbol{\pi}(\boldsymbol{x}_t)$ for $t = 1, \dots, T$. We make no further assumptions (such as, e.g., a parametric form) about $\boldsymbol{\pi}$. The system dynamics $\boldsymbol{f}$ will be learned from observed data and modeled as a Gaussian process (GP). We aim to minimize the system interactions necessary to learn the optimal policy $\boldsymbol{\pi}$.

C. Gaussian Processes as Forward Dynamics Models

In the following, we will briefly recap Gaussian process regression. Given noisy observations $\mathscr{D} = \{(\boldsymbol{z}^i, y^i = f(\boldsymbol{z}^i) + \varepsilon^i) \mid 1 \le i \le N\}$, where $\varepsilon^i \sim \mathcal{N}(0, \sigma_n^2)$, the prior on the values of f is $\mathcal{N}(0, K(Z,Z) + \sigma_n^2 I)$ with the observed inputs $Z = (\boldsymbol{z}^1, \cdots, \boldsymbol{z}^N)^\intercal$. The covariance matrix $K(Z,Z)$ is defined by the choice of covariance function k as $[K(Z,Z)]_{ij} = k(\boldsymbol{z}^i, \boldsymbol{z}^j)$. In this paper, we employ the squared exponential covariance function

$$k(\boldsymbol{z}, \boldsymbol{w}) = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{w})^\intercal \Lambda^{-1} (\boldsymbol{z} - \boldsymbol{w})\right),$$

with signal variance σ_f^2 and squared lengthscales $\Lambda = \operatorname{diag}(l_1^2, \dots, l_{D+F}^2)$ for all input dimensions. This choice is not a requirement of the proposed approach, but has been chosen as squared exponential kernels are generally well suited for smooth dynamics. The proposed approach and all presented analysis can be straightforwardly employed with other kernels.

Given a query point $\boldsymbol{z}_*$, the conditional probability of $\hat{f}(\boldsymbol{z}_*)$ is

$$\begin{aligned}\hat{f}(\boldsymbol{z}_*) \mid \mathscr{D} \sim \mathcal{N}\big(&\boldsymbol{k}(\boldsymbol{z}_*, Z)\boldsymbol{\beta}, \\ &k(\boldsymbol{z}_*, \boldsymbol{z}_*) - \boldsymbol{k}(\boldsymbol{z}_*, Z)(K(Z,Z) + \sigma_n^2 I)^{-1} \boldsymbol{k}(Z, \boldsymbol{z}_*)\big)\end{aligned} \tag{2}$$

with $\boldsymbol{\beta} = (K(Z,Z) + \sigma_n^2 I)^{-1} \boldsymbol{y}$. The hyperparameters, e.g., $\sigma_n^2, \sigma_f^2, \Lambda$ for the squared exponential kernel, are estimated by maximizing the log marginal likelihood of the data [34].

In this paper, we employ a Gaussian process $\boldsymbol{g}$ to model system dynamics. It takes state-action pairs $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{u})^\intercal$ and outputs differences to successor states, i.e., $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{u}_t)$. As these outputs are multivariate, we train conditionally independent GPs for each output dimension. We write $\sigma_{n,m}^2$, $\sigma_{f,m}^2$, Λ_m for the GP hyperparameters in output dimension m and k_m for the corresponding covariance function.

We will employ the GP dynamics model learned from observed data to infer an optimal policy $\boldsymbol{\pi}$. In the following, we will briefly recap value function based approaches.

D. MDPs and Value Iteration

In reinforcement learning, an agent chooses actions in an environment and recieves immediate rewards for the visited states, aiming to maximize the cumulated long-term reward. A common assumption is that the next system state depends only on the current state and the chosen action. This setting can be described as a *Markov Decision Process (MDP)*. Formally, an MDP consists of a set of states S, a set of actions A, a state transition p that maps state-action pairs $(\boldsymbol{s}, \boldsymbol{a})$ to the probability $p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a})$ of the successor state $\boldsymbol{s}'$, the immediate reward function $r\colon S \to \mathbb{R}$ and a discount factor $0 \le \gamma < 1$.

The goal in an MDP is to find a policy $\boldsymbol{\pi}\colon \boldsymbol{s} \mapsto \boldsymbol{a}$, that maximizes the expected long-term reward. For any policy $\boldsymbol{\pi}$, the *Value Function* $V_{\boldsymbol{\pi}}\colon S \to \mathbb{R}$ rates how desirable a state $\boldsymbol{s}$ is in the long run when following the policy $\boldsymbol{\pi}$. More precisely, it is defined as

$$V_{\boldsymbol{\pi}}(\boldsymbol{s}_0) = \sum_{t=0}^{T} \gamma^t \mathbb{E}_{\boldsymbol{s}_t}[r(\boldsymbol{s}_t)] \tag{3}$$

$$= r(\boldsymbol{s}_0) + \gamma \int_S p(\boldsymbol{s}_1 \mid \boldsymbol{s}_0, \boldsymbol{\pi}(\boldsymbol{s}_0)) V_{\boldsymbol{\pi}}(\boldsymbol{s}_1) d\boldsymbol{s}_1 \tag{4}$$

for $T \le \infty$, i.e., the expected long-term reward for choosing the actions $\boldsymbol{a} = \boldsymbol{\pi}(\boldsymbol{s})$ in the considered MDP. The optimal value function V^* is defined as

$$V^*(\boldsymbol{s}) = \max_{\boldsymbol{\pi}} V_{\boldsymbol{\pi}}(\boldsymbol{s}) \tag{5}$$

$$= \max_{\boldsymbol{a} \in A} \int_S p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) \Big(r(\boldsymbol{s}) + \gamma V^*(\boldsymbol{s}')\Big) d\boldsymbol{s}' \tag{6}$$

and given the optimal value function V^*, the optimal policy can be obtained as

$$\boldsymbol{\pi}^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a} \in A} \int_S p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) V^*(\boldsymbol{s}') d\boldsymbol{s}'. \tag{7}$$

The optimal value function can be computed for each $\boldsymbol{s} \in S$ as given in Equation (6), which is commonly known as the *Bellman Equation*. Instead of trying to solve this equation directly, an iterative approach can be taken. Setting $V^0(\boldsymbol{s}) = 0$ the equation

$$V^{t+1}(\boldsymbol{s}) = \max_{\boldsymbol{a} \in A} \int_S p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) \Big(r(\boldsymbol{s}) + \gamma V^t(\boldsymbol{s}')\Big) d\boldsymbol{s}' \tag{8}$$

can be iterated and is guaranteed to converge to V^*. This approach is known as *Value Iteration*. Variations of the value iteration approach include *Policy Iteration*, where Equation (8) is modified to solve for $V_{\boldsymbol{\pi}}$ iteratively and alternated with a policy improvement step that greedily optimizes the policy.

For finite state and action spaces, Equations (6) and (8) can be solved analytically. However, for continuous state and action spaces, these equations are only tractable if the dynamics are linear. For nonlinear dynamics, Equation (8) can be solved approximately by choosing a suitable function approximator for V^t. Under some conditions, approximate value iteration is guaranteed to converge to the true value function [15]. A well studied class of approximate value iteration schemes represents the functions V^t as linear combination of basis functions $\phi_1^t, \dots, \phi_N^t$, i.e., $V^t(\boldsymbol{s}) \approx \sum_{i=1}^N \alpha_i^t \phi_i^t(\boldsymbol{s})$. These approximations have some desirable properties, e.g., the representation is easy to interpret and to handle. The choice of suitable basis functions $\phi_1^t, \dots, \phi_N^t$ significantly affects the approximation quality. Thus, finding suitable basis functions is crucial for approximate value iteration schemes. Typically, the basis functions are problem specific and must be hand-engineered for each task.

III. AVINQ

The AVINQ algorithm learns globally optimal control from interactions with the system. At the same time, we aim to minimize the amount of system interactions required for learning, while still assuming no prior knowledge about the dynamics. In the following, we introduce AVINQ and analyze convergence properties of the proposed algorithm.

A. Algorithm Sketch

The proposed algorithm proceeds in an episodic setting. In the beginning, a starting point is sampled from the initial state distribution $p(\boldsymbol{s}_0)$ and a rollout $(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1), \cdots, (\boldsymbol{s}_{T-1}, \boldsymbol{a}_{T-1}, \boldsymbol{s}_T)$ with randomly chosen actions $\boldsymbol{a}_0, \cdots, \boldsymbol{a}_{T-1}$ is computed. An initial dynamics model $\boldsymbol{g}_0$ is trained on these observations. Based on this dynamics model, an approximation $\widetilde{V}^*$ of the optimal value function V^* is computed employing an approximate value iteration scheme. With this approximate optimal value function, the optimal policy given the dynamics model can be computed for any $\boldsymbol{s}$. A new rollout is computed with the currently optimal policy. This completes the episode. New episodes are computed until the optimal value function (and, thus, the optimal policy) converges. Algorithm 1 shows a high level overview of the proposed approach. Please note that the high level steps of looping model learning, policy optimization and rollout are also followed by [1], however with parametric assumptions on the policy. The approximate value iteration scheme and our approach to learn basis functions for value function approximation are detailed in the following sections.

1) Approximate Value Iteration for GP Dynamics: To infer an optimal value function from the learned GP dynamics model, we propose an approximate value iteration approach. In value iteration, the Bellman Equation (6) is not solved directly, but iterated as

$$V^{t+1}(\boldsymbol{s}) = \max_{\boldsymbol{a}\in A} \int_S p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a})\Big(r(\boldsymbol{s}) + \gamma V^t(\boldsymbol{s}')\Big) d\boldsymbol{s}', \quad (9)$$

while setting $V^0(\boldsymbol{s}) = 0$. We write $\mathcal{A}$ for the Bellman integral operator, that maps V^t to the right hand side of Equation (9), i.e., $\mathcal{A}V^t := V^{t+1}$. This iteration continues until $t = T$, if $T < \infty$ or until the value function converges, if $T = \infty$. In our case, $p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a})$ is given as the prediction of the dynamics GP

Algorithm 1 High level overview of AVINQ

```
Input: time horizon $T \leq \infty$, discount factor $\gamma$, reward function $r$
Output: approximation $\widetilde{V}^*$ of optimal value function $V^*$
  Sample $\boldsymbol{s}_0 \sim p(\boldsymbol{s}_0)$ and random actions $\boldsymbol{a}_0, \cdots, \boldsymbol{a}_{T-1}$
  Observe $\mathscr{D} = \{(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1), \cdots, (\boldsymbol{s}_{T-1}, \boldsymbol{a}_{T-1}, \boldsymbol{s}_T)\}$
  $\widetilde{V}^0 \leftarrow 0$, $e \leftarrow 1$
  while not converged do
    Train GP dynamics model $\boldsymbol{g}_e$ on $\mathscr{D}$
    Compute $\widetilde{V}_e^1, \widetilde{V}_e^2, \cdots \widetilde{V}_e^*$ as in Alg. 2
    $\boldsymbol{\pi}_e(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}\in A} \int p(\boldsymbol{s}'|\,\boldsymbol{s}, \boldsymbol{a}) \widetilde{V}_e^*(\boldsymbol{s}') d\boldsymbol{s}'$
    Sample $\boldsymbol{s}_0 \sim p(\boldsymbol{s}_0)$
    Rollout $(\boldsymbol{s}_0, \boldsymbol{\pi}_e(\boldsymbol{s}_0), \boldsymbol{s}_1), \cdots, (\boldsymbol{s}_{T-1}, \boldsymbol{\pi}_e(\boldsymbol{s}_{T-1}), \boldsymbol{s}_T)$
    $\mathscr{D} \leftarrow \mathscr{D} \cup \{(\boldsymbol{s}_0, \boldsymbol{\pi}_e(\boldsymbol{s}_0), \boldsymbol{s}_1), \cdots, (\boldsymbol{s}_{T-1}, \boldsymbol{\pi}_e(\boldsymbol{s}_{T-1}), \boldsymbol{s}_T)\}$
    $e \leftarrow e + 1$
  end while
```

and is, thus, normally distributed. However, Equation (9) cannot be solved in closed form for all but the linear kernel. Thus, we will solve this equation approximately, choosing a linear representation $\widetilde{V}^t(\boldsymbol{s}) = \sum_{i=1}^{N_t} \alpha^t \phi_i^t(\boldsymbol{s})$ of the value function iterate. In particular, we choose Gaussian basis functions as weighted sums of Gaussians (also called radial basis function networks) have the universal approximation property. With this choice of approximator, the integral in Equation (9) becomes

$$R(\boldsymbol{s}, \boldsymbol{a}) := \int_S p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a})\Big(r(\boldsymbol{s}) + \gamma V^t(\boldsymbol{s}')\Big) d\boldsymbol{s}' =$$
$$r(\boldsymbol{s}) + \gamma \sum_{i=1}^N \alpha_i^t \int_S p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) \phi_i^t(\boldsymbol{s}') d\boldsymbol{s}'. \quad (10)$$

The integral of the product of Gaussians can be computed as

$$\int \mathcal{N}(\boldsymbol{s} \mid \boldsymbol{x}, X) \mathcal{N}(\boldsymbol{s} \mid \boldsymbol{y}, Y) d\boldsymbol{s} =$$
$$(2\pi)^{-\frac{D}{2}} \det(Z)^{-\frac{1}{2}} \exp(-\frac{1}{2}\boldsymbol{z}^\intercal Z^{-1} \boldsymbol{z}) \quad (11)$$

with $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{y}$ and $Z = X + Y$. As $p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a})$ and $\phi_i^t(\boldsymbol{s}')$ are both Gaussian, we can apply this formula and as a result obtain an analytical solution to Equation (10). However, the maximum of $R(\boldsymbol{s}, \boldsymbol{a})$ with respect to $\boldsymbol{a}$ remains analytically intractable. For any fixed $\boldsymbol{s}$, we can maximize $R(\boldsymbol{s}, \boldsymbol{a})$ with respect to $\boldsymbol{a}$ via gradient ascent. Thus, we can solve the right hand side of the Bellman Equation (9) for a finite number of $\boldsymbol{s}$, if we replace V^t by $\widetilde{V}^t$. To continue our approximate value iteration scheme, we need to find an approximation $\widetilde{V}^{t+1}$ of V^{t+1} of the form $\widetilde{V}^{t+1} = \sum_{i=1}^{N_{t+1}} \alpha^{t+1} \phi_i^{t+1}$. We will introduce an algorithm to find the basis functions ϕ_i^{t+1} in the following section. For now, lets assume that $\phi_1^{t+1}, \cdots \phi_{N_t}^{t+1}$ are given. It remains to find the weights $\alpha_1^{t+1}, \cdots, \alpha_{N_{t+1}}^{t+1}$ that minimize the distance of our approximation $\widetilde{V}^{t+1}$ to the true result of the Bellman equation $V^{t+1} \approx \mathcal{A}\widetilde{V}^t$. We measure the distance of two scalar functions in L_2-norm, i.e., $\|f_1 - f_2\|_{L_2}^2 = \int (f_1(\boldsymbol{s}) - f_2(\boldsymbol{s}))^2 d\boldsymbol{s}$. In this case, the weights $\alpha_1^{t+1}, \cdots, \alpha_{N_{t+1}}^{t+1}$ of the best approximation of V^{t+1} in the linear subspace of L_2 spanned by the basis functions $\phi_1^{t+1}, \cdots, \phi_{N_{t+1}}^{t+1}$ can be obtained straightforwardly by projection. More precisely, the weight vector $\boldsymbol{\alpha}^{t+1}$ is the solution to the linear equation system

$$M\boldsymbol{\alpha}^{t+1} = \boldsymbol{b}^{t+1} \tag{12}$$

with the mass matrix $M_{i,j} = \langle \phi_i^{t+1}, \phi_j^{t+1} \rangle_{L_2} = \int \phi_i^{t+1}(\boldsymbol{s})\phi_j^{t+1}(\boldsymbol{s})d\boldsymbol{s}$ and the projections $b_i^{t+1} = \langle \phi_i^{t+1}, V^{t+1} \rangle_{L_2} = \int \phi_i^{t+1}(\boldsymbol{s})V^{t+1}(\boldsymbol{s})d\boldsymbol{s}$. The entries of M can be computed employing Equation (11). Unfortunately, the entries of $\boldsymbol{b}^{t+1}$ are intractable, as they would require a closed form expression for V^{t+1}, which we try to approximate. However, as noted above we can compute the value of V^{t+1} at any point $\boldsymbol{s}$. We propose to approximate the integrals in $\boldsymbol{b}^{t+1}$ with numerical quadrature. Numerical quadrature estimates the value of an integral by a weighted, finite sum of function evaluations at quadrature nodes. The nodes and weights are given by the quadrature rule. With this approximation, we can estimate $\boldsymbol{b}^{t+1}$ with a finite number of values of V^{t+1}. Solving Equation (12), we obtain the weights for our approximation $\widetilde{V}^{t+1}$ of V^{t+1}. We write Π_B for the projection operator onto the linear subspace of L_2 spanned by the set of basis functions B, i.e., $\widetilde{V}^t = \Pi_B[V^t]$. Algorithm 2 summarizes the proposed projection approach.

If suitable basis functions for the considered problem are known a priori, e.g., given by an expert, they can be directly employed with the algorithm described above. If no suitable basis functions are given, which we believe is the most frequent case, a set of suitable basis functions can be learned iteratively as we will describe in the following section.

```
**Algorithm 2** Value Function Approximation with NQ
**Input:** GP dynamics $\boldsymbol{g}_e$, $\widetilde{V}^t = \sum_{i=1}^{N_t} \alpha_i^t \phi_i^t$, optional: basis
    function set $B = \{\phi_1^{t+1}, \cdots, \phi_{N_{t+1}}^{t+1}\}$ given by expert
**Output:** $\widetilde{V}^{t+1} = \sum_{i=1}^{N_{t+1}} \alpha_i^{t+1} \phi_i^{t+1}$
  **if** $B$ not given, **then**
    Call Alg. 3 with function handle $\mathcal{A}\widetilde{V}^t$
  **end if**
  $M_{i,j} \leftarrow \langle \phi_i^{t+1}, \phi_j^{t+1} \rangle_{L_2}$ for $i, j = 1, \cdots, N_{t+1}$
  Get nodes $\boldsymbol{\xi}_1, \cdots, \boldsymbol{\xi}_K$ and weights $w_1, \cdots, w_K$ from NQ
  $b_i^{t+1} \leftarrow \sum_{k=1}^{K} w_k \phi_i^{t+1}(\boldsymbol{\xi}_k)\mathcal{A}\widetilde{V}^t(\boldsymbol{\xi}_k)$ for $i = 1, \cdots, N_{t+1}$
  $\boldsymbol{\alpha}^{t+1} \leftarrow M^{-1}\boldsymbol{b}^{t+1}$
```

2) Learning Features for Value Function Approximation: A set of suitable basis functions for value function approximation is crucial for the success of any approximate value iteration scheme. In most cases, the choice of basis functions is nontrivial and involves manual effort. Subsequently, we introduce an algorithm to find suitable basis functions online during value iteration.

The proposed approximate value iteration scheme minimizes the L_2-error of the approximate solution. However, convergence guarantees for approximate value iteration can typically be given, if upper bounds for the L_∞-error are known. Thus, we aim to find good basis functions to decrease the L_2-error of the approximation while being able to reach any defined L_∞-error bound. More precisely, the goal is for any value function iterate $V^t(\boldsymbol{s})$ and $\varepsilon > 0$ to find a set $B = \{\phi_1^t, \cdots, \phi_{N_t}^t\}$ of Gaussian features, such that its projection $\widetilde{V}^t$ onto the linear space spanned by B fulfills

$$\|\widetilde{V}^t - V^t\|_{L_\infty} = \max_{\boldsymbol{s}\in S} |\widetilde{V}^t(\boldsymbol{s}) - V^t(\boldsymbol{s})| < \varepsilon. \tag{13}$$

We propose an iterative approach, that adds basis function after basis function until the criterion (13) is met. We start with $B = \{\}$ and define the projection error $\rho_B(\boldsymbol{s}) = |\Pi_B[V^t(\boldsymbol{s})] - V^t(\boldsymbol{s})|$. Note that we can evaluate ρ_B at any point $\boldsymbol{s} \in S$, although we cannot express V^t in closed form. We follow a greedy approach with respect to L_∞-error. Given the set $B = \{\phi_1^t, \cdots, \phi_l^t\}$ of previously added basis functions, we find a local maximum $\boldsymbol{s}_*$ of ρ_B via gradient ascent. As this optimization is non-convex, we cannot expect to find the global maximum of ρ_B. However, we perform gradient ascent from multiple starting points to make sure that we estimate the scale of ρ_B properly. If $\rho_B(\boldsymbol{s}_*) > \varepsilon$, we add another Gaussian basis function ϕ_{l+1}^t with mean $\boldsymbol{s}_*$ and covariance Σ_*. To estimate the covariance matrix Σ_*, we compute the Hessian of ρ_B at $\boldsymbol{s}_*$. It is well-known that the Hessian of the log of a normal density equals the negative inverse of its covariance matrix, i.e.,

$$\begin{aligned} -\log(\phi_{l+1}^t(\boldsymbol{s})) &= \frac{D}{2}\log(2\pi) + \frac{1}{2}\log(\det(\Sigma_*)) \\ &\quad + \frac{1}{2}(\boldsymbol{s} - \boldsymbol{s}_*)^\intercal \Sigma^{-1}(\boldsymbol{s} - \boldsymbol{s}_*) \\ -\frac{\partial^2 \log(\phi_{l+1}^t(\boldsymbol{s}))}{\partial s_i \partial s_j} &= \left(\Sigma_*^{-1}\right)_{i,j} \end{aligned} \tag{14}$$

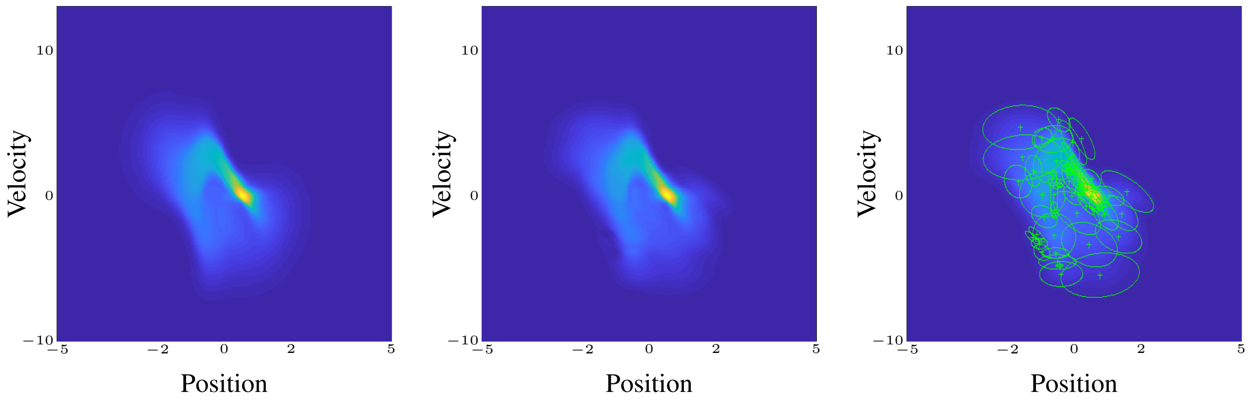
and following the chain rule

$$\begin{aligned} -\frac{\partial^2 \log(\phi_{l+1}^t)}{\partial s_i \partial s_j} &= \left(\phi_{l+1}^t(\boldsymbol{s}_*)\right)^{-2} \frac{\partial \phi_{l+1}^t}{\partial \boldsymbol{s}_i} \frac{\partial \phi_{l+1}^t}{\partial \boldsymbol{s}_j} \\ &\quad - (\phi_{l+1}^t(\boldsymbol{s}_*))^{-1} \frac{\partial^2 \phi_{l+1}^t(\boldsymbol{s})}{\partial s_i \partial s_j}. \end{aligned} \tag{15}$$

Thus, setting the Hessian of ϕ_{l+1}^t equal to the Hessian of ρ_B at $\boldsymbol{s}_*$, we can compute the covariance matrix Σ_* with Equation (14) and (15). Please note that this estimate is very accurate, if ρ_B is locally close to a Gaussian shape.

We add this new basis function to B and update ρ_B. Subsequently, the search for the maximum of ρ_B begins as above. This procedure is repeated until no more states $\boldsymbol{s}_*$ with $\rho_B(\boldsymbol{s}_*) > \varepsilon$ are found. Please note that this algorithm decreases the L_2-error of the approximation in every step. As the solution of an integral equation with a smooth kernel, the value function is continuous. Thus, the L_∞-error of our approximation also decreases with the L_2 error as we increase the number of basis functions (though not necessarily monotonously). Finally, it is important to note that this approach guarantees that the L_∞-error is smaller than ε only under the assumption that the global maximum of ρ_B is found. In our experiments, we found ρ_B well-behaved and did not encounter problems to find its global maximum.

Algorithm 3 shows the proposed approach for finding basis functions to approximate the value function iterates. Figure 2 illustrates our approach to find basis functions. In the following, we will analyze the convergence of AVINQ .



(a) Exact Solution of Bellman Eq. (10) $\mathcal{A}\widetilde{V}^{t-1}$ (b) Our Approximation $\widetilde{V}^t$ (c) $\widetilde{V}^t$ with Learned Basis Functions

Fig. 2: Finding suitable basis function for value function approximation with Alg. 3. Subplot (a) shows the exact value function iterate $\mathcal{A}\widetilde{V}^{t-1}$ computed at each pixel. Plot (b) shows the projection onto the space spanned by the obtained basis functions, i.e., our approximation $\widetilde{V}^t$. In subplot (c), the obtained basis functions are indicated by 1σ-ellipsoids.

Algorithm 3 Find Basis Functions to Approximate V^{t+1}

```
Input: function handle $V^{t+1}$, tolerance $\varepsilon$
Output: basis function set $B$, s.t. $\|\Pi_B[V^t] - V^t\|_{L_\infty} < \varepsilon$
  $B \leftarrow \{\}$, $l \leftarrow 0$, $\widetilde{V}^t = 0$
  Set function handle $\rho_B(\boldsymbol{s}) \leftarrow |\widetilde{V}^t(\boldsymbol{s}) - V^t(\boldsymbol{s})|$
  $\boldsymbol{s}_* \leftarrow \max_{\boldsymbol{s}\in S} \rho_B(\boldsymbol{s})$
  while $\|\widetilde{V}^t - V^t\|_{L_\infty} \approx \rho_B(\boldsymbol{s}_*) > \varepsilon$ do
    Compute $\Sigma_*$ with Eq. (15)
    $\phi^t_{l+1} \leftarrow \mathcal{N}(\boldsymbol{s}_*, \Sigma_*)$
    $B \leftarrow B \cup \{\phi^t_{l+1}\}$
    $\widetilde{V}^t \leftarrow \Pi_B[V^t]$
    Set function handle $\rho_B \leftarrow |\widetilde{V}^t(\boldsymbol{s}) - V^t(\boldsymbol{s})|$
    $\boldsymbol{s}_* \leftarrow \max_{\boldsymbol{s}\in S} \rho_B(\boldsymbol{s})$
  end while
```

B. Convergence Analysis

Much research has been conducted towards convergence guarantees for value iteration and approximate value iteration approaches. While guarantees can be given straightforwardly for value iteration approaches on finite state and action spaces [16], this task is a lot more challenging for approximate value iteration approaches on infinite state and action spaces. Guarantees that approximate value iteration will converge to a globally optimal policy can be given assuming the dynamics model is correct, if the value function approximation fulfills certain criteria. Subsequently, we show that such a guarantee can be given for AVINQ.

Theorem 1. *Let a GP dynamics model $\boldsymbol{g}$ be given and assume that Algorithm 3 always succeeds finding the global maximum of ρ_B. Then, it holds*

$$\limsup_{t\to\infty} \|V^* - \widetilde{V}^t\|_{L_\infty} \leq \frac{2\gamma}{(1-\gamma)^2}\epsilon, \tag{16}$$

for the functions $\widetilde{V}^t$ computed by iterating Algorithm 2 with the uniform approximation error bound $\epsilon = \varepsilon + C\varepsilon_{NQ}$.

Proof. In [15], it is shown that for the policy $\widetilde{\boldsymbol{\pi}}^t$ greedy with respect to $\widetilde{V}^t$ obtained with approximate value iteration, the performance loss is bounded by

$$\|V^* - \widetilde{V}^t\| \leq \frac{2\gamma}{(1-\gamma)^2} \max_{k=0,\cdots,t} \|\Pi_B[\mathcal{A}V^k] - V^{k+1}\|_{L_\infty} + \frac{2\gamma^{t+1}}{1-\gamma}\|V^0 - V^*\|_{L_\infty} \tag{17}$$

with the projection error $\|\Pi_B[\mathcal{A}V^k] - V^{k+1}\|$ and the initial guess error $\|V^0 - V^*\|$. As $t \to \infty$, the second term vanishes and the performance loss is governed by the projection error. Thus, we are interested in a uniform bound of the projection error introduced by AVINQ . Assuming that Algorithm 3 succeeds in finding the global optimum of ρ_B in all iteration steps, the output of this algorithm is a basis function set B such that the (exact!) L_2-projection of V^{t+1} onto B deviates from V^{t+1} by at most ε at any point. However, the projection employed in AVINQ is not exact, as the right hand side $\boldsymbol{b}^{t+1}$ of the projection equation system is approximated by numerical quadrature. Fortunately, upper bounds for the quadrature error are available, e.g., [35]. Let ε_{NQ} denote the upper bound for the quadrature error, i.e., $\varepsilon_{NQ} := \max_i |\langle \phi_i^{t+1}, \mathcal{A}V^t\rangle - b_i^{t+1}|$. Then, the obtained weight vector $\boldsymbol{\alpha}^{t+1}$ differs from the exact solution of Equation (12) by at most $\|M^{-1}\varepsilon_{NQ}\|_\infty$. It remains to note that the smallest eigenvalue of the matrices M for $t = 1, 2, \ldots$ is lower bounded as V^* is differentiable, which concludes the proof.

In an episodic setting, however, there is no guarantee that the dynamics model will converge globally to the true dynamics. If the employed dynamics model incorporates uncertainty, exploration is encouraged as in regions of high uncertainty the expected reward averages over many states. Thus, uncertain states will be preferred over states with known low value. However, without explicit exploration, no firm convergence guarantees can be given.

Another interesting question is how the computational complexity of the different components of AVINQ depend on the task to be solved. Evaluation of the Bellman Equation at a

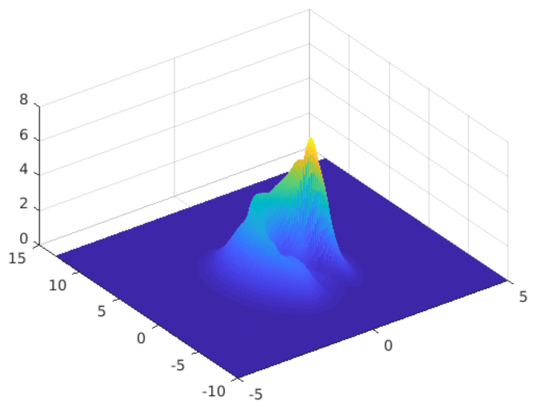


Fig. 3: Obtained value function for the mountain car task after five episodes.

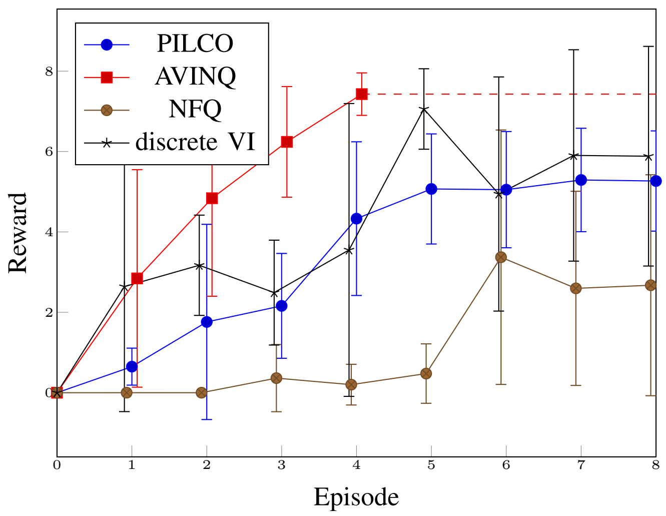


Fig. 4: Average reward as function of system interaction time for the mountain car task. One episode equals 3 seconds of system interaction time.

fixed point involves maximization with respect to the action and, thus, scales cubically with the dimension F of the action space if a Hessian based method is employed. Furthermore, it depends linearly on the number of optimization steps X_a, i.e., the overall complexity of one evaluation of $\max_a R(s,a)$ is $O(F^3 X_a)$. Maximization of ρ_B is of complexity $O(D^3 X_s)$ with the state space dimension D and the number of optimizer steps X_s. Thus, the complexity of finding one new basis function is $O(D^3 F^3 X_s X_a)$. To add the basis function, a projection of the value function onto the new basis function is computed via numerical quadrature in D dimensions. Naive generalizations of 1D quadrature to higher dimensions suffer from the curse of dimensionality. However, sparse grid approaches make numerical quadrature feasible for up to 20 dimensions [36]. Finally, the number of required basis functions to meet a specified error tolerance ε highly depends on the smoothness of the value function. More precisely, it is linked to the decrease of the Fourier coefficients [37].

IV. Evaluation

We evaluate AVINQ on the mountain car benchmark and compare the results with other approaches.

In the mountain-car domain (see, e.g., [38], [16]), a car starts at some point in a valley landscape and has to reach a certain point on the hill to the right side of the valley and stay there. However, the car's engine is not powerful enough to reach the goal directly from all starting positions. From the points in the valley, the car has to first drive in the opposite direction and gain momentum to reach the goal. The state space has two dimensions: position and velocity of the car, the magnitude of the control signal is limited to $u_{\max} = 4$.

We learn a globally optimal policy for the mountain car benchmark with AVINQ. We choose $\gamma = 0.95$ and learn the basis functions with Algorithm 3, such that the approximation error is smaller than $\varepsilon = 0.07$. This is reached with up to 60 Gaussian basis functions in all iteration steps and episodes.

For the projection onto the linear space spanned by the basis functions, we employ adaptive Gauss-Kronrod quadrature for the integrals $\int \phi_i^t(\boldsymbol{s}) V^t(\boldsymbol{s}) d\boldsymbol{s}$. This adaptive scheme subdivides the integration area and estimates the quadrature error on each subregion by comparison of a higher and a lower order quadrature rule. As Gauss-Kronrod quadrature is nested, i.e., the higher order rules contain all nodes of lower order rules, this scheme is very efficient. To further increase the precision and efficiency of the quadrature, we perform a coordinate transformation, such that the basis function ϕ_i^t becomes axis aligned in the new coordinate system. This transformation concentrates the mass close to the center of the integration area making sure that the adaptive quadrature scheme does not stop prematurely. Furthermore, we normalize the height of the integrand $\phi_i^t(\boldsymbol{s}) V^t(\boldsymbol{s})$ for numerical stability and to avoid cancellation effects. It is also well known that the mass matrix M tends to be badly conditioned, if the basis functions are not normalized with respect to L_2-norm. After normalizing the basis functions before solving Equation (12) we found the solution to be highly robust. Please note that these modifications are implementation details, introduced to increase robustness of the computations involved. They do not change the theoretical properties of AVINQ and are provided for the sake of completeness.

Figure 3 shows the resulting value function after five episodes. As can be seen, the obtained value function captures the dynamics of the mountain car very well. In Figure 4 the obtained discounted reward during learning is compared to PILCO and NFQ and discretized Value Iteration. The reported results for PILCO were obtained with the code published by the authors. As no official code is available for NFQ, we reimplemented this approach for the comparison. As a baseline, we implemented a discretized Value Iteration approach based on a GP dynamics model. For this approach, the value function was updated for the discrete states only and the update step employed transition probabilites predicted by the GP dynamics model. We chose a discretization with $50 \times 50 \times 30$ states. All results were averaged over multiple runs with different initializations of the random number generator.

AVINQ outperforms all tested algorithms. Please note also that PILCO does not learn a global policy, while the other approaches do.

V. CONCLUSION

In this paper we introduced AVINQ, an approach to learn globally optimal control on continuous state and action domains from scratch, while maintaining high data efficiency. AVINQ approximates the value function by projection onto a linear space of learned features. The forward dynamics are modeled as GP and learned in an episodic setting. Numerical quadrature is employed to enable projection of the value function. AVINQ benefits from Bayesian averaging over all plausible models by incorporating the uncertainty of the GP model. Furthermore, it requires no manual effort or problem specific knowledge to obtain a task solution. We also provided theoretical analysis of AVINQ, showing that convergence of the value iteration can be guaranteed under some assumptions, but convergence of model learning cannot be guaranteed without presupposing problem specific knowledge.

REFERENCES

[1] M. Deisenroth, D. Fox, and C. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 408–423, 2015.

[2] Y. Pan and E. Theodorou, "Probabilistic differential dynamic programming," in *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 1907–1915.

[3] E. Klenske, M. Zeilinger, B. Schölkopf, and P. Hennig, "Nonparametric dynamics estimation for time periodic systems," in *51st Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2013, pp. 486–493.

[4] J. Maciejowski and X. Yang, "Fault tolerant control using gaussian processes and model predictive control," in *Conference on Control and Fault-Tolerant Systems (SysTol 2013)*. IEEE, 2013, pp. 1–12.

[5] D. Nguyen-Tuong and J. Peters, "Model learning in robotics: a survey," *Cognitive Processing*, no. 4, 2011.

[6] Y. Engel, P. Szabo, and D. Volkinshtein, "Learning to control an octopus arm with gaussian process temporal difference methods," in *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. MIT Press, 2006, pp. 347–354.

[7] J. Kocijan, R. Murray-Smith, C. Rasmussen, and A. Girard, "Gaussian process model based predictive control," in *Proceedings of the American Control Conference, (ACC 2004).*, vol. 3. IEEE, 2004, pp. 2214–2219.

[8] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in *Proceedings of 1997 IEEE International Conference on Robotics and Automation*. IEEE, 1997, pp. 3557–3564.

[9] L. Kuvayev, "Model-based reinforcement learning with an approximate, learned model," in *in Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems*, 1997.

[10] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Machine Learning*, vol. 13, no. 1, pp. 103–130, 1993.

[11] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *In Proceedings of the Seventh International Conference on Machine Learning (ICML 1990)*. Morgan Kaufmann, 1990, pp. 216–224.

[12] J. Schneider, "Exploiting model uncertainty estimates for safe dynamic control learning," in *Advances in Neural Information Processing Systems 9 (NIPS 1996)*.

[13] T. Beckers and S. Hirche, "Stability of gaussian process state space models," in *Proceedings of the European Control Conference*. IEEE, 2016, pp. 2275–2281.

[14] J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, A. Romer, H. Schmidt, and J. Peters, "Stability of controllers for gaussian process forward models," in *Proceedings of the 33nd International Conference on Machine Learning (ICML 2016)*, 2016, pp. 545–554.

[15] D. Bertsekas and J. Tsitsiklis, *Neuro-dynamic Programming*, ser. Anthropological Field Studies. Athena Scientific, 1996.

[16] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[17] D. Bertsekas, *Dynamic Programming and Optimal Control*, ser. Athena Scientific optimization and computation series. Athena Scientific, 2005, no. vol. 1.

[18] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. John Wiley & Sons, 2005.

[19] K. Zhou and J. Doyle, *Essentials of Robust Control*, ser. Prentice Hall Modular Series for Eng. Prentice Hall, 1998.

[20] K. Narendra and A. Annaswamy, *Stable Adaptive Systems*, ser. Dover Books on Electrical Engineering. Dover Publications, 2012.

[21] G. Tao, *Adaptive Control Design and Analysis*. John Wiley & Sons, Inc., 2003.

[22] A. G. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann, "Data-efficient generalization of robot skills with contextual policy search." in *AAAI*, 2013.

[23] C. Rasmussen and M. Kuss, "Gaussian processes in reinforcement learning," in *Advances in Neural Information Processing Systems 16 (NIPS 2003)*. MIT Press, 2004, pp. 751–759.

[24] A. Rottmann and W. Burgard, "Adaptive autonomous control using online value iteration with gaussian processes," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2106–2111.

[25] M. Deisenroth, C. Rasmussen, and J. Peters, "Gaussian process dynamic programming," *Neurocomputing*, vol. 72, no. 7-9, pp. 1508–1524, Mar. 2009.

[26] B. Bischoff, D. Nguyen-Tuong, H. Markert, and A. Knoll, "Learning control under uncertainty: A probabilistic value-iteration approach," in *21st European Symposium on Artificial Neural Networks, ESANN 2013*, 2013.

[27] M. Riedmiller, "Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method," in *Proceedings of the 16th European Conference on Machine Learning (ECML)*. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 317–328.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[29] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of machine learning research*, vol. 4, no. Dec, pp. 1107–1149, 2003.

[30] P. W. Keller, S. Mannor, and D. Precup, "Automatic basis function construction for approximate dynamic programming and reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 449–456.

[31] C. Painter-wakefield and R. Parr, "Greedy algorithms for sparse reinforcement learning," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, J. Langford and J. Pineau, Eds. New York, NY, USA: ACM, 2012, pp. 1391–1398.

[32] A.-m. Farahmand and D. Precup, "Value pursuit iteration," in *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS 2012)*. Curran Associates Inc., 2012, pp. 1340–1348.

[33] A. Girard, C. E. Rasmussen, J. Q. Candela, and R. Murray-Smith, "Gaussian process priors with uncertain inputs - application to multiple-step ahead time series forecasting," in *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, 2002, pp. 529–536.

[34] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[35] M. Masjed-Jamei, "New error bounds for gauss-legendre quadrature rules." *Filomat*, vol. 28, no. 6, pp. 1281–1293, 2014.

[36] F. Heiss and V. Winschel, "Likelihood approximation by numerical integration on sparse grids," *Journal of Econometrics*, vol. 144, no. 1, pp. 62 – 80, 2008.

[37] B. Fornberg and N. Flyer, "Accuracy of radial basis function interpolation and derivative approximations on 1-d infinite grids," *Advances in Computational Mathematics*, vol. 23, no. 1, pp. 5–20, 2005.

[38] A. W. Moore and C. G. Atkeson, "The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces," *Machine Learning*, vol. 21, no. 3, pp. 199–233, 1995.