IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 14 May 2023, accepted 29 May 2023, date of publication 5 June 2023, date of current version 21 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3282842

RESEARCH ARTICLE

Variance-Based Exploration for Learning Model Predictive Control

KATRINE SEEL[1,2], ALBERTO BEMPORAD[3], (Fellow, IEEE), SÉBASTIEN GROS[1], AND JAN TOMMY GRAVDAHL[1], (Senior Member, IEEE)

[1]Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), 7034 Trondheim, Norway
[2]Department of Mathematics and Cybernetics, SINTEF Digital, 0373 Oslo, Norway
[3]IMT School for Advanced Studies, 55100 Lucca, Italy

Corresponding author: Katrine Seel (katrine.seel@ntnu.no)

This work was supported in part by the Industry Partners Borregaard, Elkem, Yara, Hydro, and the Research Council of Norway through the Project "Toward Autonomy in Process Industries" (TAPI) under Project 294544; and in part by the Project "Safe Reinforcement Learning using Model Predictive Control" (SARLEM) under Project 300172.

ABSTRACT The combination of model predictive control (MPC) and learning methods has been gaining increasing attention as a tool to control systems that may be difficult to model. Using MPC as a function approximator in reinforcement learning (RL) is one approach to reduce the reliance on accurate models. RL is dependent on exploration to learn, and currently, simple heuristics based on random perturbations are most common. This paper considers variance-based exploration in RL geared towards using MPC as function approximator. We propose to use a non-probabilistic measure of uncertainty of the value function approximator in value-based RL methods. Uncertainty is measured by a variance estimate based on inverse distance weighting (IDW). The IDW framework is computationally cheap to evaluate and therefore well-suited in an online setting, using already sampled state transitions and rewards. The gradient of the variance estimate is then used to perturb the policy parameters in a direction where the variance of the value function estimate is increasing. The proposed method is verified on two simulation examples, considering both linear and nonlinear system dynamics, and compared to standard exploration methods using random perturbations.

INDEX TERMS Inverse distance weighting, model predictive control, Q-learning, reinforcement learning.

I. INTRODUCTION

Reinforcement learning (RL) is a powerful tool for tackling Markov decision processes (MDPs). Rather than relying on a model of the state transition probabilities, samples of state transitions and associated rewards can be used to improve the performance of a control policy. RL has drawn increasing attention due to its accomplishments in robotics and games, see e.g. [1] and [2]. However, as neural networks (NNs) typically are used as function approximators, guarantees regarding the closed-loop behavior of the policy are difficult to provide.

Model predictive control (MPC) has established itself as the primary control method for the systematic handling of system constraints. The MPC scheme relies on a sufficiently accurate model of the system, to optimize the system performance with respect to a given objective while respecting constraints. Different combinations of MPC and learning have been proposed to deal with systems that may be hard to model that also inherit closed-loop behaviors that are easier to analyze, see e.g. [3], [4], and [5]. In [6], the authors suggest using parameterized MPC schemes as a function approximator of the policy and value function in RL. Parameterizing the MPC problem allows RL to improve the policy as data is acquired while maintaining an MPC structure, which offers rich tools to analyze the resulting closed-loop behavior.

RL requires that the actions applied to the real system undergo some exploration. If the same, deterministic policy is always applied to the system, it is not possible to discover alternative actions that may improve closed-loop

The associate editor coordinating the review of this manuscript and approving it for publication was Guillermo Valencia-Palomo.

performance. One way to quantify an effective exploration is in terms of regret. The notion of regret in RL is defined as the loss in reward for choosing a suboptimal over an optimal action. An effective exploration strategy can then be defined as minimizing the cumulative sum of regrets. However, we cannot directly obtain the regret as the optimal action is not known. Hence, the concept of regret in itself cannot be used to perform effective exploration.

Currently, the most commonly used methods to explore are simple heuristics. For discrete action spaces, methods such as ϵ-greedy [7] or Boltzmann exploration [8] are used. For continuous action spaces, stochastic policies for exploration are generated e.g. by adding Gaussian noise to a deterministic policy [9]. In the case of using stochastic policy gradient methods, the distribution of the stochastic policy itself is parameterized and adjusted by RL. Exploration is then ensured by sampling from the resulting distribution that describes the stochastic policy [7].

A collective term for the aforementioned exploration strategies is *dithering* strategies. Because the perturbation from one time step to the next is not coordinated, the exploration is not temporally extended or what we refer to as *deep*. For problems that require consistent exploration over several time steps to realize improved closed-loop performance, dithering strategies may in fact prevent efficient exploration. The most straightforward method for ensuring deep exploration, is random perturbations in parameter space, as suggested by the authors in [10]. A random perturbation in the parameters is introduced at the beginning of an episode, and fixed throughout that episode, such that a temporally coordinated sequence of actions is generated. However, the potential benefit of using random noise in parameter space rather than in action space is generally not obvious and needs to be evaluated on a case-by-case basis. Moreover, when using random parameter noise for exploration in large parameter spaces, we are at risk of adding a lot of disturbances that yield little effect on the resulting policy.

Although the aforementioned heuristics perform well for many tasks, they are all undirected and therefore may take exponentially long to learn the optimal policy [11]. To learn efficiently, the exploration method should prioritize potentially informative states and actions. To do this, exploration should be done with regard to a notion of uncertainty.

Directed exploration is well understood for the multi-armed bandit problem, which corresponds to a one-step stateless MDP problem. One strategy is ''optimism in the face of uncertainty'', which corresponds to preferring actions with uncertain values. This strategy has led to e.g. the upper confidence bound (UCB) algorithm. The UCB algorithm acts greedily w.r.t. to the action-value function and an exploration bonus based on a confidence interval of the reward, see e.g. [12]. The UCB algorithm in [12] has been extended to RL in different forms, but the resulting algorithms have been considered mostly as theoretical results due to the computational complexity.

Thompson sampling (TS) is a related strategy developed for the bandit problem. TS sampling is based on a Bayesian approach to maintaining a posterior distribution over models. We sample from the posterior distribution and then select the action that optimizes the sampled model [13]. Extending TS to RL would involve maintaining a distribution of MDPs, which in general is difficult. Even updating a Bayesian model of the value function for an MDP will for most realistic problem sizes be computationally intractable. For bandit problems, both UCB and TS achieve a sublinear total regret. In comparison, ϵ-greedy has a linear total regret, which is the same as with no exploration at all.

As a means to reduce the computational burden, yet inspired by TS, the authors in [14] introduced the concept of randomized value functions. The use of randomized value functions aims to approximate samples from the posterior distribution of the value function. However, the method is developed only for linear parameterizations of the value function. An extension was made to nonlinear parameterizations, more specifically to NNs, in [15], where bootstrapped deep Q-function NNs (DQN) were used to approximate the posterior distribution of the Q-function.

The concept of randomized value functions has also motivated the *NoisyNets* as proposed in [16]. Rather than training an NN with K outputs or heads to build an approximate posterior distribution of Q as in [15], the authors in [16] inject noise in the NN parameters and use RL to tune the intensity i.e. the variance of the noise distributions. A new sample of the Q-function is then obtained by sampling parameter noise from the tuned noise distributions, and exploration is done by acting greedily with respect to that Q-function.

Bootstrapped DQNs have also been used to develop a practical UCB approach that applies to MDPs. Namely, the bootstrapped DQN was used to obtain an empirical estimate of the mean and standard deviation of the Q-function posterior distribution [17]. This was in turn used to formulate a UCB, and the action was selected by maximizing the UCB. Along the same lines, the DQN framework was used by the authors in [18] in order to obtain confidence intervals to formulate a surrogate of the regret, which in turn was used to guide exploration. The use of randomized value functions constitutes an important step towards more effective exploration strategies in RL, although it for nonlinear value function parameterizations only applies to discrete action spaces.

A. CONTRIBUTION

The goal of this work is to develop a directed and deep exploration strategy for continuous action spaces, that is suitable for problems where we wish to use MPC as a function approximator in RL. For this purpose, we will adopt the principle of ''optimism in the face of uncertainty''. To the best of the authors' knowledge, few studies exist on directed exploration strategies in continuous action spaces. One important exception is the work in [19], where K value function approximators are trained independently, and the

agent is encouraged to explore states where the value function approximators show the largest disagreement. Although the exploration strategy resembles ours, it is based on knowing the true model of the MDP, which is not a requirement in our case. We present an uncertainty-based exploration method not limited to, but particularly suited for MPC, and make the following contributions:

- we introduce the use of inverse distance weighting (IDW) to estimate the variance of the MPC function approximator at a low computational cost;
- we formulate a novel approach to uncertainty-based exploration in parameter space via the IDW variance estimate;
- we compare the proposed method with random (Gaussian) perturbations in both action and parameter space.

The proposed method is verified on two simulation examples, considering both linear and nonlinear dynamics, for which variance-based exploration performs better in terms of significantly improving the cumulative rewards during learning.

The paper is structured as follows. Section II provides background information on the problem statement. This is followed by a short introduction to exploration in parameter space and its application to MPC-based RL. Section IV details the IDW framework and how this can be used to obtain a variance estimate of the selected value function approximator. The use of the IDW variance in exploration is then detailed in Section V, followed by two simulation examples in Section VI. Finally, conclusions are given in Section VII.

II. BACKGROUND

The RL framework applies to problems that can be cast as an MDP. MDPs are described by a state $\boldsymbol{s} \in \mathcal{S} \subseteq \mathbb{R}^n$ and the input $\boldsymbol{a} \in \mathcal{A} \subseteq \mathbb{R}^m$. We will consider a special case of MDPs, described by the deterministic dynamics

$$\boldsymbol{s}_{k+1} = \boldsymbol{f}(\boldsymbol{s}_k, \boldsymbol{a}_k), \tag{1}$$

where $\boldsymbol{s}_{k+1}$ denotes the next state and k denotes the physical time of the system. The function $\boldsymbol{f} : \mathbb{R}^{n+m} \to \mathbb{R}^n$ denotes the (possibly nonlinear) state transition. The system is subject to the following constraints

$$\boldsymbol{h}(\boldsymbol{s}_k, \boldsymbol{a}_k) \leq 0, \tag{2}$$

where $\boldsymbol{h} : \mathbb{R}^{n+m} \to \mathbb{R}^d$ describes a mixed input-state constraint. We will assume in the following that a stage cost $L(\boldsymbol{s}_k, \boldsymbol{a}_k)$, analogous to negative reward, is provided. Our goal is to find a deterministic policy $\boldsymbol{\pi}$ that maps from state to action i.e. $\boldsymbol{\pi} : \mathcal{S} \to \mathcal{A}$, that minimizes the sum of discounted costs. The optimal policy, denoted $\boldsymbol{\pi}^\star$, is the solution to the following infinite-horizon problem

$$\begin{aligned} V^\star(\boldsymbol{s}) = \min_{\boldsymbol{\pi}} \quad & \sum_{j=0}^{\infty} \gamma^j L(\boldsymbol{x}_j, \boldsymbol{\pi}(\boldsymbol{x}_j)) && \text{(3a)} \\ \text{s.t} \quad & \forall j \in \mathbb{I}_{\geq 0} : \boldsymbol{x}_0 = \boldsymbol{s}, && \text{(3b)} \\ & \boldsymbol{x}_{j+1} = \boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{\pi}(\boldsymbol{x}_j)), && \text{(3c)} \\ & \boldsymbol{h}(\boldsymbol{x}_j, \boldsymbol{\pi}(\boldsymbol{x}_j)) \leq 0, && \text{(3d)} \\ & \boldsymbol{\pi}(\boldsymbol{x}_j) \in \mathcal{A}, && \text{(3e)} \end{aligned}$$

where $V^\star(\boldsymbol{s})$ is the optimal value function, $\gamma \in (0, 1]$ is a discount factor and $\{\boldsymbol{x}_j\}_{j=1}^{\infty}$ is the predicted state trajectory under the policy $\boldsymbol{\pi}$ for an initial state $\boldsymbol{s}$. The optimal action-value function can then be defined as follows

$$Q^\star(\boldsymbol{s}, \boldsymbol{a}) = L(\boldsymbol{s}, \boldsymbol{a}) + \gamma V^\star(\boldsymbol{f}(\boldsymbol{s}, \boldsymbol{a})), \tag{4}$$

and is related to the value function through the Bellman equality

$$V^\star(\boldsymbol{s}) = Q^\star(\boldsymbol{s}, \boldsymbol{\pi}^\star(\boldsymbol{s})) = \min_{\boldsymbol{a}} Q^\star(\boldsymbol{s}, \boldsymbol{a}). \tag{5}$$

We parameterize the value function using parameter vector $\boldsymbol{\theta}$ and adjust these using RL towards the optimal value function, and thereby the optimal policy.

A. MPC AS A FUNCTION APPROXIMATOR IN RL

As proposed by the authors in [6], we will use a parametric MPC scheme as a function approximator in RL. A parameterized finite-horizon MPC scheme is formulated as

$$\begin{aligned} V_{\boldsymbol{\theta}}(\boldsymbol{s}) = \min_{\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\sigma}} \quad & \gamma^N (T_{\boldsymbol{\theta}}(\boldsymbol{x}_N) + \boldsymbol{\psi}_N^\top \boldsymbol{\sigma}_N) \\ & + \sum_{j=0}^{N-1} \gamma^j (\ell_{\boldsymbol{\theta}}(\boldsymbol{x}_j, \boldsymbol{u}_j) + \boldsymbol{\psi}^\top \boldsymbol{\sigma}_j) && \text{(6a)} \\ \text{s.t} \quad & \forall j \in \mathbb{I}_{0:N-1} : \boldsymbol{x}_0 = \boldsymbol{s}, && \text{(6b)} \\ & \boldsymbol{x}_{j+1} = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_j, \boldsymbol{u}_j), , && \text{(6c)} \\ & \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}_j, \boldsymbol{u}_j) \leq \boldsymbol{\sigma}_j, \quad \boldsymbol{h}_{\boldsymbol{\theta}}^N(\boldsymbol{x}_N) \leq \boldsymbol{\sigma}_N, && \text{(6d)} \\ & \boldsymbol{\sigma}_j \geq 0, \quad \boldsymbol{\sigma}_N \geq 0, && \text{(6e)} \\ & \boldsymbol{u}_j \in \mathcal{A}, && \text{(6f)} \end{aligned}$$

where $\boldsymbol{x} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N\}$ and $\boldsymbol{u} = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}\}$. In the objective $\ell_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{u})$ denotes the parameterized stage cost and $T_{\boldsymbol{\theta}}(\boldsymbol{x})$ the parameterized terminal cost. The parameterized prediction model is denoted $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{u})$, $\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{u})$ describes the mixed input and state constraints, and $\boldsymbol{h}_{\boldsymbol{\theta}}^N(\boldsymbol{x})$ describes the terminal constraint. Slack variables $\boldsymbol{\sigma}_j$ and $\boldsymbol{\sigma}_N$ are used to prevent the MPC scheme from becoming infeasible due to the possible model mismatch between the true system (1) and the prediction model $\boldsymbol{f}_{\boldsymbol{\theta}}$. The constant vectors $\boldsymbol{\psi}$ and $\boldsymbol{\psi}_N$ should be selected sufficiently large, such that constraint violations are accepted as seldom as possible while still ensuring feasibility [20]. Stability and constraint satisfaction of the parameterized MPC scheme in (6) is addressed in e.g. [21].

Using MPC as a function approximator, the policy is given by the first element in the input sequence which is the solution to (6), i.e.

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{s}) = \boldsymbol{u}_0^\star(\boldsymbol{s}, \boldsymbol{\theta}). \tag{7}$$

Remark 1: It was shown in [6, Theorem 1], that given a parameterization of the cost and constraints that is rich enough, the MPC scheme can capture the optimal policy even using an inaccurate prediction model.

We note that rich function approximators, such as NNs, can be used with the proposed framework to capture complex functions in the MPC scheme. In [22], the authors suggest using convex NNs to modify the stage cost in (6a). Next, we will describe Q-learning which is one method that can be used to update the parameters $\boldsymbol{\theta}$.

B. Q-LEARNING

Q-learning is an RL method based on learning the optimal action-value function $Q^\star(\boldsymbol{s},\boldsymbol{a})$, which describes the joint desirability of a given state-action pair [7]. Q-learning is a leading model-free RL alternative, which learns the Q-function directly from experience, without requiring access to a model.

Using MPC as a function approximator, we can estimate the Q-function by constraining the first action in the input sequence in (6), according to

$$\begin{aligned} Q_{\boldsymbol{\theta}}(\boldsymbol{s},\boldsymbol{a}) = \min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\sigma}} \quad & \gamma^N(T_{\boldsymbol{\theta}}(\boldsymbol{x}_N) + \boldsymbol{\psi}_N^\top \boldsymbol{\sigma}_N) \\ & + \sum_{j=0}^{N-1} \gamma^j(\ell_{\boldsymbol{\theta}}(\boldsymbol{x}_j,\boldsymbol{u}_j) + \boldsymbol{\psi}^\top \boldsymbol{\sigma}_j) \qquad (8a) \\ \text{s.t} \quad & (6b)-(6f'), \quad \boldsymbol{u}_0 = \boldsymbol{a}. \qquad (8b) \end{aligned}$$

It can be shown that the Q-function estimate (8), the value function estimate (6), and the policy (7) satisfy the Bellman equality in (5) [6]. This implies that in a Q-learning setting, the policy estimate is obtained as

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{s}) = \arg\min_{\boldsymbol{a}} Q_{\boldsymbol{\theta}}(\boldsymbol{s},\boldsymbol{a}). \qquad (9)$$

Q-learning aims to update the parameters $\boldsymbol{\theta}$ such as to minimize the estimation error of the Q-function, which in a deterministic setting can be expressed as (see e.g. [7])

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{s}_0}\left[\frac{1}{K}\sum_{k=0}^{K-1}\left(Q^\star(\boldsymbol{s}_k,\boldsymbol{a}_k) - Q_{\boldsymbol{\theta}}(\boldsymbol{s}_k,\boldsymbol{a}_k)\right)^2\right], \qquad (10)$$

where K is the episode length and the expectation $\mathbb{E}_{\boldsymbol{s}_0}$ is taken over either randomly distributed initial conditions or fixed initial conditions. However, as the true action-value function $Q^\star(\boldsymbol{s},\boldsymbol{a})$ is generally not known, a classical approach to Q-learning is parameter updates driven by the temporal difference (TD) error [7] defined as

$$\delta_k = L(\boldsymbol{s}_k,\boldsymbol{a}_k) + \gamma V_{\boldsymbol{\theta}}(\boldsymbol{s}_{k+1}) - Q_{\boldsymbol{\theta}}(\boldsymbol{s}_k,\boldsymbol{a}_k). \qquad (11)$$

At each time step the parameters are updated according to

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta_k \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\boldsymbol{s}_k,\boldsymbol{a}_k), \qquad (12)$$

where $\alpha > 0$ is a scalar denoting the step size. The gradient $\nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\boldsymbol{s}_k,\boldsymbol{a}_k)$ can be obtained from sensitivity analysis of the MPC scheme, as detailed in [6].

An alternative to the incremental update of parameters in (12), is a batch approach to Q-learning. This method is known to result in more stable learning [7]. A batch approach entails introducing an additional set of parameters $\tilde{\boldsymbol{\theta}}$ that is continuously being updated

$$\tilde{\boldsymbol{\theta}} \leftarrow \tilde{\boldsymbol{\theta}} + \alpha\tilde{\delta}_k \nabla_{\tilde{\boldsymbol{\theta}}} Q_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{s}_k,\boldsymbol{a}_k), \qquad (13)$$

where $\tilde{\delta}_k = L(\boldsymbol{s}_k,\boldsymbol{a}_k) + \gamma V_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{s}_{k+1}) - Q_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{s}_k,\boldsymbol{a}_k)$. The action $\boldsymbol{a}_k$ is selected according to the action-value function $Q_{\boldsymbol{\theta}}(\boldsymbol{s}_k,\boldsymbol{a}_k)$, defined by parameters $\boldsymbol{\theta}$ that remain fixed for the duration of one batch. As the updated parameters $\tilde{\boldsymbol{\theta}}$ converge, which marks the end of a batch, we replace the fixed parameters $\boldsymbol{\theta}$ with the updated ones $\tilde{\boldsymbol{\theta}}$ and begin a new batch.

Q-learning techniques aim to fit $Q_{\boldsymbol{\theta}}(\boldsymbol{s},\boldsymbol{a})$ to $Q^\star(\boldsymbol{s},\boldsymbol{a})$, with the hope that $Q_{\boldsymbol{\theta}} \approx Q^\star$ will result in $\boldsymbol{\pi}_{\boldsymbol{\theta}} \approx \boldsymbol{\pi}^\star$. To make this fitting possible, we have to deviate from the current policy estimate, i.e. explore. A standard strategy for exploring in the case of continuous actions is adding random perturbations to the policy, e.g. in the form of Gaussian noise. This results in a stochastic policy that induces undirected exploration, i.e.

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) = \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \boldsymbol{\zeta}_a, \qquad (14)$$

where $\boldsymbol{\zeta}_a$ is normally distributed according to $\boldsymbol{\zeta}_a \sim \mathcal{N}(0,\boldsymbol{\sigma}_a^2 I_a)$ where $\boldsymbol{\sigma}_a$ is the standard deviation and I_a is the identity matrix. Convergence properties for Q-learning are established in e.g. [23] and elaborated further in Section V-B. The stochastic policy in (14) will serve as a baseline for the variance-based exploration method proposed in this paper.

III. PARAMETER SPACE EXPLORATION

Exploration in parameter space is closely related to the concept of randomized value functions, which may be used as an alternative to TS without the need for an intractable exact posterior representation. Exploration in parameter space has been studied in, e.g., [10] and [16]. The referenced works are similar in the sense that NNs are used for approximating the value function, and that a sample from an approximate posterior distribution of the value function is used to explore.

To the best of the authors' knowledge, only exploration in action space has been tested for MPC as a function approximator in RL. Comparing NNs and MPC schemes as function approximators, we conjecture that exploration in parameter space is particularly suited when using MPC, as the parametrization is smaller and less convoluted than for NNs (due to the layers and consecutive nonlinear activations), and also more easily interpreted.

We therefore propose to adopt exploration in the parameter space for MPC, by adding uncorrelated Gaussian noise to the parameters as follows

$$\hat{\boldsymbol{\theta}} = \boldsymbol{\theta} + \boldsymbol{\zeta}_p, \qquad (15)$$

where $\boldsymbol{\zeta}_p \sim \mathcal{N}(0,\boldsymbol{\sigma}_p^2 I_p)$. The exploration policy is then obtained by acting greedily with respect to the Q-function defined by the perturbed parameters, i.e.

$$\hat{\boldsymbol{\pi}}_{\hat{\boldsymbol{\theta}}} = \arg\min_{\boldsymbol{a}} Q_{\hat{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a}). \qquad (16)$$

The exploration method is summarized in Algorithm 1, here for a continuous task. We note that the method easily extends to an episodic task.

Algorithm 1 Exploration in Parameter Space

Input: Initial MPC parameters $\boldsymbol{\theta}_0$, initial learning parameters $\tilde{\boldsymbol{\theta}}_0$, initial state $\boldsymbol{s}_0$, batch update frequency b, learning step size α, parameter noise standard deviation $\boldsymbol{\sigma}_p$, length of simulation $k_{\max}$;
Output: Policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$
while $k \leq k_{max}$ **do**
  if $mod(k, b) = 0$ **then**
    Update MPC parameters with learned parameters $\boldsymbol{\theta}_k = \tilde{\boldsymbol{\theta}}_k$
    Perturb parameters to get $\hat{\boldsymbol{\theta}}_k$ (15)
  Act greedily w.r.t. current Q-estimate (16)
  Update $\tilde{\boldsymbol{\theta}}_k$ (13)
  $k \leftarrow k + 1$

Remark 2: We note that exploration in parameter space in combination with Q-learning, generally calls for a batch approach to Q-learning as given by (13). For an incremental approach as in (12) where we only have one set of parameters $\boldsymbol{\theta}$, the resulting TD-error as we update the parameters according to (15) would be $L(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma V_{\hat{\boldsymbol{\theta}}}(\boldsymbol{s}_{k+1}) - Q_{\hat{\boldsymbol{\theta}}}(\boldsymbol{s}_k, \boldsymbol{\pi}_{\hat{\boldsymbol{\theta}}})$ where $Q_{\hat{\boldsymbol{\theta}}}(\boldsymbol{s}_k, \boldsymbol{\pi}_{\hat{\boldsymbol{\theta}}}) = V_{\hat{\boldsymbol{\theta}}}(\boldsymbol{s}_k)$, i.e. we are fitting the V-function rather than the Q-function.

Depending on the selected parameterization and the resulting range of parameters, which in turn depends on the problem at hand, we may choose to perturb the normalized parameters in order to use the same scale for perturbing the entire parameter vector. Alternatively, the states and actions in the problem can also be scaled, which will result in a smaller variation in parameter range, which is also known to speed up learning.

In the next section, we will consider an alternative to adding random perturbations to the parameters, namely adding a perturbation based on the uncertainty of the value function. We will use the exploration method in (15) as a second baseline for our proposed method.

IV. VALUE FUNCTION IDW VARIANCE

To guide exploration using the uncertainty of our value function estimate, a method is needed for quantifying such uncertainty. In Bayesian exploration, we use the covariance of the resulting posterior distribution of a Gaussian process (GP), to guide exploration. Alternatively, interpolation methods can be used to define non-probabilistic uncertainty measures that are computationally cheaper to evaluate. In [24], radial basis functions (RBFs) were used to formulate a measure of uncertainty based on sampled points. In [25], an uncertainty measure based on IDW was compared to a Bayesian exploration for global optimization and showed competitive performance. We propose to use IDWs for quantifying the uncertainty of the value function.

The IDW framework can be used to estimate uncertainty for both the value function and the action-value function. Because we will use the IDW variance to measure parametric uncertainty only, we do not need to consider the additional input argument of the action-value function, and therefore choose to apply IDW to the value function.

A. INVERSE DISTANCE WEIGHTING

Given a data set, IDW is an interpolation method that also provides us with a variance estimate given a predictor of the function that is sampled. We assume that we have a data set consisting of M samples $V_i = V(\boldsymbol{\eta}_i)$ of $V : \mathbb{R}^q \to \mathbb{R}$ at corresponding points $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_M$. In the following, the function V is the value function, that we for now assume that we can sample, and $\boldsymbol{\eta} = [\boldsymbol{\theta}, \boldsymbol{s}]^\top$.

We may consider the following scaling function $\boldsymbol{\phi} : \mathbb{R}^q \to \mathbb{R}^q$ to be immune to the different scaling of individual parameters and states [25]

$$\boldsymbol{\phi}(\boldsymbol{\eta}) = \operatorname{diag}\left(\frac{2}{\boldsymbol{\eta}_{\max} - \boldsymbol{\eta}_{\min}}\right)\left(\boldsymbol{\eta} - \frac{\boldsymbol{\eta}_{\max} + \boldsymbol{\eta}_{\min}}{2}\right), \quad (17)$$

so that $\boldsymbol{\phi}(\boldsymbol{\eta}) \in [-1, 1]$ for all $\boldsymbol{\eta} \in \left[\boldsymbol{\eta}_{\min}, \boldsymbol{\eta}_{\max}\right]$, where $[\boldsymbol{\eta}_{\max}, \boldsymbol{\eta}_{\min}] \subset \mathbb{R}^q$. The min and max values of the states and parameters can be based on constraints and reasonable bounds on possible parameter values.

For a new instance of $\boldsymbol{\eta}$, we consider the (scaled) squared Euclidean distance function $d^2 : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ given as

$$d^2(\boldsymbol{\eta}, \boldsymbol{\eta}_i) = (\boldsymbol{\phi}(\boldsymbol{\eta}_i) - \boldsymbol{\phi}(\boldsymbol{\eta}))^\top (\boldsymbol{\phi}(\boldsymbol{\eta}_i) - \boldsymbol{\phi}(\boldsymbol{\eta})), \quad (18)$$

for $i = 1, \ldots, M$. The IDW function $w_i : \mathbb{R}^q \to \mathbb{R}$ can then be defined as in [26]

$$w_i(\boldsymbol{\eta}) = \frac{1}{d^2(\boldsymbol{\eta}, \boldsymbol{\eta}_i)}, \quad (19)$$

and assigns larger weights $w_i(\boldsymbol{\eta})$ to samples that are close to $\boldsymbol{\eta}$ than to samples further away. In [27], the following alternative weighting function was suggested

$$w_i(\boldsymbol{\eta}) = \frac{e^{-d^2(\boldsymbol{\eta}, \boldsymbol{\eta}_i)}}{d^2(\boldsymbol{\eta}, \boldsymbol{\eta}_i)}. \quad (20)$$

The weighting function in (20) is similar to (19) for small values of d^2, but more quickly reduces the effect of points $\boldsymbol{\eta}_i$ far away from $\boldsymbol{\eta}$ due to the exponential term. We then define the following function $v_i : \mathbb{R}^q \to \mathbb{R}$ as

$$v_i(\boldsymbol{\eta}) = \frac{w_i(\boldsymbol{\eta})}{\sum_{j=1}^{M} w_j(\boldsymbol{\eta})} \quad (21)$$

which allows us to define

$$\bar{V}(\boldsymbol{\eta}) = \sum_{i=1}^{M} v_i(\boldsymbol{\eta}) V_i, \quad (22)$$

which is an IDW interpolation of $\{(\boldsymbol{\eta}_i, V_i)\}_{i=1}^{M}$. For a new instance of $\boldsymbol{\eta}$, an estimate of $V(\boldsymbol{\eta})$ is obtained by interpolating on the M existing samples of V. In its original form, the selection function in (21), will include the option that in case we evaluate an already sampled instance of $\boldsymbol{\eta}$, i.e. $\boldsymbol{\eta} \in \{\boldsymbol{\eta}_i, \ldots, \boldsymbol{\eta}_M\}$, we use that $v_i(\boldsymbol{\eta}_j) = 1$ for $i = j$ and $v_i(\boldsymbol{\eta}_i) = 0$ otherwise. However, for our purpose we will not

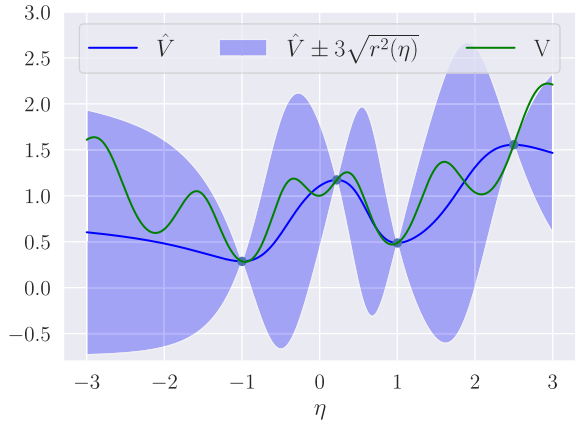


FIGURE 1. Example of IDW: Function V (green) and samples (η_i, V_i) (blue dots). Error bands are given using the IDW variance estimate in (23) i.e. $\pm 3\sqrt{r^2(\eta)}$ (shaded blue) evaluated for the predictor $\hat{V}(\eta)$.

evaluate already sampled values of $\boldsymbol{\eta}$, as we wish to explore new values of the parameters. This is explained further in Section V-A. It was shown in [25, Lemma 1], for both choices of the weighting function, (19) and (20), that the interpolation function $\bar{V}(\boldsymbol{\eta})$ is differentiable everywhere on $\mathbb{R}^q$.

Based on the IDW interpolation function, we define the IDW variance function $r^2 : \mathbb{R}^q \to \mathbb{R}$ as given by [25], [27], [28]

$$r^2(\boldsymbol{\eta}) = \sum_{i=1}^{M} v_i(\boldsymbol{\eta})\big(V_i - \hat{V}(\boldsymbol{\eta})\big)^2, \tag{23}$$

where $V_{\boldsymbol{\theta}}(\boldsymbol{s}) = \hat{V}(\boldsymbol{\eta})$, $V_{\boldsymbol{\theta}}(\boldsymbol{s})$ being the parameterized value function in (6) for which we want to estimate the variance. Essentially, the IDW variance estimate is a weighted average of the squared error between the sampled value functions V_i and the predictor $\hat{V}(\boldsymbol{\eta})$. As $\hat{V}(\boldsymbol{\eta})$ is differentiable, it follows that the IDW variance estimate is also differentiable everywhere on $\mathbb{R}^q$. Figure 1 is one example of how the IDW variance estimate can be used to define error bounds for a predictor of a function V with, in this case, a scalar argument $\boldsymbol{\eta}$.

B. P-STEP TD PREDICTION OF V

Monte Carlo (MC) learning involves learning from experience, using sequences of states, actions, and costs. MC learning offers an alternative method to TD-learning (11) of the action-value or value function. For a general MDP with stochastic dynamics and possibly a stochastic policy, MC learning uses the mean sum of discounted costs from several episodes of experience as the value function estimate. In the following, we propose to use MC learning of the value function as a target in the IDW framework.

As we are considering deterministic policies for deterministic environments, one episode is sufficient to provide an MC value function estimate. Because the true value function is defined for an infinite horizon (3), we bootstrap on our value function estimate to compensate for considering an episode with finite length. We estimate the value function as the sum of discounted costs, evaluated for p samples of the state and $p-1$ samples of the action, bootstrapping on the value function approximator for the last state. This can be described as a p-step prediction of V, based on a *first-visit* approach to MC learning [7].

We now consider a policy $\boldsymbol{\pi}_{\bar{\boldsymbol{\theta}}}$, where $\bar{\boldsymbol{\theta}}$ denotes the parameter vector. For notational convenience we let $\boldsymbol{\pi}_{\bar{\boldsymbol{\theta}}} = \bar{\boldsymbol{\pi}}$. For $p \geq 1$ we consider sampled states and actions, i.e.

$$\{\boldsymbol{s}_{k-p}, \boldsymbol{a}_{k-p}, \boldsymbol{s}_{k-p+1}, \ldots, \boldsymbol{a}_{k-1}, \boldsymbol{s}_k\}, \tag{24}$$

obtained by acting according to policy $\bar{\boldsymbol{\pi}}$. The data in (24) is used to predict the value function by evaluating the sum of discounted costs. The V estimate is given as

$$\begin{aligned} V_k(\bar{\boldsymbol{\theta}}_k, \boldsymbol{s}_{k-p}) = {} & L(\boldsymbol{s}_{k-p}, \boldsymbol{a}_{k-p}) + \gamma L(\boldsymbol{s}_{k-p+1}, \boldsymbol{a}_{k-p+1}) \\ & + \gamma^2 L(\boldsymbol{s}_{k-p+2}, \boldsymbol{a}_{k-p+2}) + \ldots \\ & + \gamma^{p-1} L(\boldsymbol{s}_{k-1}, \boldsymbol{a}_{k-1}) + \gamma^p V_{\bar{\boldsymbol{\theta}}_k}(\boldsymbol{s}_k), \end{aligned} \tag{25}$$

where we evaluate the value function estimate $V_{\bar{\boldsymbol{\theta}}}$ for the last sampled state, using the current parameter vector $\bar{\boldsymbol{\theta}}_k$. The number of samples p thereby becomes a hyperparameter. Because we bootstrap on our value function, both the targets and the predictor in (23) are based on the function approximator in (6). It is therefore important that p is large enough, such that the effect of bootstrapping is small. Moreover, the effect of discounting will reduce the bias of bootstrapping. Selecting p large relative to the batch size, means that we obtain samples of the value function for only a few instances of the state. In an episodic setting where we initialize the system from the same initial condition at the beginning of a new batch, we will evaluate the value function variance for the same state and therefore depend on fewer samples of the value function target. In a continuing task on the other hand, the system can be in different states at the beginning of each new batch, and we need to provide samples of the value function targets for more instances of the state. An alternative to the target we proposed in (25), is an exponentially weighted estimate as the authors propose for the advantage function in [29].

Remark 3: As the variance-based exploration method is perturbing in parameter space, we should, as for the random exploration in parameter space, use a batch manner to Q-learning as given in (13). This entails having two sets of parameters, where one set of parameters is continuously updated $\tilde{\boldsymbol{\theta}}$, whereas the other set of parameters $\boldsymbol{\theta} = \bar{\boldsymbol{\theta}}$ are fixed for one batch and is used to define a policy that visits informative states.

Using the IDW variance function in (23), we obtain a variance estimate based on the weighted average of the squared difference between the targets in (25) and the MPC-based value function approximator in (6). The V-function estimate in (25) is particular for the current policy estimate $\bar{\boldsymbol{\pi}}$, i.e. we are estimating $V^{\bar{\boldsymbol{\pi}}}$. As the parameters are updated from $\bar{\boldsymbol{\theta}}$ to $\bar{\boldsymbol{\theta}}'$ at the beginning of a new batch, data is collected

with an updated policy $\bar{\boldsymbol{\pi}}'$, and we are making p-step predictions of $V^{\bar{\boldsymbol{\pi}}'}$. Although the previously sampled V targets are estimated for increasingly more outdated policies, they can be useful for estimating the uncertainty w.r.t the parameters. The weights in (23) are assigned using inverse distances according to either (19) or (20), i.e. we influence the impact of previously sampled targets on our variance estimate through the choice of IDW function.

C. PRACTICAL IMPLEMENTATION

As the number of samples M increases, the IDW variance function becomes increasingly computationally heavy to evaluate. For a practical implementation, that can run fast in real-time, we set a limit for the maximum number of samples $M_{\max}$ used to evaluate (23). If our data set already contains $M_{\max}$ samples, we evaluate the following criterion for a new instance of $\boldsymbol{\eta}$,

$$\sum_{i=1}^{M_{\max}} d^2(\boldsymbol{\eta}, \boldsymbol{\eta}_i) > \min \left\{ \sum_{i=1}^{M_{\max}} d^2(\boldsymbol{\eta}_i, \boldsymbol{\eta}_1), \ldots, \sum_{i=1}^{M_{\max}} d^2(\boldsymbol{\eta}_i, \boldsymbol{\eta}_{M_{\max}}) \right\}. \tag{26}$$

If the summed distance from a new sample $\boldsymbol{\eta}$ in (26) to our current samples $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_{M_{\max}}$ is larger than the least different sample in our dataset, we will replace the old sample with the new. In the opposite case, we will not update our data set. The maximum number of samples $M_{\max}$ thereby becomes a hyperparameter.

Remark 4: Although the size of the data set in this framework can be controlled by limiting the number of samples to include, the parameter and state dimension will also affect the size. The IDW variance (23) is only evaluated at the beginning of each batch, so the computation time of the variance itself may therefore not necessarily be a problem. Nonetheless, a small parameter space will ease the work related to handling the sampled data needed to evaluate the IDW variance. This framework is therefore particularly suited for using MPC as a function approximator, which typically uses much fewer parameters than the standard choice of NNs.

V. VARIANCE-BASED EXPLORATION

In this section, we will outline how we can leverage an IDW framework as detailed in the previous section, to direct exploration to where we have uncertainty in our value function estimate. We are then acting according to the strategy of "optimism in the face of uncertainty", and hoping that by exploring areas of high uncertainty our policy will visit more informative states and actions, and hence explore more efficiently.

Combining a variance-based exploration in the parameter space with a batch approach to Q-learning, allows us to consider a perturbation to the parameters at the beginning of each batch, and keep these parameter values for the duration of one batch. The advantage of this approach is that we induce a state-dependent change in the policy over multiple time steps, what is often referred to as *deep* exploration.

A. VARIANCE-BASED PERTURBATIONS IN PARAMETER SPACE

We first define the gradient of the IDW variance function w.r.t. the parameters:

$$\nabla_{\boldsymbol{\theta}} r^2(\boldsymbol{\eta}) = \sum_{i=1}^{M} \nabla_{\boldsymbol{\theta}} v_i(\boldsymbol{\eta})(V_i - \hat{V}(\boldsymbol{\eta}))^2 - 2\, v_i(\boldsymbol{\eta})(V_i - \hat{V}(\boldsymbol{\eta}))\nabla_{\boldsymbol{\theta}} \hat{V}(\boldsymbol{\eta}), \tag{27}$$

to highlight the fact that the sensitivity of the MPC scheme, in terms of $\nabla_{\boldsymbol{\theta}} \hat{V}(\boldsymbol{\eta})$, is used in evaluating the gradient of the variance. We propose to restrict the variance gradient by using the standard deviation used for Gaussian exploration in parameter space. This will, first of all, make the two methods highly comparable, in terms of how large the applied perturbation in the parameters can be, and also prevents the addition of yet another hyperparameter, i.e.

$$\nabla_{\boldsymbol{\theta}} \hat{r}^2(\boldsymbol{\theta}, \boldsymbol{s}) = \mathrm{sat}(\nabla_{\boldsymbol{\theta}} r^2(\boldsymbol{\theta}, \boldsymbol{s}), -2\sigma_p, 2\sigma_p). \tag{28}$$

We propose the following update of the parameters

$$\bar{\boldsymbol{\theta}} = \boldsymbol{\theta} + \frac{1}{2}\nabla_{\boldsymbol{\theta}} \hat{r}^2(\boldsymbol{\theta}, \boldsymbol{s}) + \frac{1}{2}\boldsymbol{\zeta}_p, \tag{29}$$

where $\boldsymbol{\zeta}_p$ is a noise term as defined for (15). The gradient of the IDW function is added to the parameters, in the hope that exploring parameter values in a direction where our value function is uncertain, may improve our estimate. The noise term is added to ensure some random exploration in parameter space, to collect data such that the variance estimate of our function approximator is meaningful. A policy estimate based on the perturbed parameters in (29), is obtained according to

$$\boldsymbol{\pi}_{\bar{\boldsymbol{\theta}}}(\boldsymbol{s}_k) = \arg\min_{\boldsymbol{a}} Q_{\bar{\boldsymbol{\theta}}}(\boldsymbol{s}_k, \boldsymbol{a}_k). \tag{30}$$

The formulation in (29) and (30) resembles the exploration method of randomized value functions, where a value function is sampled from an approximate posterior distributed and then used for greedy action selection. However, our approach is not completely random in sampling the value function but uses the gradient of the variance estimate to guide the sampling. We predict V_i for each realization of $\bar{\boldsymbol{\theta}}$ as well as different states, using p samples of states and actions during a batch, and store it in the data set $\mathcal{D}$. The data set is used to re-evaluate the variance gradient at the beginning of the next batch, to generate a new (perturbed) parameter vector to be used. This is summarized in Algorithm 2.

B. CONVERGENCE PROPERTIES

Without considering that we are using function approximators, Q-learning will converge under the following assumptions: (1) Greedy in the limit with infinite exploration (GLIE), (2) the step sizes α_k satisfy the Robbins-Munro sequence. As we do not make any changes to the proof of convergence

Algorithm 2 Variance-Based Exploration

Input: Initial MPC parameters $\boldsymbol{\theta}_0$, initial learning parameters $\tilde{\boldsymbol{\theta}}_0$, initial state $\boldsymbol{s}_0$, data set $\mathcal{D}$, batch update frequency b, learning step size α, number of samples used to generate V targets p, maximum number of samples in data set $M_{\max}$, parameter noise standard deviation $\boldsymbol{\sigma}_p$, length of simulation $k_{\max}$;
Output: Policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$
while $k \leq k_{max}$ **do**
  if $mod(k, b) = 0$ **then**
    Update MPC parameters with learned parameters $\boldsymbol{\theta}_k = \tilde{\boldsymbol{\theta}}_k$
    Evaluate gradient of IDW variance (27)
    Ensure that the IDW variance gradient respects bound (28)
    Perturb parameters to get $\bar{\boldsymbol{\theta}}_k$ (29)
  Act greedily w.r.t. current Q-estimate (30)
  if $mod(k, b) \geq p$ **then**
    if $|\mathcal{D}| \leq M_{max}$ *or* $(|\mathcal{D}| \geq M_{max}$ *and (26))* **then**
      Calculate p-step prediction of V_k (25)
      Add $\{V_k, \boldsymbol{s}_{k-p}, \bar{\boldsymbol{\theta}}_k\}$ to $\mathcal{D}$
  Update $\tilde{\boldsymbol{\theta}}_k$ (13)
  $k \leftarrow k + 1$

for Q-learning, we do not include it here but refer the interested readers to [23]. The GLIE assumption entails that (i) all state-action pairs are explored infinitely many times, and that, (ii) as time goes to infinity, the policy converges to a greedy policy. The second assumption (2) is not directly related to the exploration method and is most commonly satisfied in practice by selecting a small constant step size. Formally, we can ensure GLIE for the proposed exploration method in Section (V-A). The first part of GLIE (i) is ensured by keeping a random term in (29) so that we ensure sufficient exploration even though the gradient of the variance eventually may converge to a small number. The second part of GLIE (ii) can be ensured by using a decaying scalar i.e. $\beta(\frac{1}{2}\nabla_{\boldsymbol{\theta}_k}\hat{r}^2(\boldsymbol{\theta}_k, \boldsymbol{s}_k) + \frac{1}{2}\boldsymbol{\zeta}_p)$ where $\beta = \beta_a \exp(-\omega k)$ and ω is a hyperparameter.

VI. SIMULATION EXAMPLES

We apply the proposed exploration method to two simulation examples, namely on a linear quadratic regulator (LQR) problem and an inverted pendulum on a cart system. The latter is a popular example in the control systems literature, as it is open-loop unstable and nonlinear. For each simulation example, we will benchmark our method with respect to both (*i*) Gaussian action noise and (*ii*) Gaussian parameter noise.

A. LQR

The following example is adapted from [21]. We consider a discrete linear system of the form

$$\boldsymbol{s}_{k+1} = A\boldsymbol{s}_k + B\boldsymbol{a}_k, \tag{31}$$

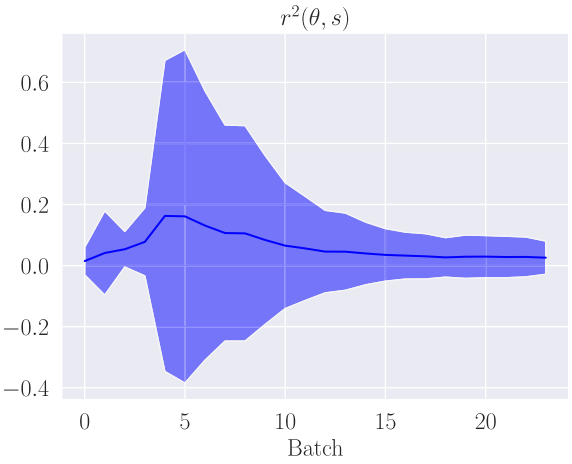


FIGURE 2. The mean and two standard deviations of the IDW variance estimate (23) over learning batches.

with system matrices

$$A = \kappa \begin{bmatrix} cos\beta & sin\beta \\ sin\beta & cos\beta \end{bmatrix}, \quad B = \begin{bmatrix} 1.1 & 0 \\ 0 & 0.9 \end{bmatrix}, \tag{32}$$

where we use $\kappa = 0.95$, and $\beta = 22°$ [deg]. The baseline stage cost is selected as

$$L(\boldsymbol{s}, \boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{s} - \boldsymbol{s}_{ref}\|^2 + \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{a}_{ref}\|^2, \tag{33}$$

where $\boldsymbol{s}_{ref} = [0.1, 0.1]^\top$, and the reference input is found accordingly. The (inaccurate) prediction model is defined as

$$A_0 = \kappa \begin{bmatrix} cos\hat{\beta} & sin\hat{\beta} \\ sin\hat{\beta} & cos\hat{\beta} \end{bmatrix}, \quad B_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{34}$$

where $\hat{\beta} = 20°$. The parameterized MPC scheme reads as

$$\min_{\boldsymbol{x},\boldsymbol{u}} \; V_0 + \gamma^N \|\boldsymbol{x}_N - \boldsymbol{x}_{\text{ref}}\|_P^2 + \sum_{j=0}^{N-1} \gamma^j \left\| \begin{bmatrix} \boldsymbol{x}_j - \boldsymbol{x}_{\text{ref}} \\ \boldsymbol{u}_j - \boldsymbol{u}_{\text{ref}} \end{bmatrix} \right\|^2 \tag{35a}$$

$$\text{s.t} \quad \forall j \in \mathbb{I}_{0:N-1} : \boldsymbol{x}_0 = \boldsymbol{s}, \tag{35b}$$

$$\boldsymbol{x}_{k+1} = A_0\boldsymbol{x}_j + B_0\boldsymbol{u}_j, \tag{35c}$$

using a prediction horizon of $N = 10$, $\gamma = 0.99$ and P is the solution to the discrete Riccati equation obtained using the inaccurate system dynamics in (34). The parameter vector is $\boldsymbol{\theta} = \{x_{\text{ref},1}, x_{\text{ref},2}, u_{\text{ref},1}, u_{\text{ref},2}, V_0\}$. We consider a continuing task and simulate the system for a total of 5000 time steps. The parameters are updated in a batch manner, using a batch length of 200, i.e., we update the parameters in the MPC scheme every 200 time steps. A learning rate of $\alpha = 0.1$ was used. For each exploration method, we consider a range of noise distributions, defined by different standard deviations. For brevity, only the best-performing ones are reported.

For the variance-based exploration method, we use $p = 10$, do not pose any restrictions on the data sampled, i.e., $M_{\max} = 5000$ and use the weighting function in (20). The resulting IDW variance estimate (23) is plotted over learning batches

TABLE 1. Cost statistics for LQR simulations. The mean and standard deviation (in parentheses) are found for a total of 5 simulations for each exploration method.

Exploration method	Exploration			Exploitation	
	σ	$k_{\max}$	$\sum L(\boldsymbol{s}, \boldsymbol{a})$	$k_{\max}$	$\sum L(\boldsymbol{s}, \boldsymbol{a})$
Variance-based	0.1	5000	**28.74 (16.52)**	20	0.021 (0.00)
Gaussian noise in parameter space	0.1	5000	57.66 (6.55)	20	0.021 (0.00)
Gaussian noise in action space	0.1	5000	132.66 (1.721)	20	**0.020 (0.00)**

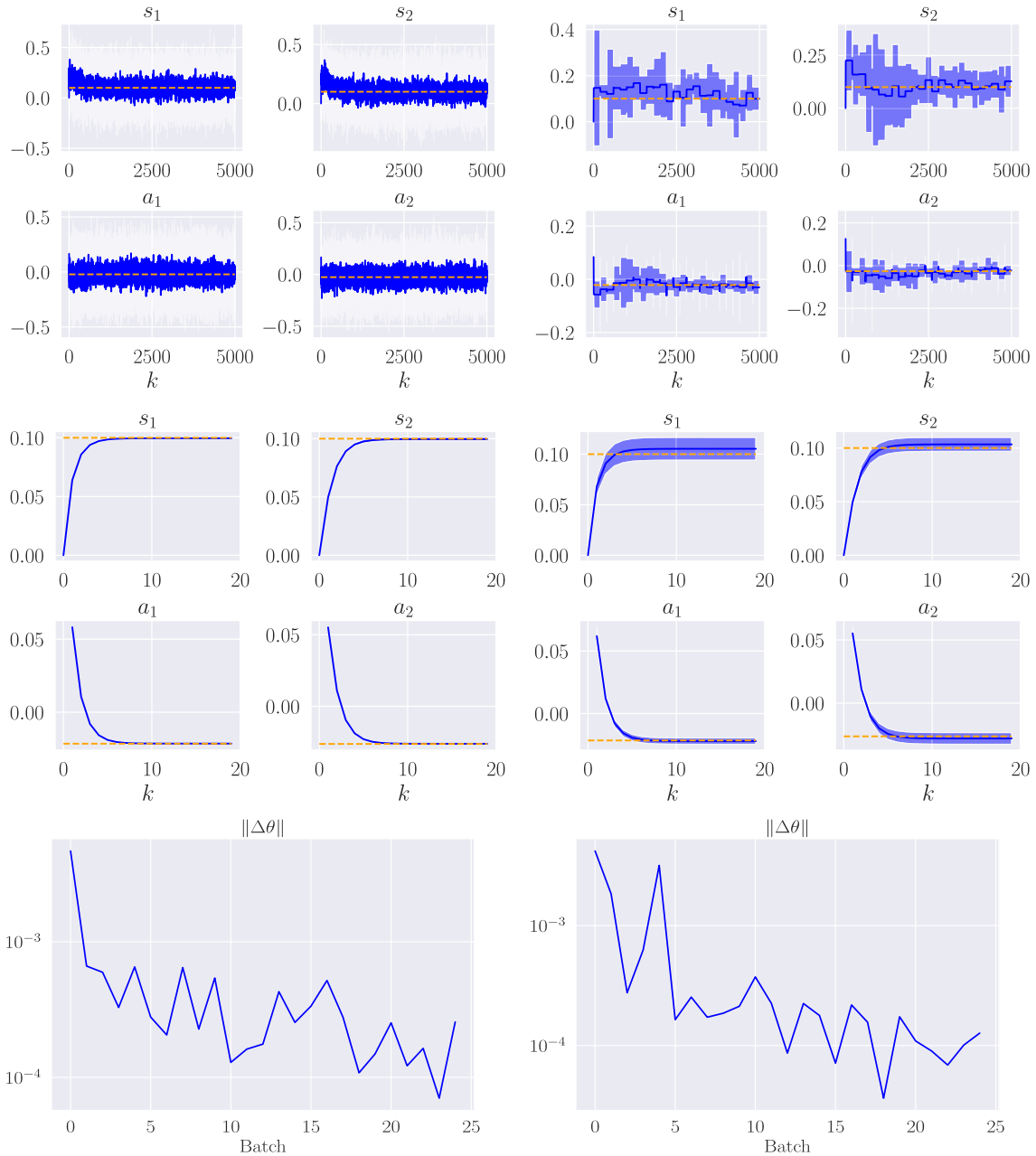


FIGURE 3. LQR simulation results. Upper plots: the mean and two standard deviations of states and actions during exploration, with Gaussian action noise (left) and variance-based exploration (right). Middle plots: the mean and two standard deviations of states and actions during exploitation, using parameters learned with Gaussian action exploration (left) and variance-based exploration (right). Bottom plots: the mean of parameter updates using Gaussian action noise (left) and variance-based exploration (right).

in Figure 2. The system states and actions are plotted both during exploration and exploitation, i.e. we use MPC with the learned parameters, resulting from Gaussian action noise and variance-based exploration, to control the system, see Figure 3. We also plot the norm of the parameter updates resulting from the different exploration methods, to indicate when the algorithm converges. Additionally, we state the cost of exploration, i.e. the sum of cost over all time steps needed to see a convergence of the parameters, as well as the sum of cost over simulations in exploitation, see Table 1. The numbers reported in Table 1 are found for a total of 5 simulations run for each exploration method.

We see that the mean of the variance in Figure 2 is initially small but eventually grows. The peak around 5 batches, seems to result in a larger parameter update which can be spotted in the lower-right plot in Figure 3. As increasingly more parameter values are tried out in the simulation, and data is gathered, the variance estimate starts decreasing. From the resulting statistics in Table 1, we see that Gaussian action noise for this particular problem obtains the best performance, in terms of minimizing the cost during exploitation, but at a much higher cost during exploration than both random perturbations as well as variance-based perturbations in parameter space. Variance-based exploration in this case obtains a slightly higher cost in exploitation compared to Gaussian action noise, although the same as with Gaussian parameter noise, while being the cheapest alternative during exploration. In Figure 3, we see that the empirical standard deviation of the simulated states and actions are visibly larger for variance-based exploration than for Gaussian action noise in exploitation, however, we note that the resulting standard deviation in the accumulated cost is small in exploitation, see Table 1.

B. INVERTED PENDULUM ON A CART

We consider the inverted pendulum on a cart as depicted in Figure 4. A classic control problem is to stabilize the inverted pendulum mounted on a cart, where the cart can only move back and forth in one dimension. In order to stabilize the pendulum in an upright position, we can change the control force acting on the cart. The dynamics, neglecting friction, are given by

$$(M+m)\ddot{z} + \frac{1}{2}ml\ddot{\phi}cos\phi = \frac{1}{2}ml\dot{\phi}^2 sin\phi + u, \tag{36a}$$

$$\frac{1}{3}ml^2\ddot{\phi} + \frac{1}{2}ml\ddot{z}cos\phi = -\frac{1}{2}mglsin\phi, \tag{36b}$$

with model parameters as specified in Table 3 and where u is the control input. The state vector is $\boldsymbol{s} = [z, \dot{z}, \phi, \dot{\phi}]^\top$, consisting of the cart displacement and velocity along the horizontal axis, the angle between the pendulum and the vertical axis and angular velocity, respectively. The action is $\boldsymbol{a} = u$. The dynamics in (36) are converted to a state space representation, discretized, and used to simulate the system. A linearized version of the state space representation is used as the prediction model in the MPC scheme. The state space representation and the linearized dynamics are found in e.g. [30]. A 4$^{\text{th}}$-order Runge Kutta scheme is used to discretize the dynamics in (36), using the step size dt $= 0.1$ s. We consider the following constraint, in newtons, on the force acting on the cart

$$-7 \leq u \leq 7. \tag{37}$$

The RL cost is given as

$$L(\boldsymbol{s}, \boldsymbol{a}) = \begin{bmatrix} \boldsymbol{s} - \boldsymbol{s}_{\text{ref}} \\ \boldsymbol{a} \end{bmatrix}^\top \begin{bmatrix} I_4 & 0 \\ 0 & 0.01 \end{bmatrix} \begin{bmatrix} \boldsymbol{s} - \boldsymbol{s}_{\text{ref}} \\ \boldsymbol{a} \end{bmatrix}, \tag{38}$$

where $\boldsymbol{s}_{\text{ref}} = [0.5, 0, 0, 0]^\top$. The linearized prediction model in the MPC scheme, defined by $\bar{A}$ and $\bar{B}$, is obtained from linearizing the system dynamics (36) at $\phi = 0$, corresponding to the pendulum being in an upright position. The parameterized MPC scheme reads as

$$\min_{\boldsymbol{x},u} \; V_0 + \gamma^N T_{\boldsymbol{\theta}}(\boldsymbol{x}_N) + \sum_{j=0}^{N-1} \gamma^j \ell_{\boldsymbol{\theta}}(\boldsymbol{x}_j, u_j) \tag{39a}$$

$$\text{s.t} \quad \forall j \in \mathbb{I}_{0:N-1} : \boldsymbol{x}_0 = \boldsymbol{s}, \tag{39b}$$

$$\boldsymbol{x}_{j+1} = \bar{A}\boldsymbol{x}_j + \bar{B}u_j, \tag{39c}$$

$$-7 \leq u_j \leq 7, \tag{39d}$$

where $N = 30$ and $\gamma = 0.99$. Moreover, we parameterize the stage cost using a quadratic function according to

$$\ell_{\boldsymbol{\theta}}(\boldsymbol{x}, u) = \begin{bmatrix} \boldsymbol{x} - \boldsymbol{x}_{\text{ref}} \\ u \end{bmatrix}^\top M(\boldsymbol{\theta}) \begin{bmatrix} \boldsymbol{x} - \boldsymbol{x}_{\text{ref}} \\ u \end{bmatrix}, \tag{40}$$

where $M(\boldsymbol{\theta})$ is a positive definite matrix, with parameterized diagonal elements. We assume that we know all state references, except the cart position, i.e. $\boldsymbol{x}_{\text{ref}} = [\theta_c, 0, 0, 0]$. A similar parameterization is also used for the terminal cost $T_{\boldsymbol{\theta}}$. The resulting parameter vector consists of the elements in $\ell_{\boldsymbol{\theta}}$ and $T_{\boldsymbol{\theta}}$, as well as V_0 and θ_c. We learn in an episodic manner, considering episodes of length 300, and let one episode correspond to one batch in terms of when we update the parameters. We learn for a total of 1000 batches and use a learning rate of $\alpha = 0.5$. We test all three exploration methods, for an interval of Gaussian distributions, defined by different standard deviations, and report the best-performing

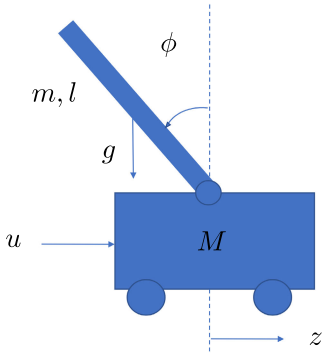


FIGURE 4. Cart-pendulum system.

TABLE 2. Cost statistics for inverted pendulum simulations. The mean and standard deviation (in parentheses) are found for a total of 3 simulations for each exploration method.

Exploration method	Exploration			Exploitation	
	σ	$k_{\max}$	$\sum L(\boldsymbol{s}, \boldsymbol{a})$	$k_{\max}$	$\sum L(\boldsymbol{s}, \boldsymbol{a})$
Variance-based	0.1	300000	**58071.39 (235.94)**	100	47.86 (0.01)
Gaussian noise in parameter space	0.1	300000	60558.04 (155.20)	100	**47.46 (0.03)**
Gaussian noise in action space	0.0001	300000	61586.95 (0.033)	100	48.86 (0.00)

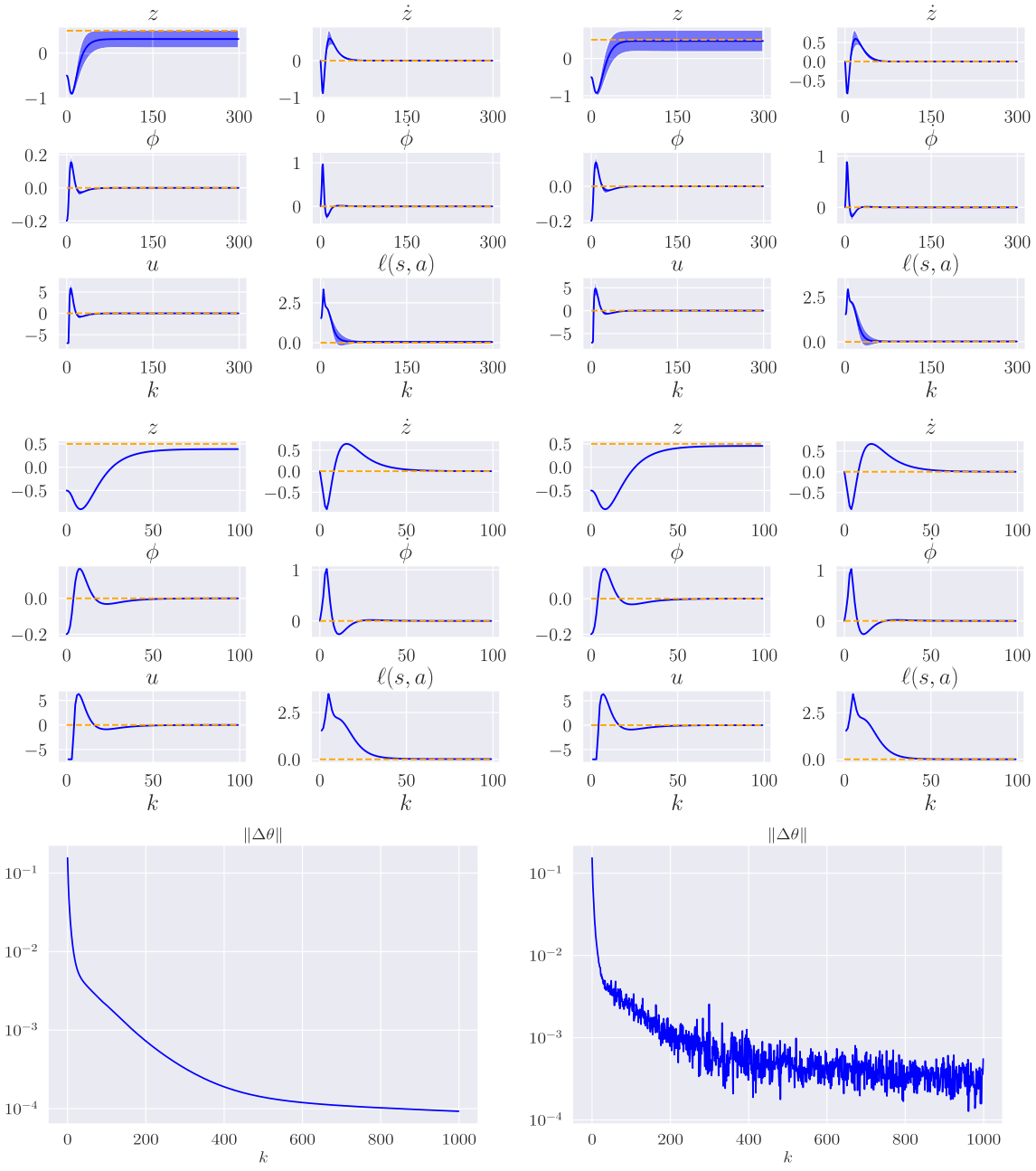


FIGURE 5. Inverted pendulum simulation results. **Upper plots:** the mean and two standard deviations of states and actions during exploration, with Gaussian action noise (left) and variance-based exploration (right). **Middle plots:** the mean of states and actions during exploitation, using parameters learned with Gaussian action exploration (left) and variance-based exploration (right). **Bottom plots:** the mean of parameter updates using Gaussian action noise (left) and variance-based exploration (right).

distribution in each category. For the variance-based exploration method, we use $p = 110$, $M_{\max} = 5000$, and the weighting function in (20).

As for the previous example, we have plotted the simulated states and actions during both exploration and exploitation, for Gaussian action noise and variance-based exploration, as well as the normed parameter updates, see Figure 5. Table 2 lists the sum of the cost for exploration and exploitation. The main goal of learning, in this case, is to obtain the true desired cart position, as well as tune the stage and terminal cost. We see from the plotted cart position during exploration, that variance-based exploration causes the system to visit positions closer to the true reference, than using Gaussian action noise. Here, this results in a small, but still visible improvement in both the plotted cart position in exploitation as well as the calculated cost in Table 2.

The statistics in Table 2 are found for a total of 3 simulations in each category. We see that Gaussian action noise in this case has very little effect, as the best performance in exploitation was obtained with $\sigma_a = 0.0001$. By using a Gaussian perturbation in parameter space we improve the performance in exploitation, while also reducing the cost of learning. Using variance-based exploration, we make exploration even cheaper while achieving a similar improvement in performance.

VII. CONCLUSION

We have presented a novel approach for variance-based exploration particularly suited for using a model predictive control (MPC) scheme as a function approximator in reinforcement learning (RL). The method is based on inverse distance weighting (IDW) to build a variance estimate of the value function approximator, which is computationally cheap compared to probabilistic methods such as Gaussian processes (GPs) and well-suited in an online setting. The proposed exploration method is tested in simulation and benchmarked against Gaussian perturbations in both action and parameter space. The results show that exploration in parameter space generally is cheaper than exploration in action space while achieving at least a similar performance in exploitation using the learned parameter values. This suggests that Gaussian exploration in parameter space, as already suggested for neural networks (NNs) as function approximators in RL, successfully can be used also with MPC. The simulation results also revealed that variance-based exploration in parameter space further reduces the cost of exploration, compared to Gaussian perturbations, with the same performance in exploitation. This means that exploration can be made even cheaper, with only a small increase in computational cost and with minor overall changes to the existing implementation. An interesting direction for future work includes verification of the proposed method for stochastic systems.

VIII. APPENDIX

See Table 3.

TABLE 3. Inverted pendulum model parameters.

Description	Symbol	Value
Cart weight	M	2.4 kg
Pendulum weight	m	0.23 kg
Acceleration of gravity	g	9.81 m/s^2
Pendulum length	l	0.36 m

REFERENCES

[1] P. Abbeel, A. Coates, M. Quigley, and A. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 19, 2006, pp. 1–8.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[3] M. Maiworm, D. Limon, and R. Findeisen, "Online learning-based model predictive control with Gaussian process models and stability guarantees," *Int. J. Robust Nonlinear Control*, vol. 31, no. 18, pp. 8785–8812, Dec. 2021.

[4] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 6, pp. 2736–2743, Nov. 2020.

[5] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, May 2013.

[6] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Trans. Autom. Control*, vol. 65, no. 2, pp. 636–648, Feb. 2020.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[8] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.

[9] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[10] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–18.

[11] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Mach. Learn.*, vol. 49, nos. 2–3, pp. 209–232, 2002.

[12] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.

[13] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, nos. 3–4, pp. 285–294, Dec. 1933.

[14] I. Osband, B. Van Roy, and Z. Wen, "Generalization and exploration via randomized value functions," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2377–2386.

[15] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.

[16] M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Noisy networks for exploration," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–21. [Online]. Available: https://openreview.net/forum?id=rywHCPkAW

[17] R. Y. Chen, S. Sidor, P. Abbeel, and J. Schulman, "UCB exploration via Q-ensembles," 2017, *arXiv:1706.01502*.

[18] Z. He, "Information-directed exploration via distributional deep reinforcement learning," in *Proc. Int. Symp. Comput. Technol. Inf. Sci. (ISCTIS)*, Jun. 2021, pp. 211–216.

[19] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1–15.

[20] E. C. Kerrigan and J. M. Maciejowski, "Soft constraints and exact penalty functions in model predictive control," in *Proc. UKACC Int. Conf. Control*, 2000, pp. 1–6.

[21] S. Gros and M. Zanon, "Towards safe reinforcement learning using NMPC and policy gradients: Part I—Stochastic case," 2019, *arXiv:1906.04057*.
[22] K. Seel, A. B. Kordabad, S. Gros, and J. T. Gravdahl, "Convex neural network-based cost modifications for learning model predictive control," *IEEE Open J. Control Syst.*, vol. 1, pp. 366–379, 2022.
[23] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Mach. Learn.*, vol. 16, no. 3, pp. 185–202, Sep. 1994.
[24] H. Gutmann, "A radial basis function method for global optimization," *J. Global Optim.*, vol. 19, pp. 201–227, Jan. 2001.
[25] A. Bemporad, "Global optimization via inverse distance weighting and radial basis functions," *Comput. Optim. Appl.*, vol. 77, no. 2, pp. 571–595, Nov. 2020.
[26] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proc. 23rd ACM Nat. Conf.*, 1968, pp. 517–524.
[27] V. R. Joseph and L. Kang, "Regression-based inverse distance weighting with applications to computer experiments," *Technometrics*, vol. 53, no. 3, pp. 254–265, Aug. 2011.
[28] A. Bemporad, "Active learning for regression by inverse distance weighting," *Inf. Sci.*, vol. 626, pp. 275–292, May 2023.
[29] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*.
[30] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge, U.K.: Cambridge Univ. Press, 2022.

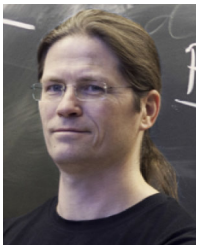
KATRINE SEEL received her M.Sc. Degree in marine cybernetics from Norwegian University of Science and Technology (NTNU) in 2017.

She currently works at the Department of Engineering Cybernetics at NTNU as a Ph.D. student, and holds a part-time position at SINTEF Digital, in the Analytics and artificial intelligence group. Her research interests include combining model predictive control with learning methods, such as reinforcement learning.

ALBERTO BEMPORAD (Fellow, IEEE) received his Master's degree cum laude in Electrical Engineering in 1993 and his Ph.D. in Control Engineering in 1997 from the University of Florence, Italy. In 1996/97 he was with the Center for Robotics and Automation, Department of Systems Science & Mathematics, Washington University, St. Louis. In 1997 and 1999 he held a postdoctoral position at the Automatic Control Laboratory, ETH Zurich, Switzerland, where he collaborated as a Senior Researcher until 2002. In 1999 and 2009 he was with the Department of Information Engineering of the University of Siena, Italy, becoming an Associate Professor in 2005. In 2010 and 2011 he was with the Department of Mechanical and Structural Engineering of the University of Trento, Italy. Since 2011he is Full Professor at the IMT School for Advanced Studies Lucca, Italy, where he served as the Director of the institute in 2012 and 2015. He spent visiting periods at Stanford University, University of Michigan, and Zhejiang University. In 2011 he co-founded ODYS S.r.l., a company specialized in developing model predictive control systems for industrial production. He has published more than 400 papers in the areas of model predictive control, hybrid systems, optimization, automotive control, and is the co-inventor of 21 patents. He is author or coauthor of various software packages formodel predictive control design and implementation, including the Model Predictive Control Toolbox (The Mathworks, Inc.) and the Hybrid Toolbox for MATLAB. He was an Associate Editor of the IEEE Transactions on Automatic Control during 2001 and 2004 and Chair of the Technical Committee on Hybrid Systems of the IEEE Control Systems Society in 2002 and 2010. He received the IFAC High-Impact Paper Award for the 2011 and 14 triennial, the IEEE CSS Transition to Practice Award in 2019, and the 2021 SAE Environmental Excellence in Transportation Award.

SÉBASTIEN GROS received his Ph.D. degree from EPFL, Switzerland, in 2007. After a journey by bicycle from Switzerland to the Everest base camp in full autonomy, he joined a R&D group hosted at Strathclyde University focusing on wind turbine control.

In 2011, he joined the university of KU Leuven, where his main research focus was on optimal control and fast MPC for complex mechanical systems. He joined the Department of Signals and Systems at Chalmers University of Technology, Goteborg in 2013, where he became associate Prof. in 2017. He is now full Prof. at NTNU, Norway and guest Prof. at Chalmers. His main research interests include numerical methods, real-time optimal control, reinforcement learning, and the optimal control of energy-related applications.



JAN TOMMY GRAVDAHL (Senior Member, IEEE) received the Siv.ing. and Dr.ing. degrees in Engineering Cybernetics from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 1994 and 1998, respectively.

He is since 2005 Professor at the Department of Engineering Cybernetics, NTNU, where he also served as Head of Department in 2008 and 09. He has supervised the graduation of 160 MSc and 15 PhD candidates. He has published five books and more than 250 papers in international conferences and journals. His current research interests include mathematical modeling and nonlinear control in general, in particular applied to turbomachinery, marine vehicles, spacecraft, robots, and highprecision mechatronic systems.

Prof. Gravdahl is since 2020 associate editor for IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY. He served as senior editor of the *IFAC journal Mechatronics* in 2016 and 20, and associate editor for the *journal Simulation Modelling Practice and Theory* in 2007 and 11. He has been on the editorial board and IPC for numerous international conferences. In 2000 and again in 2017, he was awarded the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY Outstanding Paper Award.