Variance-driven Exploration for Learning-based Model Predictive Control

Katrine Seel, Alberto Bemporad, Sébastien Gros, Jan Tommy Gravdahl

***Abstract*—Using model predictive control (MPC) schemes as function approximators in reinforcement learning (RL) enables the learning of policies with closed-loop properties that we can analyze. In general, simple heuristics based on random perturbations are still the most commonly used exploration methods in RL. This paper considers variance-based exploration in RL geared towards MPCs as function approximators. 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, actions, and rewards to estimate the uncertainty. 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 schemes using random perturbations.**

***Index Terms*—Model predictive control, Reinforcement learning, Exploration, Inverse distance weighting.**

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 are 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 to capture the policy, guarantees regarding the closed-loop behavior of the policy are difficult to provide. In [3], the authors suggest using a parameterized model predictive control (MPC) scheme as a function approximator of the policy and value function in RL. Parameterizing the MPC problem allows RL to improve the policy as data are acquired while maintaining an MPC structure, for which rich tools are available 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 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 one that minimizes 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 [4] or Boltzmann exploration [5] are used. For continuous action spaces, stochastic policies for exploration are generated e.g. by adding Gaussian noise to a deterministic policy [6]. 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 [4].

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 in order 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 [7]. 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 [8]. In order to learn efficiently, the exploration scheme should prioritize potentially informative states and actions. To do this, exploration should be done with regard to a notion of uncertainty in the value function.

Directed exploration is well understood for the bandit 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 added an exploration bonus based on a confidence interval of the reward, see, e.g., [9]. For the bandit problem, Hoeffding's inequality can be applied to obtain the UCB. Whereas for an MDP in an RL setting, this is not straightforward.

Thompson sampling (TS) is a related strategy developed for the bandit problem. A Bayesian model of the posterior distribution of rewards is consecutively sampled, and updated as data is gathered. Actions are selected by acting greedily w.r.t. the current sample [10]. Building 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 achieves 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 [11] 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 for linear parameterizations of the value function. An extension was made to nonlinear parameterizations, more specifically to NNs, in [12], 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 [13]. Rather than training an NN with K outputs or heads to build an approximate posterior distribution of Q as in [12], the authors in [13] inject noise in the NN parameters and use RL to tune the intensity i.e. the variance of the distributions. Samples of the Q-function are obtained by sampling noise intensities from the tuned noise distributions, and actions are selected by acting greedily with respect to the sampled Q-function.

Bootstrapped DQNs have also been used to develop a UCB approach that applies to RL. Namely, the bootstrapped DQN was used to create an empirical estimate of the standard deviation of the Q-function distribution [14]. This was in turn used to formulate a UCB that was added as an exploration bonus in the reward function. Along the same lines, the DQN framework was used by the authors in [15] 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 [16], 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 scheme 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 variance-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

We consider real systems that can be described as discrete-time systems with continuous state and action spaces. The state space satisfies the Markov property, i.e. future states depend only on the current state, and not past states. We denote the underlying transition probability matrix for the states $s \in \mathcal{S} \subseteq \mathbb{R}^n$ and actions $a \in \mathcal{A} \subseteq \mathbb{R}^m$ as $\mathcal{T}$, i.e.

$$s_{k+1} = \mathcal{T}(s_k, a_k), \tag{1}$$

where s_{k+1} denotes the next state and k denotes the physical time of the system. We will assume in the following that a stage cost

$$L(s_k, a_k), \tag{2}$$

is provided. Our goal is to find the parameters θ of a policy $\pi(s)$, that can be both stochastic or deterministic, and maps from state to action i.e. $\pi_\theta : \mathcal{S} \to \mathcal{A}$, so as to minimize the sum of discounted cost

$$J(\pi_\theta) = \mathbb{E}_{s_0 \sim \mathcal{S}_0, s \sim \mathcal{T}(s,\pi(s))} \left[\sum_{k=0}^{K} \gamma^k L(s_k, a_k) \mid a_k = \pi_\theta(s_k) \right], \tag{3}$$

where $\mathcal{S}_0$ is a distribution of initial states and $\gamma \in (0, 1]$ is a discount factor. We recognize that minimization of cost in (3) aligns with the RL objective of maximizing the sum of future discounted rewards $r(s, a)$ by stating that $r(s, a) = -L(s, a)$. In the following we will use $\mathbb{E}_{\pi_\theta}[\cdot]$ as short for $\mathbb{E}_{s_0 \sim \mathcal{S}_0, s \sim \mathcal{T}(s,\pi(s))}[\cdot]$.

In the next section, we will introduce MPC as a function approximator. Moreover, we will consider deterministic systems which is a special case of (1), i.e. dynamical systems of the form

$$s_{k+1} = f(s_k, a_k). \tag{4}$$

A. MPC as a function approximator in RL

As proposed by the authors in [3], we will use a parametric MPC scheme as a function approximator in RL. A parameterized finite-horizon MPC scheme is formulated as

$$\begin{aligned} V_\theta(s) = \min_{x,u,\sigma} \quad & -\lambda_\theta(s) + \gamma^N T_\theta(x_N) + \psi_N{}^\top \sigma_N \\ & + \sum_{k=0}^{N-1} \gamma^k \ell_\theta(x_k, u_k) + \psi_k^\top \sigma_k && \text{(5a)} \\ \text{s.t} \quad & x_{k+1} = f_\theta(x_k, u_k), \quad x_0 = s, && \text{(5b)} \\ & h_\theta(x_k, u_k) \leq \sigma_k, \ h_\theta(x_N) \leq \sigma_N, && \text{(5c)} \\ & \sigma_k \geq 0, \sigma_N \geq 0 && \text{(5d)} \end{aligned}$$

where $x = \{x_0, \dots, x_N\}$ and $u = \{u_0, \dots, u_{N-1}\}$. In the objective $\ell_\theta(x, u)$ denotes the parameterized stage cost, $T_\theta(x)$ the parameterized terminal cost, and $\lambda_\theta(s)$ is a storage function. The storage function is primarily useful in the case of learning stable policies for economic problems, where the economic stage cost is not necessarily positive definite. For problems that are *dissipative*, using storage function allows us to reformulate the cost, such that stability can be proved, while the solution to the MPC remains unchanged. For more details, the reader is referred to [3]. A discount factor $\gamma \in (0, 1]$ is used to establish the importance of future rewards over immediate rewards. The function $f_\theta(x, u)$ is used to model the system dynamics, $h_\theta(x_k, u_k)$ describes the mixed input and state constraints, and $h_\theta(x_N)$ describes the terminal constraint. Slack variables σ_k and σ_N are used to prevent the MPC scheme from becoming infeasible due to the possible model mismatch between the true system (4) and the prediction model f_θ. The constant vectors ψ_k and ψ_N should be selected sufficiently large, such that constraint violations are accepted as seldom as possible while still ensuring feasibility [17]. Although not clearly visible in the performance measure in (3), the inequality constraints (5c)-(5d) may be incorporated in the cost $L(s, a)$.

Using MPC as a function approximator, the policy is given by the first element in the input sequence, the solution to (5), i.e.

$$\pi_\theta(s) = u_0(s, \theta). \tag{6}$$

Next, we will consider one RL method we can use to update the parameters θ.

B. Q-learning

Q-learning is an RL method based on learning the optimal action-value function $Q^\star(s, a)$ [4]. Using MPC as a function approximator, we can estimate the optimal Q-function by constraining the first action in the input sequence, according to

$$\begin{aligned} Q_\theta(s, a) = \min_{x,u,\sigma} \quad & -\lambda_\theta(s) + \gamma^N T_\theta(x_N) + \psi_N{}^\top \sigma_N \\ & + \sum_{k=0}^{N-1} \gamma^k \ell_\theta(x_k, u_k) + \psi_k^\top \sigma_k && \text{(7a)} \\ \text{s.t} \quad & (5b) - (5d), \quad u_0 = a. && \text{(7b)} \end{aligned}$$

It can be shown that the Q-function estimate from (7), the value function estimate (5), and the policy (6) satisfy the Bellman equations [3], i.e.

$$Q^\star(s, a) = L(s, a) + \gamma V^\star(s^+), \tag{8a}$$

$$V^\star(s) = Q^\star(s, \pi^\star(s)) = \arg\min_a Q^\star(s, a), \tag{8b}$$

where s^+ denotes the consecutive state. In a Q-learning setting, the optimal policy is given by

$$\pi^\star(s) = \arg\min_a Q^\star(s, a). \tag{9}$$

We make the following assumption for the choice of parameterization, which is a common assumption in theoretical RL.

Assumption 1. *The parameterization is rich enough, i.e. there exists a parameter vector $\theta^\star$ such that*

$$Q_{\theta^\star}(s, a) = Q^\star(s, a). \tag{10}$$

For a rich parameterization, we can characterize the optimal parameters as those that minimize the following least squares problem

$$\theta^\star = \arg\min_\theta \ \mathbb{E}\left[\frac{1}{2}(Q^\star(s, a) - Q_\theta(s, a))^2\right]. \tag{11}$$

In practice, the selected parameterization of $Q_\theta(s, a)$ will satisfy Assumption 1 asymptotically as it is added more complexity. Moreover, the least-squares problem in (11) cannot be tackled directly, as the true Q-function $Q^\star(s, a)$ is unknown. A classical approach to Q-learning is therefore trying to approximate the solution of (11) by updating the parameters using the temporal difference (TD) error defined as

$$\delta_k = L(s_k, a_k) + \gamma V_\theta(s_{k+1}) - Q_\theta(s_k, a_k), \tag{12}$$

used to formulate the following parameter update

$$\Delta\theta_Q = \alpha \delta_k \nabla_\theta Q_\theta(s_k, a_k), \tag{13}$$

where $\Delta\theta_Q = \theta_{k+1} - \theta_k$ and $\alpha > 0$ is a scalar denoting the step size. The gradient $\nabla_\theta Q_\theta(s_k, a_k)$ can be obtained from sensitivity analysis of the MPC scheme, as detailed in [3].

An alternative to the incremental update of parameters in (13), is a batch approach to Q-learning. This method is known to result in more stable learning [4]. A batch approach entails introducing an additional set of parameters $\tilde{\theta}$ that is continuously being updated, e.g.

$$\Delta\tilde{\theta} = \alpha \tilde{\delta}_k \nabla_{\tilde{\theta}} Q_{\tilde{\theta}}(s_k, a_k), \tag{14}$$

where $\tilde{\delta}_k = L(s_k, a_k) + \gamma V_{\tilde{\theta}}(s_{k+1}) - Q_{\tilde{\theta}}(s_k, a_k)$ whereas a_k is selected according to a fixed policy obtained from $Q_\theta(s_k, a_k)$. As the updated parameters $\tilde{\theta}$ converge, we may replace the fixed parameters θ with the updated ones, and begin a new batch.

In order to learn the optimal parameters in (11), 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.,

$$\mu_\theta(a|s) = \pi_\theta(s) + \zeta_a, \tag{15}$$

where ζ_a is normally distributed according to $\zeta_a \sim \mathcal{N}(0, \sigma_a^2 I_a)$. Convergence properties for Q-learning are established in e.g. [18] and elaborated further in Section V-B. The stochastic policy in (15) will serve as a baseline for the variance-based exploration scheme 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., [7] and [13]. The referenced work is 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 induce exploration.

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

$$\hat{\theta} = \theta + \zeta_p, \tag{16}$$

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

$$\hat{\pi}_{\hat{\theta}} = \arg\min_a Q_{\hat{\theta}}(s, a). \tag{17}$$

Remark 1. *We note that exploration in parameter space in combination with Q-learning, generally calls for a batch approach to Q-learning as given by (14). For an incremental approach as in (13) where we only have one set of parameters θ, the resulting TD-error as we update the parameters according to (16) would be $L(s_k, a_k) + \gamma V_{\hat{\theta}}(s_{k+1}) - Q_{\hat{\theta}}(s_k, \pi_{\hat{\theta}})$ where $Q_{\hat{\theta}}(s_k, \pi_{\hat{\theta}}) = V_{\hat{\theta}}(s_k)$, 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 scheme in (16) as a second baseline for our proposed method.

IV. Value function IDW variance

With the goal of guiding exploration using the uncertainty of our value function estimate, we need a method 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 [19], radial basis functions (RBFs) were used to formulate a measure of uncertainty based on sampled points. In [20], 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.

A. Inverse distance weighting

IDW is a method for interpolation given a data set, that also allows us to define a variance function given a predictor of the function that is sampled. Let $\eta = (\theta, s)$, where $\eta \in \mathbb{R}^q$. We may consider the following scaling function $\phi : \mathbb{R}^q \to \mathbb{R}^q$ in order to be immune to the different scaling of individual parameters and states

$$\phi(\eta) = \text{diag}\left(\frac{2}{\eta_{\max} - \eta_{\min}}\right)\left(\eta - \frac{\eta_{\max} + \eta_{\min}}{2}\right), \tag{18}$$

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

The IDW framework could 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 argument of the action-value function, and therefore choose to apply IDW to the value function. For convenience we define $V_\theta(s) = \hat{V}(\eta)$, where $V_\theta(s)$ is the parameterized V-function from (5). We then assume that we have samples of the true V-function available and that we have collected a vector $V_t = \{V_{t_i}, \ldots, V_{t_M}\}$ of M samples where $V_{t_i} = V_t(\eta_i)$ at corresponding points $\eta_i, \ldots, \eta_M$. For a new instance of η, we consider the (scaled) squared Euclidean distance function $d^2 : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ given as

$$d^2(\eta, \eta_i) = (\phi(\eta_i) - \phi(\eta))^\top (\phi(\eta_i) - \phi(\eta)), \quad i = 1, \ldots, M. \tag{19}$$

The inverse distance weighting function $w_i : \mathbb{R}^q \to \mathbb{R}$ can then be defined as in [21]

$$w_i(\eta) = \frac{1}{d^2(\eta, \eta_i)}, \tag{20}$$

and assigns larger weights $w_i(\eta)$ to samples that are close to η than to samples further away. In [22], the following alternative weighting function was suggested

$$w_i(\eta) = \frac{e^{-d^2(\eta, \eta_i)}}{d^2(\eta, \eta_i)}. \tag{21}$$

The weighting function in (21) is similar to (20) for small values of d^2, but more quickly reduces the effect of points η_i

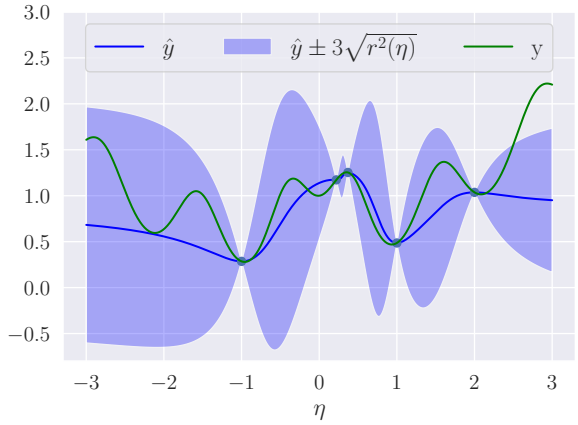

Fig. 1: Example of IDW: Function y (green) and samples (η_k, y_k) (blue dots). Error bands are given using the IDW variance estimate in (24) i.e. $\pm 3\sqrt{r^2(\eta)}$ (shaded blue) evaluated for the predictor $\hat{y}(\eta)$.

far away from η due to the exponential term. We then define the following function $v_i : \mathbb{R}^q \to \mathbb{R}$ as

$$v_i(\eta) = \begin{cases} 1 & \text{if } \eta = \eta_i \\ 0 & \text{if } \eta = \eta_j, j \neq i \\ \frac{w_i(\eta)}{\sum_{j=1}^{M} w_j(\eta)} & \text{otherwise} \end{cases} \tag{22}$$

which allows us to define

$$\bar{V}(\eta) = \sum_{i=1}^{M} v_i(\eta) V_{t_i}, \tag{23}$$

which is an IDW interpolation of $\{(\eta, V_{ti})\}_{i=1}^{M}$. Using the selection function in (22), the function in (23) will be equal to an already sampled function value in case we have η in our data set, if not we will interpolate on the M existing samples. It was shown in [23, Lemma 1], for both choices of the weighting function, (20) and (21), that the interpolation function $\bar{V}(\eta)$ 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 [22]–[24]

$$r^2(\eta) = \sum_{i=1}^{M} v_i(\eta) \big(V_{t_i} - \hat{V}(\eta)\big)^2. \tag{24}$$

Essentially, the IDW variance estimate is a weighted average of the squared error between our sampled points V_{t_i} and our predictor $\hat{V}(\eta)$. As $\hat{V}(\eta)$ 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, in this case, a scalar function y.

B. p-step TD prediction of V

Monte Carlo (MC) learning involves learning from experience, using sequences of states, actions, and rewards. In fact, MC learning offers an alternative method to TD-learning based on (13) for learning the action-value or value function. Whereas TD-learning provides a low variance but high bias estimate of V, an MC estimate of V will have a high variance but a low bias, see, e.g., [4]. Here we propose to use MC learning for providing V-targets to be used with IDW. Using the IDW framework, our variance estimate will be based on a weighted average of the squared difference between these targets and our function approximator. The weights are assigned using inverse distances according to (20), i.e. targets sampled for states and parameters similar to current values in η will influence the variance estimate more.

We now consider a policy $\pi_{\bar{\theta}}$, where $\bar{\theta}$ denotes the parameter vector. For notational convenience we let $\pi_{\bar{\theta}} = \bar{\pi}$. We consider a selected number of p states and actions, i.e.

$$\{s_{k-p}, a_{k-p}, s_{k-p+1}, \ldots, a_{k-1}, s_k\} \sim \bar{\pi} \tag{25}$$

where $\sim$ signifies that states and actions were obtained by acting according to policy $\bar{\pi}$. The data in (25) is used to construct a V-function sample by evaluating the sum of discounted costs. In order to generalize to a continuing task or very long episodes, we bootstrap on our value function estimate for the last sampled state. The V-prediction is given as

$$\begin{aligned} &V_t(\bar{\theta}_k, s_{k-p}) = L(s_{k-p}, a_{k-p}) + \\ &\gamma L(s_{k-p+1}, a_{k-p+1}) + \ldots + \gamma^p V_{\bar{\theta}_k}(s_k), \end{aligned} \tag{26}$$

evaluated for the current parameter vector $\bar{\theta}_k$. The V-estimate provided by (26) can be thought of as a p-step TD prediction of V [4], and p thereby becomes a hyperparameter. Because we bootstrap on our value function, both our targets and predictor in (24) is based on the function approximator in (5). It is therefore important that p is large enough, such that the effect of bootstrapping is small. Moreover, the effect of discounting will contribute to reducing the bias of bootstrapping. However, selecting p large relative to the batch size, means that we obtain few samples of the value function, which will influence the variance estimate. An alternative to the target we proposed in (26), is an exponentially weighted estimate as the authors propose for the advantage function in [25].

Remark 2. *As the variance-based exploration scheme 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 (14). This entails having two sets of parameters, where one set of parameters is continuously updated $\tilde{\theta}$, whereas the other set of parameters $\theta = \bar{\theta}$ is used to define a policy that visits informative states.*

The V-function estimate in (26) is particular for the current policy estimate $\bar{\pi}$, i.e. we are estimating $V^{\bar{\pi}}$. As the parameters are updated from $\bar{\theta}$ to $\bar{\theta}'$ at the beginning of a new batch, data is collected with an updated policy $\bar{\pi}'$, and we are making p-step predictions of $V^{\bar{\pi}'}$. Although the previously sampled V-functions are made for an outdated policy, they may be useful for our purpose, which is to estimate uncertainty w.r.t the parameters. Also, in our choice of weighting function in the IDW variance estimate, e.g. either (20) or (21), we can influence the impact of previously sampled targets on our variance estimate.

Remark 3. *IDW methods are known to be successful and efficient in deterministic settings, but also have some robustness with respect to measurement noise as demonstrated in [20], [24]. This means that the selected framework can also be used for the exploration of stochastic systems.*

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 is able to run fast in real-time, we set a limit for the maximum number of samples $M_{\max}$ used to evaluate (24). If our data set already contains $M_{\max}$ samples, we evaluate the following distance measure for a new instance of η

$$\sum_{i=1}^{M_{\max}} d^2(\eta, \eta_i) > \min\left\{ \sum_{i=1}^{M_{\max}} d^2(\eta_i, \eta_1), \dots, \sum_{i=1}^{M_{\max}} d^2(\eta_i, \eta_{M_{\max}}) \right\}. \tag{27}$$

If the summed distance from a new sample η in (27) to our current samples $\eta_1, \dots, \eta_{M_{\max}}$ 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 dimension, in addition to the state dimension, will also affect the size. The IDW variance (24) 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

The most common method for ensuring exploration during learning in RL is using or adding randomness to actions or parameters. In this section, we will see how we can leverage an IDW framework as detailed in the previous section, in order to direct exploration to where we have uncertainty in our value function estimate. We are then acting according to the strategy of "optimism under 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 therefore propose to use the variance estimate obtained in (24) to perturb the parameters at the beginning of each batch, in a more meaningful way than adding pure random perturbations as in (16). We first define the gradient of the IDW variance function w.r.t. the parameters:

$$\begin{aligned}\nabla_\theta r^2(\eta) = \sum_{i=1}^{M} \nabla_\theta v_i(\eta)(V_{t_i} - \hat{V}(\eta))^2 \\ -2v_i(\eta)(V_{t_i} - \hat{V}(\eta))\nabla_\theta \hat{V}(\eta),\end{aligned} \tag{28}$$

in order to highlight the fact that the sensitivity of the MPC scheme, in terms of $\nabla_\theta \hat{V}(\eta)$, is used in evaluating the gradient of the variance. We propose the following update of the parameters in order to efficiently explore

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

where ζ_p is a noise term as defined for (16) and $\nabla_\theta \hat{r}^2(\theta, s)$ is the bounded gradient of the variance estimate w.r.t. the parameters, obtained using IDW. 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, in order to collect data such that the variance estimate of our function approximator is meaningful. We propose to bound the variance gradient by using the standard deviation used for Gaussian exploration in parameter space, which first of all makes the two methods highly comparable, and also prevents the addition of yet another hyperparameter, e.g.,

$$\nabla_\theta \hat{r}^2(\theta, s) = \text{sat}(\nabla_\theta r^2(\theta, s), -2\sigma_p, 2\sigma_p). \tag{30}$$

A policy estimate based on the perturbed parameters in (29), is obtained according to

$$\pi_{\bar{\theta}}(s_k) = \arg\min Q_{\bar{\theta}}(s_k, a_k). \tag{31}$$

The formulation in (29) and (31) resembles the exploration scheme 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 estimate of variance to guide the sampling. We make the prediction of V for each realization of $\bar{\theta}$ 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 at the beginning of the next batch, in order to generate a new (perturbed) parameter vector to be used. This is summarized in Algorithm 1.

B. Convergence properties

Q-learning will converge under the assumption of satisfying two conditions: (1) Greedy in the limit with infinite exploration (GLIE), (2) the step-sizes α_k satisfy the Robbins-Munro sequence. For more details, the reader is referred to [18]. The GLIE condition entails that all state-action pairs are explored infinitely many times, and that, as time goes to infinity, the

Algorithm 1: Variance-based exploration

Input : Initial MPC parameters θ_0, initial learning parameters $\tilde{\theta}_0$, initial state s_0, data set $\mathcal{D}$, batch update frequency b, learning step size α, number of samples used to generate V-target p, maximum number of samples in data set M_{max}, parameter noise distribution σ_p;
Output : Optimal policy π_θ

while $k \leq k_{max}$ **do**
  if *mod*$(k, b) = 0$ **then**
    Update MPC parameters with learned parameters $\theta_k = \tilde{\theta}_k$
    Perturb parameters to get $\bar{\theta}_k$ according to (29) based on $\mathcal{D}$
    Ensure that the IDW variance gradient respects bound (30)
  end
  Act greedily w.r.t. current Q-estimate (31)
  if *mod*$(k, b) \geq p$ **then**
    if $|\mathcal{D}| \leq M_{max}$ *or* $(|\mathcal{D}| \geq M_{max}$ *and (27))* **then**
      Calculate p-step prediction of V_{t_k} from (26)
      Add $\{V_{t_k}, s_k, a_k, \bar{\theta}_k\}$ to $\mathcal{D}$
    end
  end
  Update $\tilde{\theta}_k$ according to (14)
  $k \leftarrow k + 1$
end

policy converges to the optimal policy. The second condition (2) is not directly related to the exploration scheme and is most commonly satisfied in practice by the selection of small constant step sizes. Formally, we can ensure GLIE for the proposed exploration scheme in Section (V-A). The first part of GLIE is ensured by keeping a random term in (29) so that we ensure sufficient exploration even though the gradient of variance eventually may converge to a small number. The second part of GLIE can be ensured by using a decaying scalar i.e. $\beta(\nabla_{\theta_k}\hat{r}^2(\theta_k, s_k) + \frac{1}{2}\zeta_p)$ where $\beta = \beta_a \exp(-\omega k)$ and ω is a hyperparameter.

VI. Simulation examples

We apply the proposed exploration scheme to two simulation examples, namely on an LQR problem and a cart pendulum 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 adopted from [26]. We consider a discrete linear system of the form

$$s_{k+1} = As_k + Ba_k, \tag{32}$$

with system matrices

$$A = \kappa \begin{bmatrix} \cos\beta & \sin\beta \\ \sin\beta & \cos\beta \end{bmatrix}, \; B = \begin{bmatrix} 1.1 & 0 \\ 0 & 0.9 \end{bmatrix}, \tag{33}$$

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

$$L(s, a) = \frac{1}{2}\|s - s_{\text{ref}}\|^2 + \frac{1}{2}\|a - a_{\text{ref}}\|^2, \tag{34}$$

where $s_{\text{ref}} = [0.1, 0.1]^\top$, and the reference input is found accordingly. The prediction model is defined as

$$A_0 = \kappa \begin{bmatrix} \cos\hat{\beta} & \sin\hat{\beta} \\ \sin\hat{\beta} & \cos\hat{\beta} \end{bmatrix}, \; B_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{35}$$

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

$$\min_{x,u} \; V_0 + \gamma^N \|x_N - x_{\text{ref}}\|_P^2 + \sum_{k=0}^{N-1} \gamma^k \left\| \begin{bmatrix} x_k - x_{\text{ref}} \\ u_k - u_{\text{ref}} \end{bmatrix} \right\|^2 \tag{36a}$$

$$\text{s.t } x_{k+1} = A_0 x_k + B_0 u_k, \quad x_0 = s, \tag{36b}$$

using a prediction horizon of $N = 10$ and P is the solution to the discrete Riccati equation obtained using the inaccurate system dynamics in (35). We let the parameter vector be $\theta = \{x_{\text{ref}}, u_{\text{ref}}, V_0\}$ and use Q-learning with a learning rate of $\alpha = 0.1$. We consider a continuing task and let $\gamma = 0.99$. We simulate the system for a total of 5000 time steps, and use Q-learning in batches of length 200, i.e., we update the parameters in the MPC scheme every 200 time steps. For each exploration scheme, we consider a range of noise distributions. For brevity, only the best-performing ones are reported. The system states and actions are plotted both during exploration and exploitation, i.e., we use the learned parameters to simulate the system, for Gaussian action noise and variance-based exploration, see Figure 3. We also plot the norm of the parameter updates resulting from the different exploration schemes, in order to give an indication of 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 I. The numbers reported in Table I are found for a total of 5 simulations run in each category.

For the variance-based exploration scheme, we use $p = 10$, do not pose any restrictions on the data sampled, i.e., $M_{\text{max}} = 5000$ and use the weighting function in (21). The resulting IDW variance estimate (24) is plotted over learning batches in Figure 2. We see that the mean of the variance 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 I, 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 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

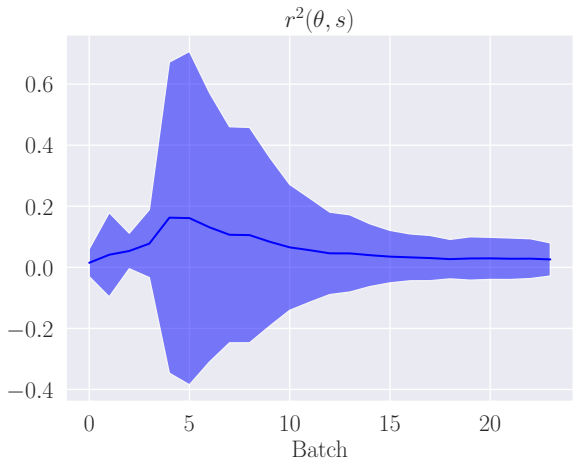

Fig. 2: The mean and two standard deviations of the IDW variance estimate (24) over learning batches.

exploitation compared to Gaussian action noise, although the same as 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 I.

B. Cart pendulum

We consider the cart-pendulum system as depicted in Figure 4.

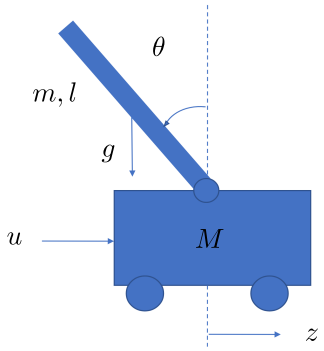

Fig. 4: Cart-pendulum system, with model parameters as specified in Table III.

The dynamics of the cart pendulum, neglecting friction, are given by

$$(M+m)\ddot{z} + \frac{1}{2}ml\ddot{\phi}\cos\phi = \frac{1}{2}ml\dot{\phi}^2\sin\phi + u, \tag{37a}$$

$$\frac{1}{3}ml^2\ddot{\phi} + \frac{1}{2}ml\ddot{x}\cos\phi = -\frac{1}{2}mgl\sin\phi, \tag{37b}$$

where z and $\dot{z}$ are the cart position and velocity respectively, and similarly ϕ and $\dot{\phi}$ are the pendulum's angle from the vertical axis and angular velocity. The force u is the control input and g is the acceleration of gravity. Hence, the state is $s = [z, \dot{z}, \phi, \dot{\phi}]^\top$ and the action $a = u$. We use the model parameters as listed in Table III. The dynamics in (37) 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. [27].

A 4^{th}-order Runge Kutta scheme is used to discretize the dynamics in (37), using the step size dt $= 0.1$ s. We consider the following constraint, in newtons, on the force acting on the cart

$$-7 \leq a \leq 7. \tag{38}$$

Moreover, we use a discounting factor of $\gamma = 0.99$ and use the following RL cost

$$L(s,a) = \begin{bmatrix} s - s_{\text{ref}} \\ a \end{bmatrix}^\top \begin{bmatrix} I_4 & 0 \\ 0 & 0.01 \end{bmatrix} \begin{bmatrix} s - s_{\text{ref}} \\ a \end{bmatrix}, \tag{39}$$

where $s_{\text{ref}} = [0.5, 0, 0, 0]^\top$. The linearized prediction model in the MPC scheme, defined by $\bar{A}$ and $\bar{B}$, is obtained from linearizing the system dynamics (37) at $\phi = 0$, corresponding to the pendulum being in an upright position. The parameterized MPC scheme reads as

$$\min_{\hat{s},u} \; V_0 + \gamma^N \|\hat{s}_N - \hat{s}_{\text{ref}}\|_P^2 + \sum_{k=0}^{N-1} \gamma^k \ell_\theta(\hat{s}, u) \tag{40a}$$

$$\text{s.t } \hat{s}_{k+1} = \bar{A}\hat{s}_k + \bar{B}u_k, \quad x_0 = s, \tag{40b}$$

$$-7 \leq u_k \leq 7, \tag{40c}$$

where $\hat{s}$ is the predicted state, P is found using $\bar{A}, \bar{B}$ and $N = 30$. We assume that we know all state references, except the cart position, i.e. $\hat{s}_{\text{ref}} = [\theta_c, 0, 0, 0]$. Moreover, we parameterize the stage cost using a quadratic function according to

$$\ell_\theta(\hat{s}, u) = \begin{bmatrix} \hat{s} - \hat{s}_{\text{ref}} \\ u \end{bmatrix}^\top M(\theta) \begin{bmatrix} \hat{s} - \hat{s}_{\text{ref}} \\ u \end{bmatrix}, \tag{41}$$

where $M(\theta)$ is a positive definite matrix. The parameter vector can be written as, with some abuse of notation, $\theta = \{V_0, M(\theta), \theta_c\}$. 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 schemes, for an interval of Gaussian distributions, and report the best-performing distribution in each category. For the variance-based exploration scheme, we use $p = 110$, $M_{\text{max}} = 5000$, and the weighting function in (21) .

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 II 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. We see from 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

TABLE I: Cost statistics for LQR simulations. The mean and standard deviation (in parentheses) are found for a total of 5 simulations in each category.

Exploration method	Exploration			Exploitation	
	σ	k	$\sum L(s,a)$	k	$\sum L(s,a)$
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)**

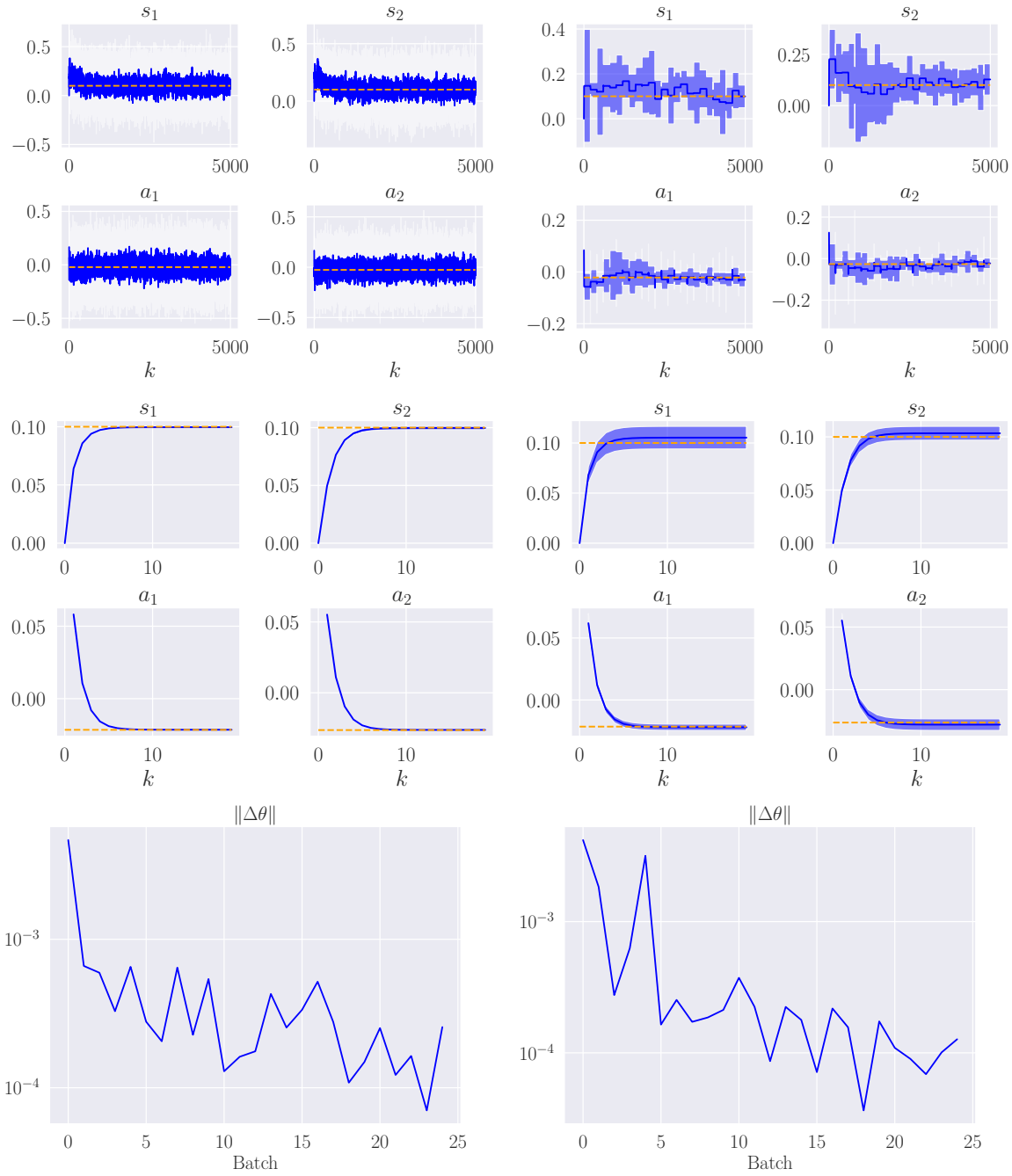

Fig. 3: LQR simulation results. **Upper plots**: the mean and two standard deviations of states and actions during exploration, with Gaussian action noise (left) and variance-based exploration (right). **Middle plots**: the mean and two standard deviations of states and actions during exploitation, using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

TABLE II: Cost statistics for cart pendulum simulations. The mean and standard deviation (in parentheses) are found for a total of 3 simulations in each category.

Exploration method	Exploration			Exploitation	
	σ	k	$\sum L(s,a)$	$k_{\max}$	$\sum L(s,a)$
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)

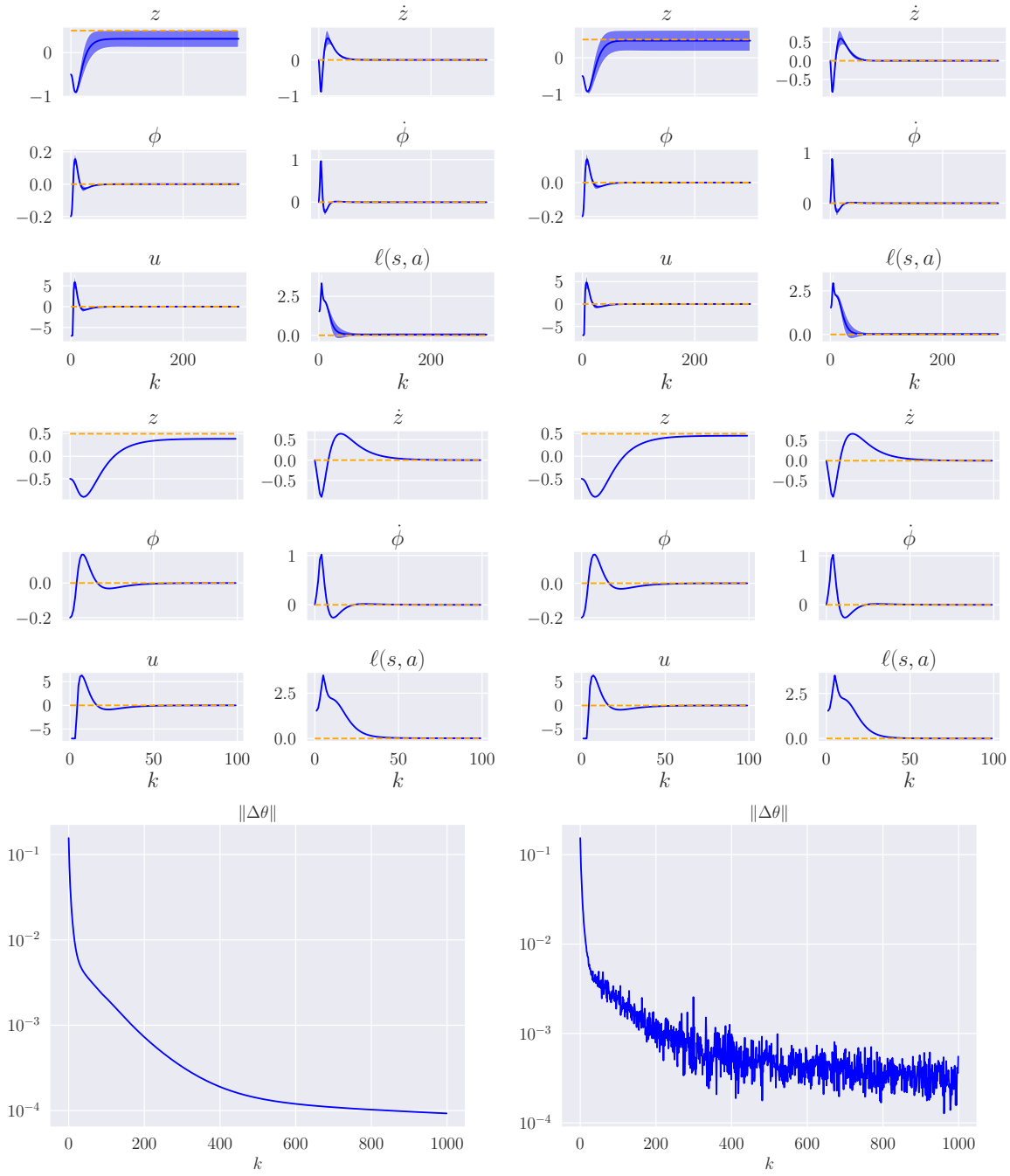

Fig. 5: Cart pendulum simulation results. **Upper plots**: the mean and two standard deviations of states and actions during exploration, with Gaussian action noise (left) and variance-based exploration (right). **Middle plots**: the mean and two standard deviations of states and actions during exploitation, 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).

visible improvement in both the plotted cart position as well as the calculated cost of exploitation in Table II.

The statistics in Table II are found for a total of 3 simulations in each category. For Gaussian exploration in action space, we see that very little noise is needed in order to obtain the best performance in exploitation, namely $\sigma_a = 0.0001$. By using a Gaussian perturbation in parameter space with a larger standard deviation we are actually able to improve the performance in exploitation, while also reducing the cost of learning. Combining Gaussian parameter noise with the variance gradient, we are able to make exploration even cheaper while achieving a similar improvement in performance.

VII. Conclusion

We have presented a first attempt at variance-based exploration particularly suited for using a model predictive control (MPC) scheme as a function approximator in RL. The method is based on inverse distance weighting (IDW) to build a variance estimate of the V-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 scheme 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 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.

Appendix

TABLE III: Cart 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

Acknowledgment

The authors gratefully acknowledge the support by the industry partners Borregaard, Elkem, Hydro, Yara and the Research Council of Norway through the project Towards Autonomy in Process Industries (TAPI), project number 294544, and the project Safe reinforcement learning using MPC (SARLEM), project number 300172.

References

[1] P. Abbeel, A. Coates, M. Quigley, and A. Ng, "An application of reinforcement learning to aerobatic helicopter flight," *Advances in neural information processing systems*, vol. 19, 2006.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 636–648, 2020.

[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[5] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," *Advances in neural information processing systems*, vol. 30, 2017.

[6] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[7] 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," *International Conference on Learning Representations*, 2018.

[8] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Machine learning*, vol. 49, pp. 209–232, 2002.

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

[10] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3-4, pp. 285–294, 1933.

[11] I. Osband, B. Van Roy, and Z. Wen, "Generalization and exploration via randomized value functions," in *International Conference on Machine Learning*. PMLR, 2016, pp. 2377–2386.

[12] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," *Advances in neural information processing systems*, vol. 29, 2016.

[13] 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 *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rywHCPkAW

[14] R. Y. Chen, J. Schulman, P. Abbeel, and S. Sidor, "UCB and infogain exploration via Q-ensembles," *arXiv preprint arXiv:1706.01502*, vol. 9, 2017.

[15] N. Nikolov, J. Kirschner, F. Berkenkamp, and A. Krause, "Information-directed exploration for deep reinforcement learning," *International Conference on Learning Representations*, 2019.

[16] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," *International Conference on Machine Learning*, 2018.

[17] E. C. Kerrigan and J. M. Maciejowski, "Soft constraints and exact penalty functions in model predictive control," *Proceedings of the UKACC International Conference on Control*, 2000.

[18] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Machine learning*, vol. 16, no. 3, pp. 185–202, 1994.

[19] H.-M. Gutmann, "A radial basis function method for global optimization," *Journal of global optimization*, vol. 19, no. 3, pp. 201–227, 2001.

[20] A. Bemporad, "Global optimization via inverse distance weighting and radial basis functions," *Computational Optimization and Applications*, vol. 77, no. 2, pp. 571–595, 2020.

[21] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM national conference*, 1968, pp. 517–524.

[22] V. R. Joseph and L. Kang, "Regression-based inverse distance weighting with applications to computer experiments," *Technometrics*, vol. 53, no. 3, pp. 254–265, 2011.

[23] A. Bemporad, "Global optimization via inverse distance weighting and radial basis functions," *Computational Optimization and Applications*, vol. 77, no. 2, pp. 571–595, 2020.

[24] ——, "Active learning for regression by inverse distance weighting," *Information Sciences*, 2023, in press. Also available on http://arxiv.org/abs/2204.07177. Code: http://cse.lab.imtlucca.it/ bemporad/ideal.

[25] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *International Conference on Learning Representations*, vol. abs/1506.02438, 2015.
[26] S. Gros and M. Zanon, "Towards safe reinforcement learning using NMPC and policy gradients: Part I-stochastic case," *arXiv preprint arXiv:1906.04057*, 2019.
[27] S. L. Brunton and J. N. Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.