2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Combining Q-learning and Deterministic Policy Gradient for Learning-based MPC

Katrine Seel, Sébastien Gros, Jan Tommy Gravdahl

***Abstract*—This paper considers adjusting a fully parametrized model predictive control (MPC) scheme to approximate the optimal policy for a system as accurately as possible. By adopting MPC as a function approximator in reinforcement learning (RL), the MPC parameters can be adjusted using Q-learning or policy gradient methods. However, each method has its own specific shortcomings when used alone. Indeed, Q-learning does not exploit information about the policy gradient and therefore may fail to capture the optimal policy, while policy gradient methods miss any cost function corrections not affecting the policy directly. The former is a general problem, whereas the latter is an issue when dealing with economic problems specifically. Moreover, it is notoriously difficult to perform second-order steps in the context of policy gradient methods while it is straightforward in the context of Q-learning. This calls for an organic combination of these learning algorithms, in order to fully exploit the MPC parameterization as well as speed up convergence in learning.**

***Index Terms*—Model predictive control, Deterministic policy gradient method, 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, sampled state transitions and observed costs can be used to improve the performance of a control policy. RL has drawn increasing attention due to its accomplishments for robotics and games, see e.g. [1] and [2]. However, as deep neural networks (DNNs) are typically used as function approximators to capture the policy, it is hard to provide guarantees regarding the resulting closed-loop behavior.

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, mainly to deal with systems that are difficult to model. Because these learning-based controllers inherit the closed-loop behaviors of MPC, these are easier to analyze, see e.g. [3], [4] and [5]. In [6], the authors suggest using parameterized model predictive control (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.

Updating the parameters of MPC schemes to improve the closed-loop performance has successfully been tested using Q-learning, see e.g. [7], and deterministic policy gradient algorithms (DPG), see e.g. [8]. While simple to use, Q-learning methods do not come with formal guarantees regarding the closed-loop optimality of the resulting policy [9]. Whereas policy gradient methods come with such (local) guarantees but do not fully exploit the MPC parameterization in learning the policy. Exploiting the parametrization fully is crucial when using RL to verify *dissipativity* for economic problems. Dissipativity is verified by the existence of an appropriate *storage function*, which is generally difficult to find, but that can be learned using Q-learning as proposed in [10]. In this paper, we detail how to combine these RL algorithms for MPC, such that their respective drawbacks are tackled.

To the best of the authors' knowledge, Q-learning and policy gradient methods have to a small extent been combined to formulate parameter updates. In [11], Q-learning and DPG parameter updates were combined in the context of RL-based MPC using a null space projection to alleviate the difficulties experienced when using the methods independently. An issue with this combination is the potential inaccuracies in the null space computation related to small, but non-zero, eigenvalues. In [12], the authors propose to combine the parameter update for a regularized policy gradient technique with that of Q-learning. The authors provide empirical evidence of the combined update scheme resulting in improved data efficiency and stability. The idea of combining parameter updates from different RL methods is the same as in this paper but the resulting combined parameter update is different. Also, the authors in [12] use a DNN as a function approximator, and propose an augmentation to the standard architecture, to facilitate learning of both the policy as well as the action-value function. When using MPC as a function approximator, both the policy, as well as the action-value function, can be approximated, by adding just one additional constraint.

Classical Q-learning and policy gradient methods typically use first-order methods to update the policy and value function parameters. Second-order methods tend to yield a much faster convergence. The natural policy gradient method is based on the Fisher information matrix to provide a policy Hessian approximation, and approximate second-order steps [13]. However, the convergence rate is affected by the quality of that Hessian approximation. For Q-learning on the other hand, the Hessian is straightforward to obtain. Therefore, we propose to use the Q-learning Hessian to improve the second-order step of policy gradient methods.

In this paper, the first contribution is a combination of RL methods using a multi-objective approach. Secondly, we

979-8-3503-0123-6/23/$31.00 ©2023 IEEE

propose a second-order method based on the multi-objective approach, to speed up convergence in learning. We demonstrate the performance of the combination of RL methods in simulation and compare the convergence rate with the natural policy gradient and a second-order Q-learning method.

The paper is organized into five sections. Section 2 gives a brief introduction to MDPs and the use of MPC as a function approximator in RL, as well as methods that can be used to learn the parameters. Section 3 presents a multi-objective approach to combining RL algorithms. Two simulated examples are included in Section 4, and finally, Section 5 concludes the paper.

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. the statistics of future states depend only on the current state. We denote the underlying conditional transition probability density as p, defined for the state $s \in \mathcal{S} \subseteq \mathbb{R}^n$ and action $a \in \mathcal{A} \subseteq \mathbb{R}^m$. The function p gives the probability density of transitioning to state s_{k+1} given action a_k in state s_k i.e.

$$s_{k+1} \sim p(\cdot|s_k, a_k), \tag{1}$$

where s_{k+1} denotes the next state and k denotes the physical time of the system. We assume that a stage cost, $L(s_k, a_k)$, is provided. Our goal is to find the parameters θ of a deterministic policy $\pi(s)$, that maps from state to action i.e. $\pi_\theta : \mathcal{S} \to \mathcal{A}$, so as to minimize the sum of discounted cost

$$J(\pi_\theta) = \mathop{\mathbb{E}}_{\substack{s_0 \sim p_0, \\ s \sim p(\cdot|s,\pi_\theta(s))}} \left[\sum_{k=0}^{K} \gamma^k L(s_k, a_k) \;\middle|\; a_k = \pi_\theta(s_k) \right], \tag{2}$$

where p_0 is the distribution of initial states and $\gamma \in (0, 1]$ is a discount factor used to establish the importance of future costs over immediate costs. In the following, we will use $\mathbb{E}_{\pi_\theta}[\cdot]$ to denote the expectation of the Markov chain in closed-loop with policy π_θ. The value function V_{π_θ} and action-value function Q_{π_θ} are defined as

$$Q_{\pi_\theta}(s, a) = L(s, a) + \gamma \mathbb{E}_{s^+ \sim p(\cdot|s,a)} \left[V_{\pi_\theta}(s^+) \middle| s, a \right], \tag{3a}$$

$$V_{\pi_\theta}(s) = Q_{\pi_\theta}(s, \pi_\theta(s)) \tag{3b}$$

where s^+ denotes the subsequent state.

A. MPC as a function approximator

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

$$V_\theta(s) = \min_{x,u,\sigma} \; -\lambda_\theta(s) + \gamma^N T_\theta(x_N) + \sum_{i=0}^{N-1} \gamma^i \ell_\theta(x_i, u_i) \tag{4a}$$

$$\text{s.t} \quad \forall i \in \mathbb{I}_{0:N-1} : x_{i+1} = f_\theta(x_i, u_i), \tag{4b}$$

$$h_\theta(x_i, u_i) \leq 0, \; h_\theta^N(x_N) \leq 0, \tag{4c}$$

$$u_i \in \mathcal{A}, \qquad x_0 = s, \tag{4d}$$

where $x = \{x_0, \ldots, x_N\}$ and $u = \{u_0, \ldots, u_{N-1}\}$. We note that we use x and u to distinguish the predicted state and input sequences from the actual state and input, s and a. The objective consists of a parameterized stage cost $\ell_\theta(x, u)$, a parameterized terminal cost $T_\theta(x)$, and a storage function $\lambda_\theta(s)$. The terminal cost compensates for the fact that our MPC scheme considers a finite horizon N, that may be shorter than the (possibly infinite) horizon K in the performance measure (2). The storage function is primarily useful in the case of learning stable policies for economic problems, for which the economic stage cost in general is not positive definite. For problems that are *dissipative*, the storage function allows us to reformulate the cost, such that nominal stability can be proved, while the solution to the MPC remains unchanged. In this case, the learned stage cost $\ell_\theta(x, u)$ is constrained to be lower-bounded by $\mathcal{K}_\infty$. For more details, the reader is referred to e.g. [14]. The function $f_\theta(x, u)$ models the system dynamics. In case of model mismatch, the constraints $h_\theta(x, u)$ and $h_\theta^N(x)$ may be complemented by slack variables, or implemented as soft constraints in the stage cost. For robust constraint satisfaction in the face of model error, see [7].

The deterministic policy is given by the first element in the input sequence which is the solution to (4), i.e.

$$\pi_\theta(s) = u_0^\star(s, \theta). \tag{5}$$

In order to introduce exploration, we add a small disturbance to the policy, i.e.

$$\varphi_\theta(a|s) = \pi_\theta(s) + \zeta_a, \tag{6}$$

with $\zeta_a \sim \mathcal{N}(0, \sigma_a^2 I_m)$, where σ_a is the standard deviation and I_m is the identity matrix. To ensure that the actions from (6) respect the input constraints, we project violating actions back to the feasible space. As shown in [15], this will only mildly bias the policy gradient estimation when using DPG methods. In the following, we will consider two types of methods that can be used to adjust the parameters θ.

B. Q-learning

Q-learning is an RL method that aims to learn the optimal action-value function in (3a). An estimate of the Q-function is obtained by constraining the first action in the input sequence in the MPC scheme, according to

$$Q_\theta(s, a) = \min_{x,u,\sigma} \quad (4a), \tag{7a}$$

$$\text{s.t} \quad (4b) - (4d), \tag{7b}$$

$$u_0 = a. \tag{7c}$$

It can be shown that the Q-function estimate from (7), the value function estimate (4), and the policy (5) satisfies the Bellman equations [6]. We make the following assumption for our parameterization.

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

$$Q_{\theta^\star}(s, a) = Q^\star(s, a). \tag{8}$$

Remark 1. *Assumption 1 is strong, although common in theoretical RL that makes use of function approximators. Making the parameterization rich, entails using universal function approximators for the cost terms and constraints in the MPC scheme. If the parameterization is not rich, RL will find the best parameters among the set of functions provided by the selected parameterization, see e.g. [16]. The need for using universal function approximators such as NNs is arguably problem-dependent, but as shown in [11], simpler parameterizations such as e.g. a quadratic cost parametrization, may be sufficient to improve closed-loop performance considerably.*

In a Q-learning setting, the policy in (5), is obtained from the action-value function according to

$$\pi_\theta(s) = \arg\min_a Q_\theta(s,a). \tag{9}$$

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

$$\theta^\star = \arg\min_\theta \ \mathbb{E}_{\pi_\theta}\left[\frac{1}{2}(Q^\star(s,a) - Q_\theta(s,a))^2\right]. \tag{10}$$

As the optimal action-value function is in general unknown, (10) cannot be solved directly. Next, we will discuss both a first-order and second-order method for Q-learning. First-order methods use gradient information, whereas second-order methods in addition to the gradient also use second-order information to converge faster to the optimum.

1) First-order Q-learning: A classical approach to Q-learning is trying to achieve (10) by updating the parameters using the temporal difference (TD) error defined as $\delta_k = y_k - Q_\theta(s_k,a_k)$ where $y_k = L(s_k,a_k) + \gamma V_\theta(s_{k+1})$. Here y can be construed as a fixed target, evaluated using a sampled state transition and the cost. The parameter updates are driven by minimizing the TD error, i.e.

$$\min_\theta \ \mathbb{E}_{\pi_\theta}\left[\frac{1}{2}(y - Q_\theta(s,a))^2\right], \tag{11}$$

where y is considered independent of θ. To solve (11) in practice, we need to collect data also when deviating from policy π_θ, i.e. explore, e.g. using the policy in (6). For the minimization problem in (11), we define the following first-order (semi)-gradient step [9]

$$\Delta\theta_Q = \alpha_q \delta_k \nabla_\theta Q_\theta(s_k,a_k), \tag{12}$$

where $\Delta\theta_Q = \theta_{k+1} - \theta_k$, $\nabla_\theta Q_\theta(s_k,a_k)|_{\theta=\theta_k}$ and $\alpha_q > 0$ is a scalar denoting the step size. The gradient $\nabla_\theta Q_\theta(s_k,a_k)$ is obtained from sensitivity analysis. For more details on sensitivity analysis of MPC for Q-learning, the reader is referred to [6].

2) Second-order Q-learning: Rather than considering the TD error at each step as in (12), we can consider the sum of TD errors over a *batch* of state transitions, known as a least-squares TD (LSTD) algorithm [17]. We formulate a second-order LSTD algorithm for Q-learning (LSTDQ), by considering a root-finding problem, using the gradient in (12), i.e.

$$\mathbb{E}_{\pi_\theta}\left[\delta \nabla_\theta Q_\theta(s,a)\right] = 0. \tag{13}$$

For the root-finding problem in (13), we adopt the following Newton step i.e.

$$\Delta\theta_Q^H = -\alpha_d A^{-1} b, \tag{14}$$

using $\Delta\theta$ and $\Delta\theta^H$ to distinguish the first-order and second-order steps, respectively, and $\alpha_d > 0$ to denote the learning rate. The parameter update is given by

$$A = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \delta \nabla_\theta Q_\theta(s,a)^\top + \delta \nabla_\theta^2 Q_\theta(s,a)\right], \tag{15a}$$

$$b = \mathbb{E}_{\pi_\theta}\left[\delta \nabla_\theta Q_\theta(s,a)\right], \tag{15b}$$

where

$$\nabla_\theta \delta = \gamma \nabla_\theta V_\theta(s^+) - \nabla_\theta Q_\theta(s,a), \tag{16}$$

with $\nabla_\theta Q_\theta(s,a)$ and $\nabla_\theta^2 Q_\theta(s,a)$ obtained from sensitivity analysis of the MPC in (7). We note that for a well-posed update in (14), A is negative definite, and hence generalizes to the first-order step in (12) by replacing A with negative identity. The expectations in (15a) are evaluated in an episodic manner, by considering m episodes of length K, i.e.

$$A = \frac{1}{m}\sum_{j=1}^{m}\sum_{k=1}^{K}\Big[\nabla_\theta \delta_k \nabla_\theta Q_\theta(s_{k,j},a_{k,j})^\top + \delta_k \nabla_\theta^2 Q_\theta(s_{k,j},a_{k,j})\Big], \tag{17}$$

$$b = \frac{1}{m}\sum_{j=1}^{m}\sum_{k=1}^{K}\delta_{k,j}\nabla_\theta Q_\theta(s_{k,j},a_{k,j}). \tag{18}$$

For Q-learning techniques, it should be mentioned that there is no guarantee to find the optimal policy. This is because the parameter update of Q-learning methods is not designed to optimize closed-loop performance directly. Instead, Q-learning aims to fit Q_θ as closely as possible to $Q^\star$, and assumes that $Q_\theta \approx Q^\star$ results in $\pi_\theta \approx \pi^\star$. However, there are no guarantees that the former approximation implies the latter, and for certain shapes of Q-functions, it can be challenging to determine the optimal policy, even with an almost correct Q-function estimate. In this scenario, policy-based methods such as DPG methods tend to be more suited.

C. Deterministic policy gradient

The lack of convergence guarantees for Q-learning methods has motivated the need for alternatives such as policy gradient methods with more formal (local) convergence guarantees [18]. Using policy gradient methods, the parameters are updated to directly improve the performance of the policy. In the following, we focus on methods for deterministic policies.

For a rich parameterization, the optimal parameters $\theta^\star$ are characterized by

$$\theta^\star = \arg\min_\theta J(\pi_\theta). \tag{19}$$

1) First-order DPG: Policy gradient methods typically solve (19) using gradient descent, which results in the following first-order parameter update

$$\Delta\theta_J = -\alpha_p \nabla_\theta J(\pi_\theta), \tag{20}$$

where $\alpha_p > 0$ is the learning rate and $\nabla_\theta J(\pi_\theta)|_{\theta=\theta_k}$. For a DPG method, we use the following expression for the policy gradient [19]

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \pi_\theta(s) \nabla_a Q_{\pi_\theta}(s,a)|_{a=\pi_\theta(s)}\right], \tag{21}$$

where $Q_{\pi_\theta}(s,a)$ is the true action-value function for the policy π_θ as defined in (3a). Instead of using the true action-value function, which is generally not known, we will replace it with a function approximator $Q_w(s,a)$, with parameter vector w. In general, assuming that $Q_{\pi_\theta}(s,a) \approx Q_w(s,a)$ will introduce a bias in the policy gradient estimate. However, under certain conditions, as outlined in [19], this approximation can be made without affecting the policy gradient, and in this case, we denote the function approximator as *compatible*. The function approximator of Q_{π_θ} can e.g. take the form

$$Q_w(s,a) = \underbrace{(a-\pi_\theta(s))^\top \nabla_\theta \pi_\theta(s)^\top}_{\Psi(s,a)^\top} w + V_v(s), \tag{22}$$

where $\Psi(s,a)$ is a state-action feature vector, using the following value function approximation

$$V_v(s) = \Phi(s)^\top v, \tag{23}$$

where $\Phi(s)$ is a state feature vector to be selected and v is a parameter vector. The gradient of the Q-function can then be approximated as

$$\nabla_a Q_{\pi_\theta}(s,a) \approx \nabla_a Q_w(s,a) = \nabla_\theta \pi_\theta(s)^\top w. \tag{24}$$

The parameters v and w are then found using an LSTD approach [20], i.e.

$$v = \frac{1}{m}\sum_{j=0}^{m}\left\{\left[\sum_{k=0}^{K}\left[\Phi(s_{k,j})(\Phi s_{k,j} - \gamma\Phi(s_{k+1,j}))^\top\right]\right]^{-1} \sum_{k=1}^{K}\left[\Phi(s_{k,j})L(s_{k,j},a_{k,j})\right]\right\}, \tag{25}$$

$$w = \frac{1}{m}\sum_{j=0}^{m}\left\{\left[\sum_{k=0}^{K}\left[\Psi(s_{k,j},a_{k,j})\Psi(s_{k,j},a_{k,j})^\top\right]\right]^{-1} \sum_{k=1}^{K}\left[(L(s_{k,j},a_{k,j}) + \gamma V_v(s_{k+1,j}) - V_v(s_{k,j}))\Psi(s_{k,j},a_{k,j})\right]\right\}. \tag{26}$$

2) Second-order DPG: For the policy gradient objective in (19), we can also formulate a second-order Newton step i.e.

$$\Delta\theta_J^H = -\alpha_r \nabla_\theta^2 J(\pi_\theta)^{-1} \nabla_\theta J(\pi_\theta), \tag{27}$$

with learning rate $\alpha_r > 0$. An analytic expression of the deterministic policy Hessian is derived in [21], revealing that it is difficult to estimate from data. The Hessian can be replaced by the Fisher information matrix, resulting in a parameter update known as the *natural policy gradient method* [13]. The Fisher information matrix for deterministic policies is defined as

$$F(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \pi_\theta \, \nabla_\theta \pi_\theta^\top\right], \tag{28}$$

and we note that this is only an approximation of the Hessian. Because of the Hessian approximation, the natural policy gradient method still has a linear rate of convergence, despite being a second-order method [22]. This is the same rate of convergence as the first-order DPG method.

III. Combining RL methods

In the following, we propose to combine Q-learning and DPG methods, to learn a parameterization that optimizes both the Q-learning objective (10) and policy gradient objective (19) simultaneously. We propose to do so using multi-objective optimization. Multi-objective optimization is applied in many fields where optimal solutions are needed in the presence of trade-offs between two or more conflicting objectives. As opposed to most multi-objective problems, the objective of Q-learning and policy gradient methods are not necessarily in conflict. However, as we can not minimize the true action-value error directly, and may suffer from limitations related to the richness of our function approximator, the parameter update resulting from each RL method may be in conflict. This suggests that an alternative to the naive sum of update laws is needed.

A. Multi-objective RL

With the purpose of combining Q-learning and DPG methods, we define the following multi-objective problem

$$\min_\theta \quad \omega \cdot \mathbb{E}_{\pi_\theta}\left[\frac{1}{2}(Q^\star(s,a) - Q_\theta(s,a))^2\right] + J(\pi_\theta), \tag{29}$$

where ω is a scalar that weighs the importance of the Q-learning objective relative to the policy gradient objective. In (29) we propose a weighted sum method in order to convert the multi-objective problem to a single-objective problem. This method is appealing because it is simple, with the disadvantage that in practice we introduce an additional hyperparameter that must be tuned. However, the richer the parameterization, the less the tuning of ω will influence the solution to (29). Alternative methods for solving the multi-objective problem are discussed in e.g. [23].

Theorem 1. *Under Assumption 1, and a stability condition for the system under the optimal policy, there exists a parameter vector $\theta^\star$, such that the following holds for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$:*

1) $V_{\theta^\star}(s) = V^\star(s)$
2) $\pi_{\theta^\star}(s) = \pi^\star(s)$
3) $Q_{\theta^\star}(s,a) = Q^\star(s,a)$

Proof. See proof of Theorem 1 in [6]. □

Theorem 1 implies that for a given MDP, an MPC scheme with a possibly inaccurate model can deliver the optimal value function, action-value function, and policy for an appropriate parameterization θ. Building on Theorem 1, we state the following Corollary, which is an important statement for the multi-objective RL approach.

Corollary 1. *Under Assumption 1 there is no trade-off between the Q-learning and policy gradient objective in (29). The minimum of the two objectives will coincide, independent of the scalar value ω.*

Proof. For a rich parameterization, we have for the optimal parameters $\theta^\star$ that $Q_{\theta^\star}(s,a) = Q^\star(s,a)$, hence minimizing the Q-learning objective. Moreover, the optimal parameters $\theta^\star$ also minimize the performance $J(\pi_\theta)$ as stated in (19). □

As pointed out earlier, we can not directly address the Q-learning objective in (29). We therefore adopt the TD approach to Q-learning, as outlined in Section II-B. The multi-objective problem is then

$$\min_{\theta} \quad \omega \cdot \mathbb{E}_{\pi_\theta}\left[\frac{1}{2}(y - Q_\theta(s,a))^2\right] + J(\pi_\theta). \tag{30}$$

The parameter updates as defined in Section II-B and II-C, can then be used to define a Newton step towards the solution of (30). The resulting parameter update is given as the solution to

$$\begin{aligned}\min_{\Delta\theta_m^{H'}} \frac{1}{2}\Delta\theta_m^{H'\top}(-\omega A + F(\theta) + \tau I)\Delta\theta_m^{H'} \\ +\alpha_m(\nabla_\theta J(\pi_\theta) - \omega b)^\top \Delta\theta_m^{H'},\end{aligned} \tag{31}$$

where A and b are defined in (17) and (18) respectively. The Newton step is regularized using a scalar $\tau > 0$ and the identity matrix I.

Remark 2. *As we replace the true action-value error with the TD error in (29), Corollary 1 no longer holds, even under Assumption 1. Generally, we cannot expect to achieve an average TD error of zero. This is because the parameter update in (14) involves taking the product of two expectations including the next state s^+. To obtain an unbiased sample of this product, two independent samples of s^+ are needed. During normal interaction with a system, this is not possible. However, we can expect to reduce the average TD errors toward the true minimum.*

The Fisher information matrix will be rank deficient in case we include a storage function in the cost (4). Because the Q-learning Hessian is not the Hessian of the true Q-learning objective in (10), the matrix A can be ill-conditioned. The regularization term in (31) is therefore added to prevent the Hessian approximation from becoming singular. Regularization may also be added to ensure positive definiteness.

Remark 3. *The Fisher information matrix in (28) is positive semi-definite by construction. A well-posed LSTDQ step is characterized by a negative definite Hessian. We therefore modify the signs of the LSTDQ update, such that the multi-objective Hessian should be positive definite. Potentially indefinite Hessian approximations can be tackled using regularization, as proposed here, or trust-region methods, see e.g. [24].*

We note that the computation of the Q-learning Hessian is based on a sensitivity analysis of (7), which is not computationally heavy, and much cheaper than solving the optimization problem (7) itself. The Fisher information matrix (28) often used as an approximation of the policy Hessian is rather crude, and we therefore propose the alternative second-order update, where the policy Hessian approximation is omitted, i.e.

$$\min_{\Delta\theta_m^H} \frac{1}{2}\Delta\theta_m^{H\top}(-\omega A + \tau I)\Delta\theta_m^H + \alpha_m(\nabla_\theta J(\pi_\theta) - \omega b)^\top \Delta\theta_m^H. \tag{32}$$

We can further simplify the step in (32), by replacing the multi-objective Hessian approximation with the identity matrix, and obtain the multi-objective first-order step as the solution to the following

$$\min_{\Delta\theta_m} \frac{1}{2}\Delta\theta_m^\top\Delta\theta_m + \alpha_v(\nabla_\theta J(\pi_\theta) - \omega b)^\top \Delta\theta_m. \tag{33}$$

IV. Simulations

In this section, we consider two simulation examples. The first simulation example is included as a motivating example and is a seemingly simple case of the economic linear quadratic regulator (ELQR), i.e. an LQR with weighting matrices that are not positive definite. The example becomes challenging due to the shape of the action-value function, which calls for a combination of RL methods in order to capture both the correct value function and policy. The second example is a linear MPC (LMPC) that we use to benchmark the convergence of the different parameter update regimes.

A. Economic LQR

We consider an ELQR for a system with dynamics

$$s_{k+1} = 0.1 s_k + a_k, \tag{34}$$

and stage cost

$$L(s,a) = -s^2 + 10a^2. \tag{35}$$

We introduce the following artificial constraints, to work with compact and bounded constraint sets and thereby comply with dissipativity theory for economic problems [14]

$$-100 \le a \le 100, \quad -100 \le s \le 100. \tag{36}$$

For the set of states that never activate the constraints, we can solve the Riccati equation for the discrete system, and obtain the optimal value function and policy. For the dynamics (34) and stage cost (35) in the unconstrained case, the optimal value function and policy is

$$V^\star(s) = -1.0113 s^2, \ \pi^\star(s) = 0.0113 s. \tag{37}$$

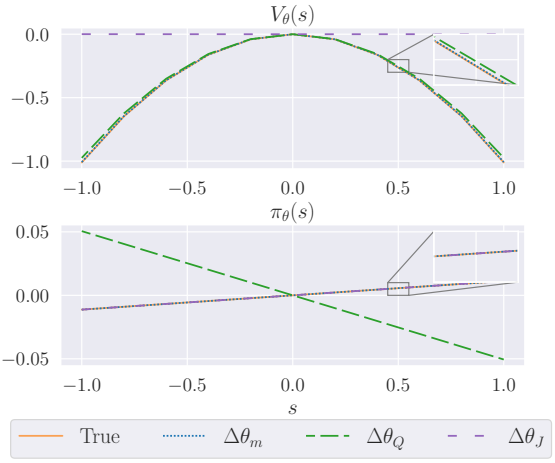

Fig. 1: Economic LQR: Value function estimate (top) and policy (bottom).

We formulate the following finite-horizon linear MPC scheme

$$\min_{x,u} \quad -\lambda_\theta(s) + T_\theta(x_N) + \sum_{i=0}^{N-1} \ell_\theta(x_i, u_i) \tag{38a}$$

$$\text{s.t} \quad \forall i \in \mathbb{I}_{0:N-1} : x_{i+1} = 0.1x_i + u_i, \tag{38b}$$

$$-100 \leq x_i \leq 100, \tag{38c}$$

$$-100 \leq u_i \leq 100, \quad x_0 = s. \tag{38d}$$

We let λ_θ, T_θ and ℓ_θ be fully parameterized quadratic functions. The parameter vector θ is then a vector containing all elements in $\lambda_\theta, T_\theta, \ell_\theta$. All matrices were initialized with the identity matrix. We learned in an episodic manner, using trajectories of length $K = 10$ for initial condition $s_0 = 1.0$ with learning rate $\alpha = 0.1$ for all update schemes. For exploration, we use the policy as defined in (6), with Gaussian noise described by $\sigma_p = 10^{-3}$. We tested learning using standard first-order Q-learning (12), denoted $\Delta\theta_Q$, DPG (20) denoted $\Delta\theta_J$ as well as first-order multi-objective combination (33) using $\omega = 1$, denoted $\Delta\theta_m$. For simplicity, we let the feature vector in (23) be polynomials up to 2^{nd} degree, for both simulation examples. As the optimal Q-function for this system is known, and this serves only as a motivational example, we use the analytical gradient in the formulation in (21). We saw convergence for all methods after 20 batches consisting of $m = 1$ episodes. In Figure 1 we have plotted the value function estimate and the policy using the final values of the parameters. We see that the pure DPG method fails to learn the value function, but estimates the true policy well. The pure Q-learning step, on the other hand, learns the value function well, but not the policy. The combination of methods using the first-order multi-objective update, however, manages to learn both the true value function and policy after 20 batches.

For economic problems, we can use Q-learning as a tool to learn the storage function, needed to verify dissipativity as detailed in [10]. Verification is done by checking that the rotated cost, denoted by $\bar{\ell}_\theta(s, a)$ and defined by the learned storage function, λ_θ, according to

$$\bar{\ell}_\theta(s, a) = L(s, a) - \lambda_\theta(s^+) + \lambda_\theta(s), \tag{39}$$

satisfies the following condition i.e.

$$\bar{\ell}_\theta(s, a) \geq \rho(\|s\|), \tag{40}$$

where $\rho \in \mathcal{K}_\infty$. In Figure 2 we see that using DPG alone, the resulting rotated cost is not lower-bounded in s, whereas for the multi-objective combination, we obtain a rotated cost that clearly satisfies this lower bound. In summary, we can verify that this problem is dissipative, and also capture the optimal policy, by using a combination of RL methods.

B. Linear MPC

We consider a discrete linear system of the form

$$s_{k+1} = As_k + Ba_k + n, \tag{41}$$

where n describes Gaussian process noise i.e. $n \sim \mathcal{N}(0, \sigma_n^2 I_n)$, with standard deviation $\sigma_n = 10^{-3}$ and where I_n is the identity matrix. The system matrices are given as

$$A = \kappa \begin{bmatrix} \cos\beta & \sin\beta \\ \sin\beta & \cos\beta \end{bmatrix}, \; B = \begin{bmatrix} 1.1 & 0 \\ 0 & 0.9 \end{bmatrix}, \tag{42}$$

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

$$L(s, a) = \frac{1}{20}\|s - s_{\text{ref}}\|^2 + \frac{1}{2}\|a - a_{\text{ref}}\|^2, \tag{43}$$

where $s_{\text{ref}} = [0.1, 0.1]^\top$, and a_{ref} is found according to (41). The parameterized MPC scheme reads as

$$\min_{x,u} \; V_0 + \gamma^N \|x_N - x_{\text{ref}}\|_P^2 + \sum_{i=0}^{N-1} \gamma^i f^\top \begin{bmatrix} x_i \\ u_i \end{bmatrix} + \sum_{i=0}^{N-1} \gamma^i \left\| \begin{bmatrix} x_i - x_{\text{ref}} \\ u_i - u_{\text{ref}} \end{bmatrix} \right\|^2 \tag{44a}$$

$$\text{s.t} \quad \forall i \in \mathbb{I}_{0:N-1} : x_{i+1} = \begin{bmatrix} \theta_1 & \theta_2 \\ \theta_3 & \theta_4 \end{bmatrix} x_i + \begin{bmatrix} \theta_5 & 0 \\ 0 & \theta_6 \end{bmatrix} u_i, \tag{44b}$$

$$\begin{bmatrix} -0.05 \\ -0.05 \end{bmatrix} \leq u_i \leq \begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix}, \quad x_0 = s, \tag{44c}$$

using a prediction horizon of $N = 10$ and discount factor $\gamma = 0.99$. The stage cost consists of a quadratic function of the state and input and a linear term that can be used to shift the minimum of the stage cost. We let $x_{\text{ref}} = s_{\text{ref}}$, but use the prediction model to obtain the input reference, i.e. $u_{\text{ref}} \neq a_{\text{ref}}$. The linear term is defined by a vector $f^\top = [f_1, f_2, f_3, f_4]$. The parameter vector is $\theta = \{V_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, f_1, f_2, f_3, f_4\}$. The prediction model is initialized with parameter values $\theta_1 = \theta_3 = \cos\hat{\beta}$ and $\theta_2 = \theta_4 = \sin\hat{\beta}$, where $\hat{\beta} = 30$ [deg], and $\theta_5 = 1.3, \theta_6 = 0.7$. We let $f_1 = f_2 = f_3 = f_4 = 0.3$

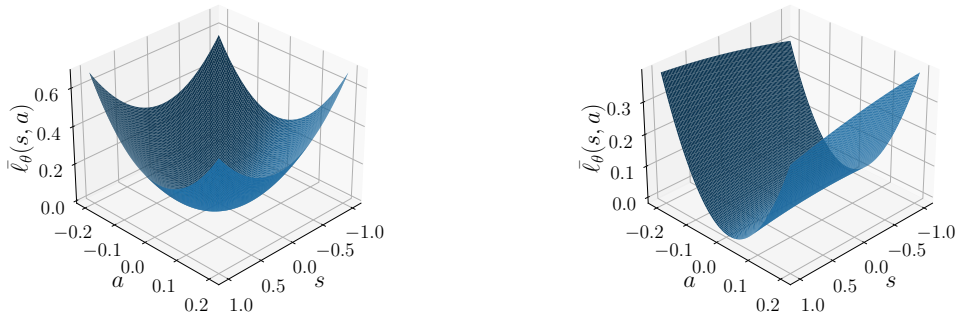

Fig. 2: Economic LQR: The rotated cost (39) as defined by the learned storage function using multi-objective RL (left) and using DPG (right).

TABLE I: LMPC: Update schemes tested in simulation.

Label	Name (eq. reference)	Learning rate	L_{tot}
$\Delta\theta_Q$	Q-learning (12)	$\alpha_q = 0.01$	19.69
$\Delta\theta_Q^H$	LSTDQ (14)	$\alpha_d = 0.5$	18.63
$\Delta\theta_m^H$	Second-order multi-objective (32)	$\alpha_m = 0.5$	**4.63**
$\Delta\theta_m^{H'}$	Second-order multi-objective (31)	$\alpha_m = 0.5$	6.64
$\Delta\theta_J^H$	Natural policy gradient (27)	$\alpha_r = 0.5$	5.12
$\Delta\theta_m$	First-order multi-objective (33)	$\alpha_v = 0.5$	12.46
$\Delta\theta_J$	DPG (20)	$\alpha_p = 0.5$	12.37

and $V_0 = 0$. We use a quadratic terminal cost, for which P is found solving the discrete Riccati equation for the prediction model, using the initial values of the parameters.

We learned in an episodic manner, simulating the system from initial condition $s_0 = [0, 0]^\top$, for a total of 25 batches, consisting of $m = 10$ episodes of length $K = 50$. To explore, the policy was perturbed as defined in (6), with $\sigma_a = 5 \cdot 10^{-3}$. For this example, we tested both first- and second-order multi-objective RL as described in Section III-A. We compared their performance with both the second-order and first-order policy gradient methods, as well as classical Q-learning and LSTDQ. For the Q-learning Hessian and Fisher information matrix, we regularized using $\tau = 10^{-1}$ and $\tau = 10^{-2}$ respectively, at all time steps. The learning rates were selected by testing each method for an interval of learning rates and selecting the best-performing one. The sum of costs was evaluated for the best-performing case, reported in Table I, evaluated according to

$$L_{\text{tot}} = \sum_{j=1}^{m} \sum_{k=1}^{K} \gamma^k L(s_{k,j}, a_{k,j}). \tag{45}$$

All plotting labels are also listed and explained in Table I.

In Figure 3, we see the closed-loop performance during learning for each of the parameter update schemes. Because the RL cost in (43) is rather flat in s-direction, this is

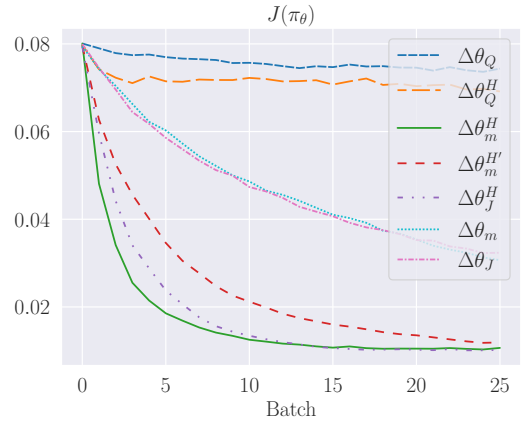

Fig. 3: LMPC: Mean closed-loop performance over learning batches.

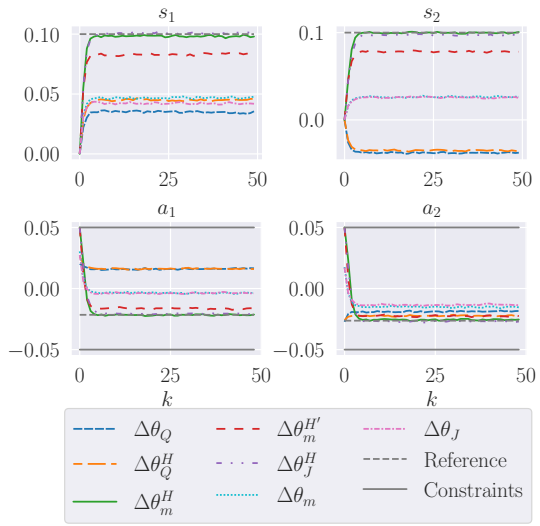

Fig. 4: LMPC: Simulated states and actions using the learned parameters.

typically a challenge for Q-learning alone, and hence we see both classical TD-learning and LSTD-learning achieving the poorest performance. First-order DPG and the first-order multi-objective approach achieve more or less the same closed-loop performance. Moreover, we see that the closed-loop performance obtained using the natural policy gradient method converges faster than both first-order methods. The second-order multi-objective approach performs better without the Fisher information matrix in the Hessian, as given in (32), and speeds up convergence further. The aforementioned observations align also with the calculated sum of discounted costs during learning for the different update schemes as listed in Table I. For the selected learning rates, we obtained the best closed-loop performance for the multi-objective approach by using $\omega = 0.1$ in (32), (31) and (33).

In Figure 4 we have plotted the states and inputs resulting from the learned parameters in exploitation, i.e. without exploration noise. We note that the upper bounds on the inputs become active constraints as we approach the optimal policy, which is the case for the second-order multi-objective and natural policy gradient method.

For the second-order multi-objective approach, we saw that the Q-learning gradient part caused learning to converge to a closed-loop behavior with a small steady-state error. This is likely caused by the same effect as addressed in Remark 2. We therefore removed the first-order Q-learning contribution to produce the results shown in Figures 3 and 4, at the cost of some reduction in convergence rate. Ideally, ω could be gradually reduced, to gain the full increase in convergence rate, while still converging towards the optimum.

V. Conclusion

In this paper, we proposed a multi-objective approach for combining RL methods, in order to fully exploit the parameterization provided by the MPC scheme and increase the convergence rate of learning. The first simulation example illustrates that we need a combination of Q-learning and policy gradient methods for certain economic policies to verify dissipativity and learn both the optimal value function and policy. The second simulation example demonstrates that the proposed multi-objective second-order step, combining Q-learning and policy gradient methods, speeds up convergence in learning compared to both a second-order policy gradient method and a second-order Q-learning method.

Acknowledgment

This work was supported by the industry partners Borregaard, Elkem, Yara, Hydro and the Research Council of Norway through the project TAPI: Towards Autonomy in Process Industries, project number: 294544.

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

[4] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2019.

[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, 2013.

[6] 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.

[7] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3638–3652, 2020.

[8] S. Gros and M. Zanon, "Learning for MPC with stability & safety guarantees," *Automatica*, vol. 146, p. 110598, 2022.

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

[10] A. B. Kordabad and S. Gros, "Verification of dissipativity and evaluation of storage function in economic nonlinear MPC using Q-learning," *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 308–313, 2021.

[11] K. Seel, A. B. Kordabad, S. Gros, and J. T. Gravdahl, "Convex neural network-based cost modifications for learning model predictive control," *IEEE Open Journal of Control Systems*, vol. 1, pp. 366–379, 2022.

[12] B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih, "Combining policy gradient and Q-learning," *International Conference on Learning Representations*, 2017.

[13] S. M. Kakade, "A natural policy gradient," *Advances in neural information processing systems*, vol. 14, 2001.

[14] R. Amrit, J. B. Rawlings, and D. Angeli, "Economic optimization using model predictive control with a terminal cost," *Annual Reviews in Control*, vol. 35, no. 2, pp. 178–186, 2011.

[15] S. Gros and M. Zanon, "Bias correction in reinforcement learning via the deterministic policy gradient method for MPC-based policies," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 2543–2548.

[16] A. B. Kordabad and S. Gros, "Q-learning of the storage function in Economic Nonlinear Model Predictive Control," *Engineering Applications of Artificial Intelligence*, vol. 116, p. 105343, 2022.

[17] J. A. Boyan, "Least-squares temporal difference learning," in *International Conference on Machine Learning*, 1999, pp. 49–56.

[18] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[19] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *31st International Conference on Machine Learning, ICML 2014*, vol. 1, pp. 605–619, 2014. [Online]. Available: http://proceedings.mlr.press/v32/silver14.pdf

[20] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, no. 6, pp. 1107–1149, 2004.

[21] A. B. Kordabad, H. N. Esfahani, W. Cai, and S. Gros, "Quasi-Newton iteration in deterministic policy gradient," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 2124–2129.

[22] T. Furmston, G. Lever, and D. Barber, "Approximate Newton methods for policy search in Markov decision processes," *Journal of Machine Learning Research*, vol. 17, 2016.

[23] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[24] Y. Yuan, "A review of trust region algorithms for optimization," in *International Council for Industrial and Applied Mathematics*, vol. 99, no. 1, 2000, pp. 271–282.