

Practical Reinforcement Learning of Stabilizing Economic MPC

Mario Zanon¹, Sébastien Gros² and Alberto Bemporad¹

Abstract—Reinforcement Learning (RL) has demonstrated a huge potential in learning optimal policies without any prior knowledge of the process to be controlled. Model Predictive Control (MPC) is a popular control technique which is able to deal with nonlinear dynamics and state and input constraints. The main drawback of MPC is the need of identifying an accurate model, which in many cases cannot be easily obtained. Because of model inaccuracy, MPC can fail at delivering satisfactory closed-loop performance. Using RL to tune the MPC formulation or, conversely, using MPC as a function approximator in RL allows one to combine the advantages of the two techniques. This approach has important advantages, but it requires an adaptation of the existing algorithms. We therefore propose an improved RL algorithm for MPC and test it in simulations on a rather challenging example.

I. INTRODUCTION

Reinforcement learning (RL) is a *model-free* control technique which recursively updates the controller parameters in order to achieve optimality. Once the controller parameters have been learned, the controller will implement the control action which yields an infinite-horizon optimal cost for each initial condition [1]. RL has drawn increasing attention thanks to the striking results obtained in beating chess and go masters [2], and in learning how to make a robot walk or fly without supervision [3], [4].

In order to be able to solve the RL problem in practice, function approximation strategies must be employed. Such approximations typically rely on a set of basis functions, or *features*, and corresponding parameters that multiply them. In this case, the function approximator is linear in the parameters. However, nonlinear function approximators are commonly deployed using Deep Neural Networks (DNN).

Model Predictive Control (MPC) is a *model-based* technique which exploits a model of the system dynamics to predict the system’s future behavior and optimize a given performance index, possibly subject to input and state constraints [5], [6], [7]. The success of MPC is due to its ability to enforce constraints and yield optimal trajectories. While a plethora of efficient algorithms for the online solution of MPC problems has been developed, the main drawback of this control technique is the need of identifying the open-loop model offline, which is typically the most time-consuming phase of control design.

The advantages of MPC and RL can be combined together by framing MPC as a function approximator within an RL context. The tuning parameters of the MPC problem (e.g.,

cost weighting matrices, model parameters, etc.) can be framed as parameters of a nonlinear function approximator, namely the MPC optimization problem. This setup can be adopted to approximate the feedback policy, the value function, or the action-value function.

The main advantages of combining MPC with RL are (a) the ease of introducing a constraint-enforcing policy within RL, (b) the possibility of improving existing models and controllers by using them as an initial guess in RL, and (c) the possibility of interpreting the learned algorithm in a model-based framework.

The combination of learning and control techniques has been proposed in, e.g., [8], [9], [10], [11]. Some attempts at combining RL and the linear quadratic regulator have been presented in [12], [13]. To the best of our knowledge, however, [14] is the first work proposing to use NMPC as a function approximator in RL.

In this paper, building on the ideas from [14], we further analyse the combination of RL and MPC. The main contribution of this paper is the development of an improved algorithmic framework tailored to the problem formulation which overcomes some shortcomings of basic RL algorithms. We demonstrate the potential of our approach in simulations on an involved nonlinear example from the process industry.

This paper is structured as follows. Section II introduces Q -learning. The main contributions of this paper are presented in Section III, which discusses the use of MPC as a function approximator in RL, and Section IV, which presents an algorithm adaptation, tailored to the problem. Some numerical examples are given in Section V. The paper is concluded by Section VI which also outlines future research directions.

II. REINFORCEMENT LEARNING

Consider a Markov Decision Process (MDP) with state transition dynamics $\mathbb{P}[s_+|s, a]$, where s and a denote the states and actions (or controls) respectively. We also introduce the stage cost $\ell(s, a)$ and the discount factor $0 < \gamma \leq 1$. The action-value function $Q_*(s, a)$ and value function $V_*(s)$ associated with the optimal policy $\pi_*(s)$ are defined by the Bellman equations:

$$Q_*(s, a) = \ell(s, a) + \gamma \mathbb{E}[V_*(s_+) | s, a], \quad (1a)$$

$$V_*(s) = Q_*(s, \pi_*(s)) = \min_a Q_*(s, a). \quad (1b)$$

Q -learning parametrizes the action value function as $Q_\theta(s, a)$, where θ is a vector of parameters whose values have to be learned, and aims at minimizing $\|Q_*(s, a) -$

¹Mario Zanon and Alberto Bemporad are with the IMT School for Advanced Studies Lucca, Italy. {name.surname}@imtlucca.it

²Sébastien Gros is with the Department of Electrical Engineering, Chalmers University of Technology, Hörsalsvägen 9a, Sweden. grosse@chalmers.se

$Q_\theta(s, a)\|_2^2$. Given a state-action pair (s, a) and the next state s_+ , standard algorithms [1] update the parameter using

$$\delta = \ell(s, a) + \gamma \min_{a_+} Q_{\tilde{\theta}}(s_+, a_+) - Q_\theta(s, a), \quad (2a)$$

$$\theta \leftarrow \theta + \alpha \delta \nabla_\theta Q_\theta(s, a), \quad (2b)$$

where δ is known as the *Temporal-Difference (TD) error*. In batch policy updates $\tilde{\theta}$ is kept constant for N_{upd} steps after which it is updated as $\tilde{\theta} \leftarrow \theta$. In instantaneous policy updates $N_{\text{upd}} = 1$ such that $\tilde{\theta}$ is updated at every time instant.

It is important to underline that, while in this paper we focus on Q -learning (which has been successfully deployed on some applications, e.g., [15], [16], [17]), our approach can be readily deployed on other TD-algorithms such as SARSA [18]. Even in approaches directly optimizing the policy, learning the action-value function is often necessary [19], [20], such that the developments of this paper apply to a fairly broad class of RL algorithms.

While a very common choice is to use Deep Neural Networks (DNN) as function approximators, it is hard to analyze closed-loop stability in DNN-based RL. Moreover, in case a controller is already available, information on how to control the system cannot be easily incorporated in the problem formulation. For this reason, we propose to use MPC as a function approximator, since it makes it possible to enforce closed-loop stability guarantees and an existing controller can be used as initial guess for the algorithm.

III. MPC-BASED RL

The use of MPC to parametrize the action-value function has been first advocated in [14]. In the following, we first present the function approximation used and recall the most important result in Theorem 1, then we further analyze the properties of MPC-based RL and in the next section we will propose a new variant of the Q -learning algorithm tailored to MPC-based RL.

A. Parametrization of the Function Approximations

We parametrize the action-value function using an MPC scheme of the form

$$Q_\theta^N(s, a) = \min_z \lambda_\theta(x_0) + \gamma^N V_\theta^f(x_N) + \sum_{k=0}^{N-1} \gamma^k \ell_\theta(x_k, u_k)$$

$$+ \sum_{k=0}^N \gamma^k (s_k^\top W_s s_k + w_s^\top s_k) \quad (3a)$$

$$\text{s.t. } x_0 = s, \quad u_0 = a, \quad (3b)$$

$$x_{k+1} = f_\theta(x_k, u_k), \quad (3c)$$

$$g_\theta(u_k) \leq 0, \quad (3d)$$

$$h_\theta(x_k, u_k) \leq s_k, \quad h_\theta^f(x_N) \leq s_N, \quad (3e)$$

where $z = (x_0, u_0, s_0, \dots, x_N)$. Consequently, we obtain

$$\pi_\theta^N(s) = \arg \min_a Q_\theta^N(s, a), \quad V_\theta^N(s) = \min_a Q_\theta^N(s, a).$$

Note that the policy $\pi_\theta^N(s)$ and value function $V_\theta^N(s)$ are equivalently obtained by solving Problem (3) with constraint $u_0 = a$ removed. In order to address feasibility issues in

Problem (3), in the presence of state-dependent constraints we adopt an exact relaxation of such constraints [21].

Remark 1: MPC formulations in which the stage cost is not positive-definite are commonly referred to as *Economic MPC*. In order to enforce closed-loop stability and the existence of the MPC solution, we assume that ℓ_θ and V_θ^f are positive definite functions. Therefore, in order to deal with the situation in which the true stage cost ℓ is not positive-definite, we have introduced the arrival penalty λ_θ . For all details on this topic, we refer to [14] and references therein.

Remark 2: The proposed setup readily accommodates for formulations in which the cost penalizes deviations from a given reference. In that case, the stage cost (both in RL and MPC) depends on a reference, passed to the problem as an exogenous signal. Input-output model formulations also readily fit in the proposed framework.

Among the desirable properties of the proposed formulation, we mention nominal stability guarantees, and the possibility to introduce constraints accounting for, e.g., actuator limitations, safe operation of the system, etc. While being aware that the current framework is not able to fully exploit these advantages, with the presented developments we aim at constructing a sound basis which will be used in future research with the intent of developing self-tuning, safe, and stable economic MPC controllers.

B. Learning the Model: RL and System Identification

We recall the following fundamental theorem from [14], which states that the optimal value and action-value functions as well as the optimal policy can be learned even by using MPC based on a state transition model \hat{f} which is different from the true model f . This also entails that there is no guarantee (and no need) that the RL algorithm will learn a physically meaningful model.

Theorem 1 ([14]): Consider a given (possibly stochastic) state transition model $\hat{s}_{k+1} = \hat{f}(\hat{s}_k, \hat{a}_k)$, possibly different from the true model $f(s_k, a_k)$. Define the optimal value function

$$\hat{V}^N(s) = \min_\pi \mathbb{E} \left[\gamma^N V_\star(\hat{s}_N^{\hat{\pi}}) + \sum_{k=0}^{N-1} \gamma^k \hat{\ell}(\hat{s}_k^{\hat{\pi}}, \hat{\pi}(\hat{s}_k^{\hat{\pi}})) \right]$$

associated with stage cost $\hat{\ell}$, and the terminal cost $V_\star(s)$ over an optimization horizon N . Define $\hat{s}_{0,\dots,N}^{\hat{\pi}}$ as the (possibly stochastic) trajectories of the state transition model \hat{f} under a policy $\hat{\pi}$, starting from $\hat{s}_0^{\hat{\pi}} = s$, and $\hat{\pi}^N$ the optimal policy associated with $\hat{V}^N(s)$ and $\hat{Q}^N(s, a)$ the associated action-value function. Consider the set \mathcal{S} such that

$$|\mathbb{E}[V_\star(\hat{s}_k^{\hat{\pi}^*})]| < \infty, \quad \forall s \in \mathcal{S}.$$

Then, $\exists \hat{\ell}$ such that the following identities hold on \mathcal{S} :

- (i) $\hat{V}^N(s) = V_\star(s)$
- (ii) $\hat{\pi}^N(s) = \pi_\star(s)$
- (iii) $\hat{Q}^N(s, a) = Q_\star(s, a)$ for the inputs a such that $|\mathbb{E}[V_\star(\hat{s}_+) | s, a]| < \infty$.

We can now further clarify the theorem and formalize a form of ‘‘orthogonality’’ between reinforcement learning and system identification.

Corollary 2: Let us split the parameter as $\theta = (\theta_f, \theta_Q)$, such that the model f_θ only depends on θ_f and assume a perfect parametrization, such that, under adequate exploration of the state-action space, RL learns Q and V perfectly with $\hat{\theta} \neq \theta$, i.e.,

$$\delta_k = \ell(s_k, u_k) + \gamma \mathbb{E} \left[V_{\hat{\theta}}^N(s_{k+1}) \right] - Q_{\hat{\theta}}^N(s_k, a_k) = 0.$$

Then, leaving the task of identifying θ_f to a separate identification algorithm is not detrimental to the learning of the optimal value, action-value function, and policy.

Unfortunately, Corollary 2 applies only to the case of perfect parametrization. In practice, function approximation with imperfect parametrization and lack of excitation can destroy this form of orthogonality. Ongoing research is currently further investigating such aspects.

C. On the Parametrization of V and Q

Since the Bellman principle of optimality states that $Q_\theta^N(s, a) = \ell_\theta(s, a) + V_\theta^{N-1}(f_\theta(s, a))$, one could be tempted to replace $V_\theta^N(s) = \min_a Q_\theta^N(s, a)$ by $V_\theta^{N-1}(s)$ in the computation of the TD error. We discuss in the following lemma why this approach can minimize the TD error while not learning the correct action-value function and, consequently, not delivering the optimal policy.

Lemma 3: Consider computing the TD error (2a) by replacing $\min_a Q_\theta^N(s, a)$ by $V_\theta^{N-1}(s)$ to obtain

$$\delta = \ell(s, a) + \gamma V_\theta^{N-1}(f(s, a)) - Q_\theta^N(s, a). \quad (4)$$

Then, it is possible to obtain $\delta = 0$ without minimising the error $\|Q^*(s, a) - Q_\theta^N(s, a)\|^2$ and, therefore, without learning the optimal policy.

Proof: We prove the Lemma by a simple counter example. Consider the LQR case, with $\theta = (\hat{A}, \hat{B}, \hat{P})$ and

$$Q_\theta^1(s, a) = \begin{bmatrix} s \\ a \end{bmatrix}^\top \begin{bmatrix} T + \gamma \hat{A}^\top \hat{P} \hat{A} & S + \gamma \hat{A}^\top \hat{P} \hat{B} \\ S^\top + \gamma \hat{B}^\top \hat{P} \hat{A} & R + \gamma \hat{B}^\top \hat{P} \hat{B} \end{bmatrix} \begin{bmatrix} s \\ a \end{bmatrix},$$

$$V_\theta^0(s) = s^\top \hat{P} s.$$

Then, (4) reads as

$$\delta = \gamma \begin{bmatrix} s \\ a \end{bmatrix}^\top \left(\begin{bmatrix} A & B \end{bmatrix}^\top \hat{P} \begin{bmatrix} A & B \end{bmatrix} - \begin{bmatrix} \hat{A} & \hat{B} \end{bmatrix}^\top \hat{P} \begin{bmatrix} \hat{A} & \hat{B} \end{bmatrix} \right) \begin{bmatrix} s \\ a \end{bmatrix}.$$

Assume now that the correct model is identified, i.e., $\hat{A} = A$, $\hat{B} = B$. Then, the TD error is independent of \hat{P} , which can be chosen as desired. This implies that, even though we use a perfect parametrization of the action-value function, $Q_\theta^1(s, a) \neq Q_*(s, a)$. Consequently, the learned feedback given by

$$\hat{K} = (R + \gamma \hat{B}^\top \hat{P} \hat{B})^{-1} (S^\top + \hat{B}^\top \hat{P} \hat{A}) \quad (5)$$

is not uniquely defined. ■

For comparison, we consider now the standard case, in which the TD error is computed using

$$V_\theta^1(s) = \min_a Q_\theta^1(s, a)$$

$$= s^\top (T + \gamma \hat{A}^\top \hat{P} \hat{A} - (S + \gamma \hat{A}^\top \hat{P} \hat{B}) \hat{K}) s,$$

where \hat{K} is given by (5). In this case, if $\hat{A} = A$, $\hat{B} = B$, \hat{P} must solve the algebraic Riccati equation associated with the given stage cost and model. It is important to stress, however, that infinitely many solutions exist in general such that the true model will not be identified. As an example we provide $\theta_1 = (0.5\hat{A}, 0.5\hat{B}, 4\hat{P})$: it is immediate to verify that $Q_{\theta_1}^1(s, a) \equiv Q_\theta^1(s, a)$.

D. Condensed Model-Free Parametrization

Motivated by the previous considerations on the possibility of learning a wrong model, we propose next a formulation which is truly model-free as it does not include model parameters as coefficients to be learned. For simplicity we only focus here on the case of linear dynamics with a quadratic cost. We define $z = (x_0, u_0, \dots, u_{N-1})$; then

$$Q_\theta^N(s, a) = \min_z z^\top M z + m^\top z + c \quad (6a)$$

$$\text{s.t. } x_0 = s, \quad u_0 = a, \quad (6b)$$

$$Cz \leq d, \quad (6c)$$

where $\theta = (M, C, d, m)$ and M is a symmetric (typically dense) matrix. Problem formulation (6) is sometimes used in MPC and optimal control, in which case the parameters θ are related to the system dynamics and cost [22]. In this case, the dependence of V and Q on the parameters is less nonlinear than in the model-based parametrization, see (3). However, the introduction of a prediction horizon $N > 1$ possibly introduces more parameters than the standard formulation. A special case occurs when there are only input constraints and they are known: in this case C can be fixed and only M, m need to be learned. The LQR case simplifies to

$$Q_\theta^1(s, a) = \begin{bmatrix} s \\ a \end{bmatrix}^\top \begin{bmatrix} \bar{T} & \bar{S} \\ \bar{S}^\top & \bar{R} \end{bmatrix} \begin{bmatrix} s \\ a \end{bmatrix}, \quad (7)$$

such that $K = \bar{R}^{-1} \bar{S}^\top$ and $\pi^*(s) = -Ks$

IV. ALGORITHM

The main practical difficulties related to using MPC as a function approximator in RL are that: (a) the function approximator is nonlinear in the parameters θ , and (b) the MPC problem is guaranteed to have a meaningful solution only if the cost is positive-definite. In this section we propose an algorithm that addresses both issues.

We start by recalling that the main motivation for the stochastic gradient approach typically used in Q -learning stems from an equivalence with the table-lookup case, i.e., when the state-action space is discrete and the action-value function is parametrized as

$$Q_\theta(s, a) = \sum_{\bar{s} \in \mathcal{S}, \bar{a} \in \mathcal{A}} \theta_{\bar{s}, \bar{a}} \phi_{\bar{s}, \bar{a}}(s, a),$$

and $\phi_{\bar{s}, \bar{a}}(s, a) = 1$ if $\bar{s} = s, \bar{a} = a$ and 0 otherwise. In this case, Q_θ is linear in the parameter θ . Parameter α in (2b) is used in order to approximate the expected value of the TD error: one can roughly interpret it as the inverse of the amount of samples over which the average is computed.

In this case, the update (2b) obtained with the (exact) function approximation matches that of the enumeration

$$\sum_{\bar{s} \in \mathcal{S}, \bar{a} \in \mathcal{A}} \theta_{\bar{s}, \bar{a}} \delta_{\bar{s}, \bar{a}}(s, a) = \alpha \delta.$$

The update (2b) can also be written as $\theta \leftarrow \theta + \alpha \Delta \theta^*$ with

$$\Delta \theta^* = \arg \min_{\Delta \theta} (\delta_k - \nabla_{\theta} Q_{\theta} \Delta \theta)^2.$$

We remark that this is not the case if ϕ is not normalized or if Q is a nonlinear function of the parameter θ . This also implies that parameter α loses its original meaning, as it is also used to dampen the stochastic gradient step in order to enforce convergence.

We propose to apply the update $\theta \leftarrow \theta + \alpha(\theta^* - \theta)$ where θ^* is the solution of

$$\min_{\theta} \sum_{k=0}^n \left(\ell(s_k, a_k) + \gamma \min_{a'} Q_{\theta}^N(s_{k+1}, a') - Q_{\theta}^N(s_k, a_k) \right)^2 \quad (8a)$$

$$\text{s.t. } \nabla^2 \ell_{\theta} \succ 0, \quad \nabla^2 V_{\theta}^f \succ 0. \quad (8b)$$

Note that we solve Problem (8) to full convergence at each step. In case of a linear parametrization of Q_{θ} , convergence is obtained in one step by using a Gauss-Newton Hessian approximation, in case Constraints (8b) are inactive. This also directly solves the issue of scaling, present in the table-lookup case if ϕ is not normalized. Finally, if instead of solving Problem (8) to full convergence constraints (8b) are neglected and one takes a single full Newton step, the update (2b) is recovered.

We highlight next the main features of Formulation (8):

- *Globalization*: globalization strategies such as line search ensure descent and, therefore, convergence. Descent is typically not enforced in stochastic gradient approaches such as (2), which could take a step which yields a larger TD error than with the previous parameter estimate.
- *Positive-definiteness enforcement*: Constraints (8b) guarantee that the cost is positive-definite and, therefore, let the MPC problem be stabilizing and well-posed. Without this constraint, even with a well-posed, positive-definite cost, throughout the RL iterates one might obtain an indefinite cost yielding an unbounded solution. We have written the positive-definiteness Constraints (8b) using the Hessian of ℓ and V^f , which is reasonable if both functions are quadratic. However, other approaches relying on richer function approximations are possible, e.g. using sum-of-squares techniques.
- *Best fit*: by solving Problem (8) to full convergence, the step $\theta^* - \theta$ is the one which minimizes δ , analogously to the lookup-table case. Therefore, the choice of parameter α can be done purely by considerations about the expected value approximation.

In summary, the main differences with the standard update (2b) are: (a) positive-definiteness enforcement, (b) insensitivity to parameter scaling and (c) guarantee of improvement at each step through globalization and full convergence.

A. Derivative Computation

We detail next how to compute the derivatives of the action-value function with respect to the parameters. To this end, we define the Lagrangian function underlying NMPC problem (3) as

$$\begin{aligned} \mathcal{L}_{\theta}^N(s, y) &= \gamma^N V_{\theta}^f(x_N) + \chi_0^{\top} (x_0 - s) + \mu_N^{\top} h_{\theta}^f(x_N) \\ &+ \sum_{k=0}^{N-1} \chi_{k+1}^{\top} (f_{\theta}(x_k, u_k) - x_{k+1}) + \nu_k^{\top} g_{\theta}(u_k) \\ &+ \gamma^k \ell_{\theta}(x_k, u_k) + \mu_k^{\top} h_{\theta}(x_k, u_k) + \zeta^{\top} (u_0 - a), \end{aligned}$$

where χ, μ, ν, ζ are the multipliers associated with constraints (3b)-(3e) and $y = (z, \chi, \mu, \nu, \zeta)$. Note that, for $\zeta = 0$, $\mathcal{L}_{\theta}^N(s, y)$ is the Lagrangian function associated to the NMPC problem defining the value function $\min_a Q_{\theta}(s, a)$. We observe that [23]

$$\nabla_{\theta} Q_{\theta}^N(s, a) = \nabla_{\theta} \mathcal{L}_{\theta}^N(s, y^*) \quad (10)$$

holds for y^* given by the primal-dual solution of (3). Note that this equality holds because constraints (3b) are not an explicit function of θ . The gradient (10) is therefore straightforward to build as a by-product of solving the NMPC problem (3). We additionally observe that

$$\nabla_{\theta} \min_a Q_{\theta}^N(s, a) = \nabla_{\theta} \mathcal{L}^N(s, y^{\diamond}), \quad (11)$$

where y^{\diamond} is given by the primal-dual solution to (3) with constraint $u_0 = a$ removed and $\zeta^{\diamond} = 0$.

We remark that second-order derivatives can also be computed, though in general they depend on the derivative of the optimal primal-dual solution with respect to the parameters, given by

$$\nabla_{\theta} y^* = -\nabla_{\theta} \xi_{\theta}^N(s, y^*) \nabla_y \xi_{\theta}^N(s, y^*)^{-1}, \quad (12)$$

where $\xi_{\theta}^N(s, y)$ gathers the primal-dual KKT conditions underlying the NMPC scheme (3). For a complete discussion on parametric sensitivity analysis of NLPs we refer to [23] and references therein.

V. NUMERICAL EXAMPLE

We consider an example from the process industry, i.e. the evaporation process modelled in [24], [25] and used in [26], [27] to demonstrate the potential of economic MPC in the nominal case. For the sake of brevity we omit the model equations and non-quadratic economic cost function, which include states (X_2, P_2) (concentration and pressure) and controls (P_{100}, F_{200}) (pressure and flow). All details can be found in [26], [27]. The model further depends on concentration X_1 , flow F_2 , and temperatures T_1, T_{200} , which are assumed to be constant in the control model. In reality, these quantities are stochastic with variance $\sigma_{X_1} = 1$, $\sigma_{F_1} = 2$, $\sigma_{T_1} = 8$, $\sigma_{T_{200}} = 5$, and mean centred on the nominal value. Bounds $(25, 40) \leq (X_2, P_2) \leq (100, 80)$ on the states and $100 \leq (P_{100}, F_{200}) \leq 400$ on the controls are present. In particular, the bound $X_2 \geq 25$ is introduced in order to ensure sufficient quality of the product. All state bounds are relaxed as in (3e).

We parametrize an NMPC controller as in (3), i.e., a nonlinear non-condensed MPC formulation, with $N = 10$, $\lambda_\theta, V_\theta^f, l_\theta$ quadratic functions defined by Hessian H_\dagger , gradient h_\dagger , and constant c_\dagger , $\dagger = \{\lambda, V^f, l\}$. The model is parametrized as the nominal model with the addition of a constant, i.e., $f_\theta(x, u) = f(x, u) + c_f$. The control constraints are fixed and the state constraints are parametrized as simple bounds, i.e., $h_\theta(x, u) = (x - x_l, x_u - x)$. The vector of parameter therefore reads as:

$$\theta = (H_\lambda, h_\lambda, c_\lambda, H_{V^f}, h_{V^f}, c_{V^f}, H_l, h_l, c_l, c_f, x_l, x_u).$$

Constants $w_s = 1$, $W_s = I$ are fixed and assumed to reflect the known cost of violating the state constraints.

We use the batch policy update with $\alpha = 10^{-2}$ and update the parameters with the learned ones every $N_{\text{upd}} = 500$ time steps. In order to induce enough exploration, we use an ϵ -greedy policy which is greedy 90% of the samples, while in the remaining 10% we apply the action

$$a = \text{sat}(e, u_l, u_u), \quad e \sim \mathcal{N}(0, \sqrt{10}),$$

where $\text{sat}(\cdot, u_l, u_u)$ saturates the input between its lower and upper bounds u_l, u_u , respectively.

We initialize the ENMPC scheme by the naive initial guess $H_l = I$, $x_l = (25, 40)$, $x_u = (100, 80)$, while all other parameters are 0. While at every step we do check that $H_l, H_{V^f} \succ 0$, during the learning phase the parameters never violate this constraint. As displayed in Figure 1, the algorithm converges to a constant parameter value while reducing the average TD-error. If the standard parameter update (2b) is applied, RL does not converge with the proposed α . If a smaller value is used, the algorithm does not diverge but the parameters θ are updated very slowly, making the approach impractical.

We performed a simulation to compare the RL-tuned NMPC scheme to the naive initial guess and the NMPC tuned using the economic-based approach proposed in [27], which relies on the nominal model. The economic gain obtained by RL is approximately 14% and 12% respectively. The effectiveness of the economically-tuned NMPC had been demonstrated in [27] in the absence of stochastic perturbations. In the considered scenario we observe that, while the economically-tuned scheme still performs better than the naively-tuned one, the RL tuning is able to significantly outperform the two other NMPC schemes by explicitly accounting for the stochastic perturbations.

The concentration X_2 and the difference in instantaneous cost between the RL-tuned and the naively-tuned scheme are displayed in Figure 2. In particular, one can see that RL is trying to stabilize the concentration X_2 to a value which is higher than the nominally optimal one: the optimum in the presence of perturbations is obtained as a compromise between the loss due to operating at $X_2 > 25$ and the cost of violating the constraint $X_2 \geq 25$. Indeed, the constraint is violated but only rarely and by small amounts.

VI. CONCLUSIONS AND OUTLOOK

In this paper we analyzed the use of reinforcement learning to tune MPC schemes, aiming at a self-tuning controller

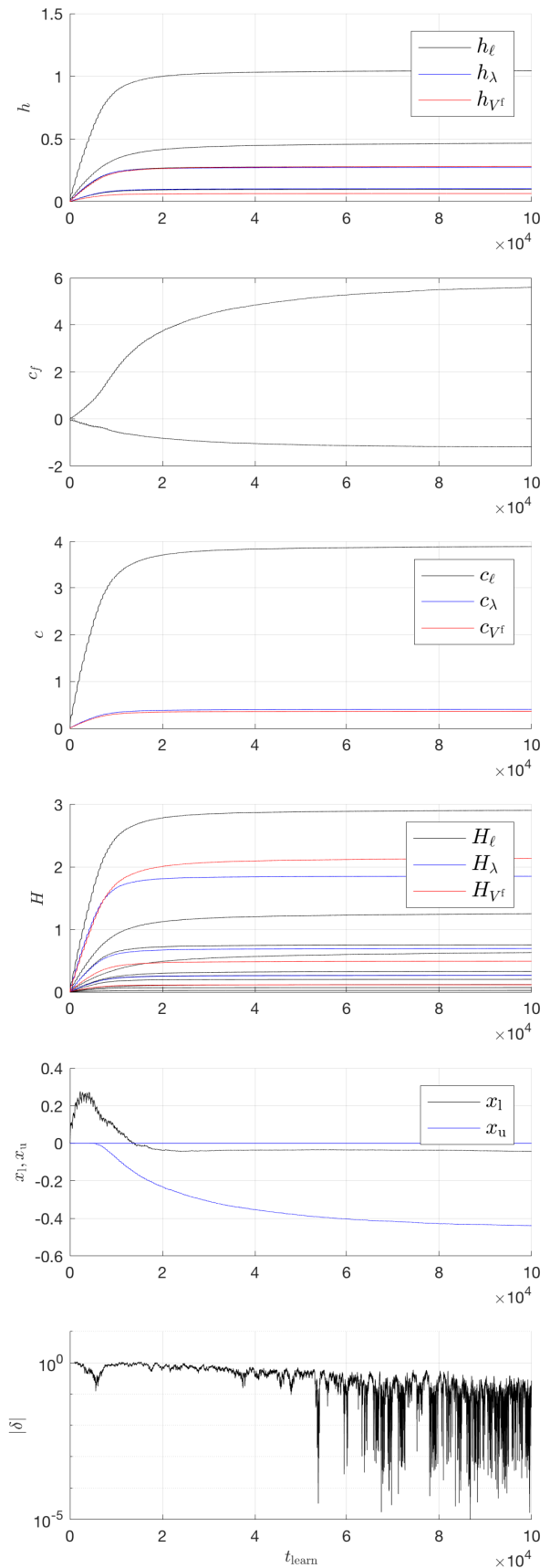


Fig. 1. Evolution of the parameters (increment w.r.t. the initial guess value) and of the TD error (averaged over the preceding 1000 samples).

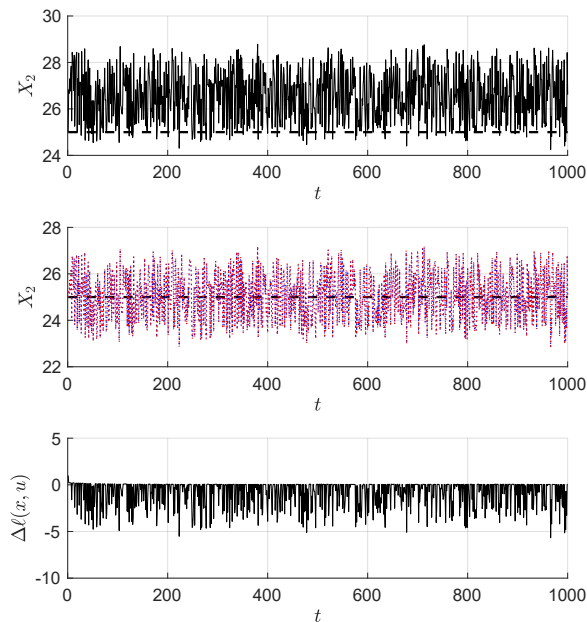


Fig. 2. NMPC closed-loop simulations. Top graph: concentration X_2 for NMPC with RL tuning. Middle graph: naive tuning (red) and nominal economic tuning (blue). In both graphs, the quality constraint is displayed in thick dashed black line. Bottom graph: difference of instantaneous cost between NMPC with RL tuning and naive tuning.

which guarantees stability, safety (i.e., constraint satisfaction), and optimality. In order to be able to apply this approach in practice, we have proposed an improved algorithm based on Q -learning and we have tested it in simulations.

Future work will consider several research directions including: (a) further improvements in the algorithmic framework with the aim of developing data-efficient algorithms; (b) developing new algorithms for other RL paradigms, such as policy gradient methods; (c) further investigating the combination of system identification and RL in order to best update the MPC parameters while guaranteeing safety.

REFERENCES

- [1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction. Second Edition*. MIT press Cambridge, 2018. [Online]. Available: <http://incompleteideas.net/book/the-book-2nd.html>
- [2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, pp. 484–503, 2016.
- [3] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, “An application of reinforcement learning to aerobatic helicopter flight,” in *In Advances in Neural Information Processing Systems 19*. MIT Press, 2007, p. 2007.
- [4] S. Wang, W. Chaovaitwongse, and R. Babuska, “Machine learning algorithms in bipedal robot control,” *IEEE Transactions on Systems, Man, and Cybernetics Part C*, vol. 42, no. 5, pp. 728–743, Sept. 2012.
- [5] J. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill, 2009.
- [6] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control*. London: Springer, 2011.
- [7] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.
- [8] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, “Learning-based Model Predictive Control for Safe Exploration and Reinforcement Learning,” 2018, published on Arxiv.
- [9] 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.
- [10] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, “Robust Constrained Learning-based NMPC enabling reliable mobile robot path tracking,” *The International Journal of Robotics Research*, vol. 35, no. 13, pp. 1547–1563, 2016.
- [11] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, “Safe Model-based Reinforcement Learning with Stability Guarantees,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 908–918.
- [12] F. L. Lewis and D. Vrabie, “Reinforcement learning and adaptive dynamic programming for feedback control,” *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, pp. 32–50, 2009.
- [13] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, “Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers,” *IEEE Control Systems*, vol. 32, no. 6, pp. 76–105, 2012.
- [14] S. Gros and M. Zanon, “Data-Driven Economic NMPC using Reinforcement Learning,” *IEEE Transactions on Automatic Control*, 2018, (under revision). [Online]. Available: <https://mariozanon.wordpress.com/rlrfnmpc>
- [15] C. Watkins, “Learning from delayed rewards,” Ph.D. dissertation, King’s College, Cambridge, 1989.
- [16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.
- [17] G. Theodorou, P. S. Thomas, and M. Ghavamzadeh, “Personalized Ad Recommendation Systems for Life-Time Value Optimization with Guarantees,” in *IJCAI*, 2015, pp. 1806–1812.
- [18] J. P. Jens Kober, J. Andrew Bagnell, “Reinforcement learning in robotics: A survey,” *The International Journal of Robotics Research*, vol. 32, 2013.
- [19] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS’99. Cambridge, MA, USA: MIT Press, 1999, pp. 1057–1063.
- [20] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML’14, 2014, pp. 1–387–1–395.
- [21] P. Sockaert and J. Rawlings, “Feasibility Issues in Linear Model Predictive Control,” *AICHE Journal*, vol. 45, no. 8, pp. 1649–1659, 1999.
- [22] H. Bock and K. Plitt, “A multiple shooting algorithm for direct solution of optimal control problems,” in *Proceedings 9th IFAC World Congress Budapest*. Pergamon Press, 1984, pp. 242–247.
- [23] C. Büskens and H. Maurer, *Online Optimization of Large Scale Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, ch. Sensitivity Analysis and Real-Time Optimization of Parametric Nonlinear Programming Problems, pp. 3–16.
- [24] F. Y. Wang and I. T. Cameron, “Control studies on a model evaporation process constrained state driving with conventional and higher relative degree systems,” *Journal of Process Control*, vol. 4, pp. 59–75, 1994.
- [25] C. Sonntag, O. Stursberg, and S. Engell, “Dynamic Optimization of an Industrial Evaporator using Graph Search with Embedded Nonlinear Programming,” in *Proc. 2nd IFAC Conf. on Analysis and Design of Hybrid Systems (ADHS)*, 2006, pp. 211–216.
- [26] R. Amrit, J. B. Rawlings, and L. T. Biegler, “Optimizing process economics online using model predictive control,” *Computers & Chemical Engineering*, vol. 58, pp. 334 – 343, 2013.
- [27] M. Zanon, S. Gros, and M. Diehl, “A Tracking MPC Formulation that is Locally Equivalent to Economic MPC,” *Journal of Process Control*, 2016.