

Effects of sampling and horizon in predictive reinforcement learning

Pavel Osinenko, Dmitrii Dobriborsci

Abstract—Plain reinforcement learning (RL) may be prone to loss of convergence, constraint violation, unexpected performance, etc. Commonly, RL agents undergo extensive learning stages to achieve acceptable functionality. This is in contrast to classical control algorithms which are typically model-based. An direction of research is the fusion of RL with such algorithms, especially model-predictive control (MPC). This, however, introduces new hyper-parameters related to the prediction horizon. Furthermore, RL is usually concerned with Markov decision processes. But the most of the real environments are not time-discrete. The factual physical setting of RL consists of a digital agent and a time-continuous dynamical system. There is thus, in fact, yet another hyper-parameter – the agent sampling time. In this paper, we investigate the effects of prediction horizon and sampling of two hybrid RL-MPC-agents in a case study with a mobile robot parking, which is in turn a canonical control problem. We benchmark the agents with a simple variant of MPC. The sampling showed a kind of a “sweet spot” behavior, whereas the RL agents demonstrated merits at shorter horizons.

I. INTRODUCTION

Reinforcement Learning (RL) shows remarkable performance in playground settings of video- and table games such as Starcraft, chess and Go [1]–[3]. Industry-close applications appear more challenging to RL due to the lack of freedom in training [4]–[8]. This may be related to limited resources and technical constraints. In general, tuning of RL algorithms is a sensitive matter [9], [10]. Currently, industry is dominated by classical control-theoretic methods such as model predictive control (MPC) [11]–[13]. MPC is widely used in such areas as chemical industry and oil refining [14]–[17]. Somewhat in contrast to the classical control, RL is aimed at a learning-based, model-free (in some configurations) approach. Nevertheless, it is perhaps the model-based formal guarantees that make classical control attractive to the industry. In fact, integration of predictive mechanisms into RL is not new (see, e.g., [18], [19] for a reward roll-out methodology).

Related work. Up to date, using predictive elements from a classical control theory to improve RL approaches is an active area of research. For instance, the promising recent concept of a so-called RL Dreamer effectively uses image-based prediction reminiscent to adaptive MPC [20]. Some model estimation techniques for prediction in the so-called representational learning were introduced in [21]. In the so-called differentiable MPC [22], a combination of model-free and model-based RL elements was suggested. An off-policy actor-critic deep value MPC combining model-based trajectory optimization with the value function estimation was developed in [23]. Another attempt to combine model-based approach and learning techniques is described in

[24] where a differentiable linear quadratic MPC framework for a safe imitation learning was proposed. A predictive scheme was suggested in [25], via an RL agent combined planning with MPC to learn a forward dynamics model. The development of this direction has led to the results presented in [26]. The authors used MPC to learn a cost function from scratch via high-level objective learning and tested it on the real ground vehicle. Clearly, integration of prediction methods in RL is gaining attraction. Therefore, questions of hyper-parameter effects is of relevance.

Speaking of MPC, it has certain fundamental hyper-parameters, such as horizon length and prediction step size, impacting the overall performance [27]–[31]. The current study focuses on the effects of such hyper-parameters, but also time discretization step size (in brief, sampling time). Sampling time may have drastic effects on system stability [32]–[34]. A stable and optimally tuned system with a continuous controller may be destabilized after time discretization [35]. When it comes to RL, there is evidence of performance deterioration of Q-learning under high sampling rates [36]. Authors suggested incorporating the advantage function to remedy issues of collapsing Q-functions. An adaptive discretization for model-based RL via an optimistic one-step value iteration approach was proposed in [37]. However, not much attention has been paid to the effects of prediction step size and prediction horizon length parameters in RL.

Summary. In this study, the sensitivity of three predictive agents (MPC, a roll-out Q-learning, and the so-called stacked Q-learning) to the selected hyper-parameters (sampling rate, prediction step size and prediction horizon length) is investigated on a canonical example of a wheeled robot with dynamic steering torque and pulling force, also known as the extended nonholonomic double integrator [38]–[41]. Such a system is used in numerous studies for benchmarking [42]–[46]. Main findings can be summarized as follows. In terms of the overall tendency for all the methods, too low sampling time leads to a performance deterioration (in terms of the accumulated stage cost), which may be explained by too short-sighted prediction. Increasing the sampling time improves the performance to a certain point where prediction error starts to dominate. Increasing the horizon length boosts the effects of prediction in all the methods and generally leads to a better performance, although at a higher computational cost. Speaking of the method comparison, remarkably, the stacked Q-learning tends to outperform both the MPC and the roll-out algorithms at shorter horizons. In particular, the stacked Q-learning achieved more successful robot parking count. It should be noted here that the roll-out and stacked Q-learning have similar computational complexity.

Notation. Sequences: for any $z: \{z_{i|k}\}_i^N = \{z_{1|k}, \dots, z_{N|k}\} = \{z_k, \dots, z_{k+N-1}\}$, if the starting index k is emphasized; otherwise, just $\{z_i\}^N = \{z_1, \dots, z_N\}$. If N in the above is omitted, the sequence is considered infinite.

II. ENVIRONMENT DESCRIPTION

There are three types of the closed-loop setup (agent-environment, i.e., controller-system), namely, pure discrete, pure continuous, and hybrid. A pure discrete-time design is a discrete system with a discretized controller [47]. A pure continuous-time is a continuous-time system design with a continuous controller, which is rarely possible unless the controller is analogue. A hybrid system is a continuous-time system design with a discretized controller. A continuous-time system design is more suitable for linear systems since some important structural properties might be lost after discretization. The hybrid setting with a continuous-time environment and a discretized controller is considered as a more realistic as the other two variants, and is used as the foundation in this work. Specifically, the following so-called sample-and-hold setting (S&H), where the control actions are held constant during δ -intervals, is considered:

$$\begin{aligned} \mathcal{D}^+ x &= f(x, u^\delta), \\ x_k &:= x(k\delta), \\ u^\delta(t) &\equiv u_k = \kappa(x_k), t \in [k\delta, (k+1)\delta], \end{aligned} \quad (1)$$

where \mathcal{D}^+ is a suitable differential operator, x – state, u – action, κ – policy, t – time, $k \in \mathbb{N}$, $\delta > 0$ – discretization step. Notice that the agent’s actions are to be optimized at discrete time steps, which makes the problem tractable, unlike in a pure continuous setting. In the next sections, predictive control mechanisms are discussed in relation to the described S&H setup.

III. PREDICTIVE CONTROL

Prediction horizon effects. In general, control over a prediction horizon may help improve the agent’s performance and aid system stabilization by using long-term information about the future states. At the same time, in presence of model prediction error, the prediction horizon length and also the prediction step size are subject to careful tuning. In general, the prediction horizon length refers to a period starting from the current time to a point until which control actions are to be optimized. The prediction step size can be described in the following example: if it is two times bigger than the sampling time then the state prediction is two times finer than the sampled control actions. Evidently, higher prediction horizons lead to higher computational complexity. (the total number of action sequences to be searched over increases exponentially with the horizon). At the same time, too short horizon may lead to a failure to even stabilize the system. Traditionally, MPC, which is discussed in the next section, is considered a standard predictive controller.

A. Model Predictive Control

A fairly general optimal control in the S&H setting can be formulated as follows:

$$\begin{aligned} \min_{\{u_i\}} J_{OC}(x_0|\{u_i\}) &:= \sum_{i=1}^H \gamma^{i-1} \rho(\hat{x}_{i|0}, u_i), \\ \text{s.t. } \hat{x}_{2|i} &= \Phi(s\delta, \hat{x}_{1|i}, u_i), \hat{x}_{1|0} = x_0, \\ \mathcal{D}^+ x &= f(x, u^\delta), \end{aligned} \quad (2)$$

where ρ is the stage cost (a reward or utility in case of maximization), J_{OC} is the accumulated stage cost, where γ is the discounting factor, H is the horizon length which can be finite ($H = N, N \in \mathbb{N}$) or infinite ($H = \infty$), s is the prediction step size multiplier, i.e., the prediction step size is $s\delta$, Φ is a numerical integration scheme, i.e., $\hat{x}_{2|i} = \Phi(\delta, \hat{x}_{1|i}, u_i)$ is the predicted state emerging from $\hat{x}_{1|i}$ after the time δ and under the constant action u_i . If $H = \infty$, J_{OC} is also known as cost-to-go. The simplest numerical integration scheme is the Euler one:

$$\hat{x}_{2|i} := \hat{x}_{1|i} + s\delta f(\hat{x}_{1|i}, u_i). \quad (3)$$

In the following, the system dynamics $\mathcal{D}^+ x = f(x, u^\delta)$ are always meant, but omitted for brevity. Based on H , two basic optimal control formalisms are generally known, namely, Euler-Lagrange and Hamilton-Jacobi-Bellman [48]. Euler-Lagrange formalism possesses a “local” character – it seeks a controller that optimizes the cost some N steps ahead starting from the current state. Hamilton-Jacobi-Bellman, in contrast, is “global” – the goal here is to find a controller for cost optimization over an indefinite number of future steps. An infinite horizon may also be interpreted as an open horizon – a situation, in which the user is unsure of an exact specification of the horizon. In turn, a finite-horizon optimal control problem may be interpreted as a computationally tractable approximation of the infinite-horizon one. MPC is the de facto scheme for finite-horizon optimal control problems [49]–[51]. Here, the infinite horizon is cut at some finite time and an optimal solution is computed for the new fixed horizon at each time step. Numerous modifications and a wide variety of techniques for guaranteeing closed-loop stability of MPC were developed [52], [53]. In the S&H setting, a simple unconstrained MPC setup can be written as:

$$\begin{aligned} \min_{\{u_{i|k}\}_i^N} J_{MPC}(x_k|\{u_{i|k}\}_i^N) &:= \sum_{i=1}^N \gamma^{i-1} \rho(\hat{x}_{i|k}, u_{i|k}), \\ \text{s.t. } \hat{x}_{i+1|k} &= \Phi(s\delta, \hat{x}_{i|k}, u_{i|k}). \end{aligned} \quad (4)$$

When requiring constraint satisfaction of the kind $x_{i|k} \in \mathbb{X}, u_{i|k} \in \mathbb{U}$, MPC has the advantage of guaranteed safety [51]. Also, various schemes for stabilization guarantees are known [50], [51]. Evidently, a simple MPC controller is suboptimal, if the suboptimality is meant as the difference between the factual cost-to-go under the MPC controller and the optimized cost-to-go (the value function). This is somewhat in contrast to the philosophy behind RL where an agent seeks to approximate the value function. In general,

Algorithm 1 roll-out Q-learning

Input: System model, sampling time δ , prediction step multiplier s , prediction horizon N

while true do

 Get state x_k

 Push the current state-action pair into the buffer (experience replay)

 Update critic: $\vartheta_k := \arg \min_{\vartheta} J_k^c(\vartheta)$ (see eq. 6)

 Update actor: $\{u_{i|k}\}_i^N := \min_{\{u_{i|k}\}_i^N} J_{\text{RQL}}^a(x_k | \{u_{i|k}\}_i^N; \vartheta_k) = \sum_{i=1}^{N-1} \gamma^{i-1} \rho(\hat{x}_{i|k}, u_{i|k}) + \hat{Q}(\hat{x}_{N|k}, u_{N|k}; \vartheta_k)$, where the state

 sequence $\{\hat{x}_{i|k}\}_i^N$ is predicted via, e. g., (3)

 Apply the first action from the sequence, namely, $u_{1|k}$, to the system

end while

enlarging the horizon leads to suboptimality reduction [54]. As mentioned above, careful tuning is required in general. The next section discusses specifically fusion of MPC-elements with RL.

IV. FUSION OF RL AND PREDICTIVE CONTROLS

Value iteration Q-learning (QL) actor-critic is chosen here as the basis for RL algorithms due to its convenience, although similar derivations could be done for the standard value and policy iteration. A basic online, model-free, value iteration, on-policy QL with a neural network critic reads:

$$\begin{aligned} u_k &:= \arg \min_u \hat{Q}(x_k, u; \vartheta_k), \\ \vartheta_k &:= \arg \min_{\vartheta} \frac{1}{2} (\hat{Q}(x_k, u_k; \vartheta) - \hat{Q}(x_{k-1}, u_k; \vartheta^-) - \rho(x_k, u_k))^2, \end{aligned} \quad (5)$$

where ϑ is vector of the critic neural network weights to be optimized, ϑ^- is the vector of the weights from the previous time step, $\hat{Q}(\bullet, \bullet; \vartheta)$ – Q-function approximation parameterized by ϑ . The latter approximation is effectively the temporal difference (TD) in the value iteration form. It may be generalized to a custom size experience replay. Let $e_k(\vartheta) = \vartheta \varphi(x_{k-1}, u_{k-1}) - \gamma \vartheta^- \varphi(x_k, u_k) - \rho(x_{k-1}, u_{k-1})$ denote the temporal difference at time step k . Then, a more general critic cost function may be formulated as

$$J_k^c(\vartheta) = \frac{1}{2} \sum_{i=k}^{k+M-1} e_i^2(\vartheta), \quad (6)$$

where M is the buffer size.

The *roll-out QL* (RQL) considered here, given the MPC background of Section III-A, can be regarded as simply $N-1$ horizon MPC with a terminal cost being the Q-function approximation, namely, its actor reads:

$$\begin{aligned} \min_{\{u_{i|k}\}_i^N} J_{\text{RQL}}^a(x_k | \{u_{i|k}\}_i^N; \vartheta_k) &:= \\ \sum_{i=1}^{N-1} \gamma^{i-1} \rho(\hat{x}_{i|k}, u_{i|k}) + \hat{Q}(\hat{x}_{N|k}, u_{N|k}; \vartheta_k), & \quad (7) \\ \text{s.t. } \hat{x}_{i+1|k} &= \Phi(s\delta, \hat{x}_{i|k}, u_{i|k}). \end{aligned}$$

The *stacked QL* (SQL) [55], [56], in turn, can be regarded as simply MPC with the stage cost substituted for Q-function approximation. The justification for such a setup may be

done via Lemma 1. It says that the optimal policy from optimization of a stacked Q-function is essentially the same as the globally optimal one. First, denote the stacked Q-function as follows:

$$\bar{Q}(x_k, \{u_{i|k}\}_i^N) := \sum_{i=1}^N Q(x_{i|k}, u_{i|k}). \quad (8)$$

With this notation at hand, proceed to the lemma.

Lemma 1: For any x_k , it holds that

$$\min_{\{u_{i|k}\}_i^N} \bar{Q}(x_k, \{u_{i|k}\}_i^N) = \sum_{i=1}^N \min_{u_{i|k}} Q(x_{i|k}, u_{i|k}). \quad (9)$$

Proof: Let the optimal action sequence for the stacked Q-learning be denoted as:

$$\begin{aligned} \{\bar{u}_{i|k}^*\}_i^N &:= \arg \min_{\{u_{i|k}\}_i^N} \bar{Q}(x_k, \{u_{i|k}\}_i^N) \\ &= \sum_{i=1}^N Q(x_{i|k}, u_{i|k}), \end{aligned} \quad (10)$$

The optimal action sequence of the element-wise Q-function optimization reads:

$$\{u_{i|k}^*\}_i^N := \{\arg \min_{u_{i|k}} Q(x_{i|k}, u_{i|k})\}_i. \quad (11)$$

Denote the corresponding optimal state sequences as $\{\bar{x}_{i|k}^*\}_i^N := \{\bar{x}_k^*, \bar{x}_{k+1}^*, \dots, \bar{x}_{k+N-1}^*\}$ for the stacked Q-learning and $\{x_{i|k}^*\}_i^N := \{x_k^*, x_{k+1}^*, \dots, x_{k+N-1}^*\}$ for the ordinary one. Notice that the initial state is the same: $\bar{x}_k^* = x_k^* = x_k$. By definition the optimal Q-function is equal to the value function V :

$$V(x_{i|k}^*) = \min_u Q(x_{i|k}^*, u), \quad (12)$$

$$V(\bar{x}_{i|k}^*) = \min_u Q(\bar{x}_{i|k}^*, u). \quad (13)$$

Denote

$$\bar{V}(\bar{x}_{i|k}^*) := Q(\bar{x}_{i|k}^*, \bar{u}_{i|k}^*). \quad (14)$$

By the principle of optimality it holds that

$$V(\bar{x}_{i|k}^*) \leq \bar{V}(\bar{x}_{i|k}^*). \quad (15)$$

Algorithm 2 Stacked Q-learning

Input: System model, sampling time δ , prediction step multiplier s , prediction horizon N
while *true* **do**
 Get state x_k
 Push the current state-action pair into the buffer (experience replay)
 Update critic: $\vartheta_k := \arg \min_{\vartheta} J_k^c(\vartheta)$ (see eq. 6)
 Update actor: $\{u_{i|k}\}_i^N := \min_{\{u_{i|k}\}_i^N} J_{\text{SQL}}^a(x_k | \{u_{i|k}\}_i^N; \vartheta_k) = \sum_{i=1}^N \hat{Q}(\hat{x}_{i|k}, u_{i|k}; \vartheta_k)$, where the state sequence $\{\hat{x}_{i|k}\}_i^N$ is predicted via, e. g., (3)
 Apply the first action from the sequence, namely, $u_{1|k}$, to the system
end while

But since $x_{i|k}^*$ is the optimal state sequence,

$$V(x_{i|k}^*) \leq V(\bar{x}_{i|k}^*). \quad (16)$$

Thus, $V(x_{i|k}^*) \leq \bar{V}(\bar{x}_{i|k}^*)$ and applying sum to the both parts yields

$$\sum_{i=1}^N V(x_{i|k}^*) \leq \sum_{i=1}^N \bar{V}(\bar{x}_{i|k}^*) = \min_{\{u_{i|k}\}_i^N} \bar{Q}(x_k, \{u_{i|k}\}_i^N). \quad (17)$$

On the other hand, the minimum of the sum is not greater than the sum of minima:

$$\min_{\{u_{i|k}\}_i^N} \bar{Q}(x_k, \{u_{i|k}\}_i^N) \leq \sum_{i=1}^N V(x_{i|k}^*). \quad (18)$$

Then, as required,

$$\begin{aligned} \min_{\{u_{i|k}\}_i^N} \bar{Q}(x_k, \{u_{i|k}\}_i^N) &= \sum_{i=1}^N V(x_{i|k}^*) = \\ &= \sum_{i=1}^N \min_{u_{i|k}} Q(x_{i|k}, u_{i|k}). \end{aligned} \quad (19)$$

Since in practice, Q-functions cannot always be computed exactly, a temporal-difference-based critic can be employed to compute approximate stacked Q-function using the temporal difference method (6). This can be done, e. g., using neural networks. Finally, the stacked QL actor reads:

$$\begin{aligned} \min_{\{u_{i|k}\}_i^N} J_{\text{SQL}}^a(x_k | \{u_{i|k}\}_i^N; \vartheta_k) &= \sum_{i=1}^N \hat{Q}(\hat{x}_{i|k}, u_{i|k}; \vartheta_k), \\ \text{s.t. } \hat{x}_{i+1|k} &= \Phi(s\delta, \hat{x}_{i|k}, u_{i|k}). \end{aligned} \quad (20)$$

The main features of the described algorithms for the clarity are given in Table I.

V. NUMERICAL EXPERIMENTS

All three described setups, namely, MPC, roll-out QL and stacked QL were studied in numerical experiments with wheeled robot parking. In every experiment, a set of hyper-parameters consisting of the horizon length N , prediction step size multiplier s , and the sampling time δ , was fixed.

An experiment consisted of 30 runs, 600 s long each. The robot started at a position on a 5 m circle around the origin, turned away from the latter. The goal was to park the robot at the origin while achieving a desired orientation. The parking was considered successful if the robot entered a 50 cm circle around the origin with a 5 deg tolerance in angle. The accumulated stage cost, as well as the successful parking count, were considered the performance metrics. The next section describes the system dynamics in detail.

A. Environment

As the dynamic system, the three-wheel robot with dynamical pushing force and steering torque (a.k.a. ENDI – extended non-holonomic double integrator) was considered.

In Cartesian coordinates system description is the following:

$$\begin{aligned} \dot{x} &= v \cos \alpha, \\ \dot{y} &= v \sin \alpha, \\ \dot{\alpha} &= \omega, \\ \dot{v} &= \left(\frac{1}{m} F\right), \\ \dot{\omega} &= \left(\frac{1}{I} M\right). \end{aligned} \quad (21)$$

where x – x -coordinate [m], y – y -coordinate [m], α – turning angle [rad], v – velocity [m/s], ω – angular velocity [rad/s], F – pushing force [N], M – steering torque [Nm], m – robot mass [kg], I – robot moment of inertia around vertical axis [kg m^2] ($m = 10$, $I = 1$).

B. Simulation

Implementation of MPC, roll-out Q-learning, and stacked Q-learning were done in a custom python framework `rcognita`¹, developed specifically for hybrid simulation of RL agents (Fig. 1).

The stage cost was considered in the following quadratic form:

$$\rho = \chi^\top R \chi, \quad (22)$$

where $\chi = [y, u]$, R diagonal, positive-definite.

¹<https://github.com/AIDynamicAction/rcognita>

TABLE I: Algorithms intuition

Name	Scheme	Description
<i>MPC</i>	Baseline	Finite sum of stage costs without a terminal cost
<i>RQL</i>	$MPC + Q_N$	Finite sum of stage costs with a Q-function as the terminal cost
<i>SQL</i>	$MPC \wedge QL : r \leftarrow Q$	Finite sum of Q-functions

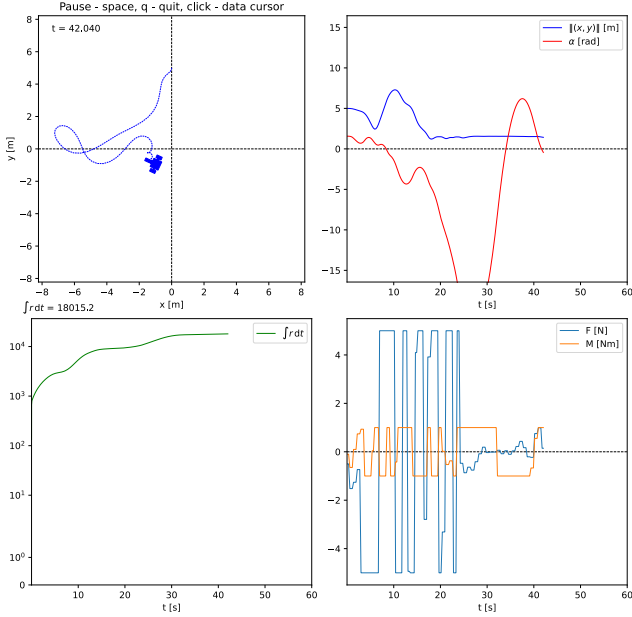


Fig. 1: Graphical output of the `rognita` Python package.

The critic structure was also chosen quadratic as follows:

$$\begin{aligned} \hat{Q}(x, u; \vartheta) &:= \vartheta \varphi^\top(x, u), \\ \varphi(x, u) &:= \text{vec}(\Delta_u([x|u] \otimes [x|u])), \end{aligned} \quad (23)$$

where ϑ – critic weights, φ – critic activation function, Δ_u – operator of taking the upper triangular matrix, vec – vector-to-matrix transformation operation, $[x|u]$ – stack of vectors x and u , \otimes – Kronecker product.

The experimental results are presented below.

VI. RESULTS AND DISCUSSION

It was observed that too low sampling time led to somewhat higher cost which can be explained by a too narrow-sighted controller. Increasing the sampling time remedies that issue, but only up to a certain point where prediction inaccuracies start to dominate, whence the cost grows again (see Fig. 2). A similar tendency can be observed in terms of successful parking count (see Fig. 3). As the prediction step size was increased, it was observed that both the accumulated stage cost and successful parking count slightly improved. This may be explained by an effective horizon enlargement (though retaining the resolution). Again, one should be aware of growing prediction errors while increasing the prediction step size. As far as the horizon length itself is concerned, there was a clear tendency of performance improvement with

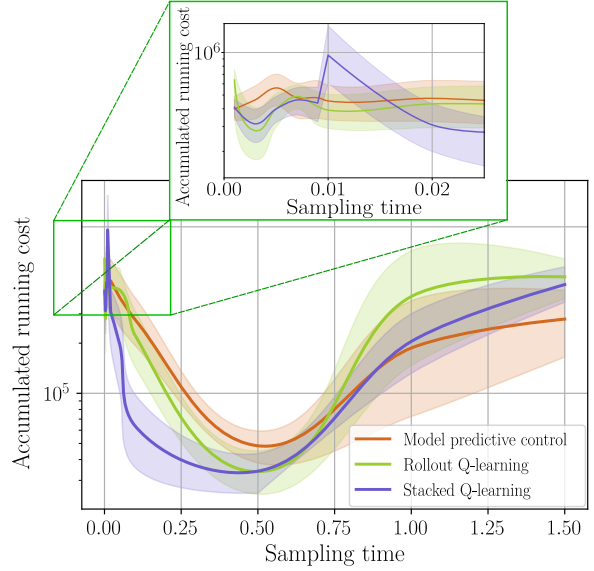


Fig. 2: Relationship between the accumulated stage cost and the sampling time. Solid line – average over 30 observations, shaded area – 95 % confidence level.

higher N . At the horizon length of 5, all the controllers succeeded to park the robot in all trials.

In terms of the algorithm comparison, some interesting phenomena could be noticed. First of all, both the roll-out and stacked QL generally outperformed MPC at shorter sampling times and horizons. Fig. 5 in turns shows far superior successful parking compared to MPC. These observations support the idea that integration of learning elements into classical controller, e. g., in the form of RL, is beneficial. In theory, the Q-function captures the performance of the agent over an infinite horizon. It seems logical that approximating the Q-function sufficiently well may yield better control actions than optimizing a plain sum of stage costs. The following hint could be made: *predictive RL is more beneficial than MPC at shorter horizons*. This is because as the horizon length grows, predictive RL becomes indistinguishable from MPC (see Fig. 6), while both simply approach the globally optimal controller. One should be aware of the computational complexity though.

Roughly, it can be described as follows. Denote the MPC complexity (in terms of optimizing J_{MPC}) by $\mathcal{O}(\Psi_{MPC}(N))$ for some function Ψ_{MPC} of the horizon length (in particular, the complexity is exponential in N). Then, the complexities of the roll-out and stacked QL are both $\mathcal{O}(\Psi_{MPC}(N)) +$

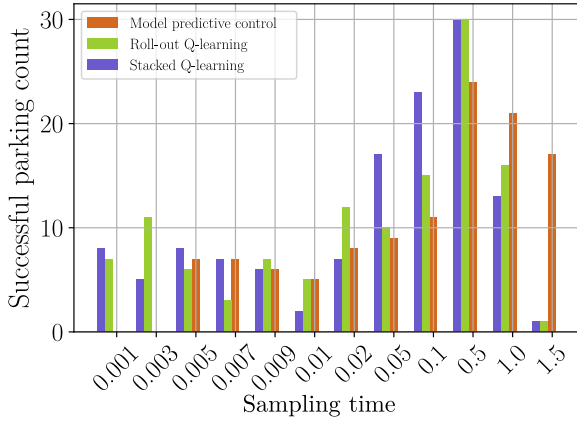


Fig. 3: Successful parking count depending on the sampling time.

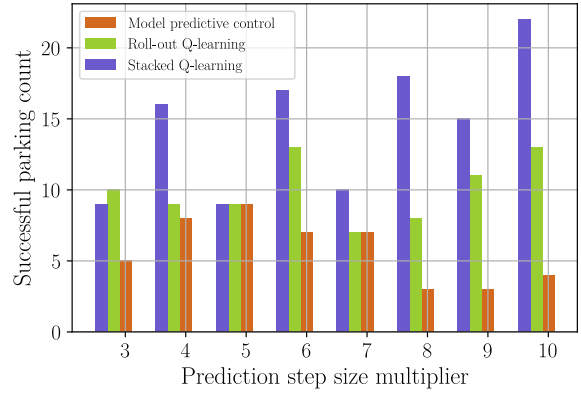


Fig. 5: Successful parking count depending on the prediction step size.

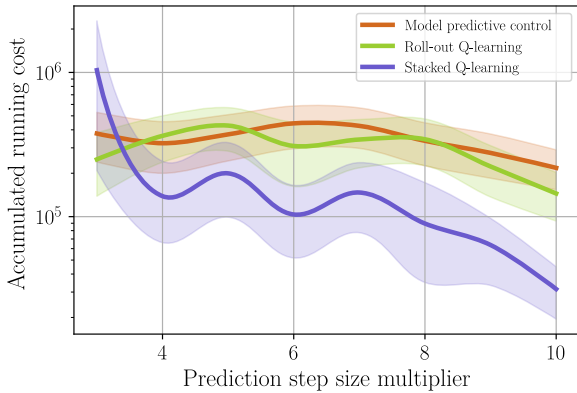


Fig. 4: Relationship between the accumulated stage cost and the prediction step size multiplier. Solid line – average over 30 observations, shaded area – 95 % confidence level.

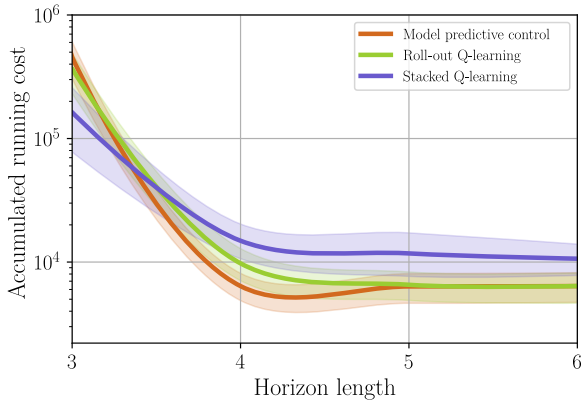


Fig. 6: Relationship between the accumulated stage cost and the prediction horizon length. Solid line – average over 30 observations, shaded area – 95 % confidence level.

$\mathcal{O}(\Gamma_{\text{QL}}(M, n_{\varphi}))$ where Γ_{QL} describes the complexity of the critic update (in terms of optimizing J_c), M is the experience replay size and n_{φ} is the number of critic weights.

A final note should be made about the difference in performance between the roll-out and stacked QL. Remarkably, the latter significantly outperformed the former at a short horizon ($N=3$), both in terms of the accumulated stage cost and successful parking count. This may be explained in a similar manner as above, when comparing RL with MPC. Namely, learning elements of RL are more beneficial at shorter horizons. Notice that the roll-out QL “retains” more from MPC than its stacked counterpart. At longer horizons, such a structure of the roll-out QL becomes more beneficial than the stacked QL. Notice also that nominally both QL methods have the same complexity. However, taking into account better performance of the stacked QL at shorter horizons, the practical complexity of the stacked QL may even be

considered lower than that of the roll-out variant. That is, the stacked QL may achieve a comparable performance to the roll-out QL with a longer horizon.

VII. CONCLUSION

As learning-based control becomes ever more attractive, it faces ever more challenges in industry, where, traditionally, such controllers as MPC are recognized due to their formal guarantees. Reinforcement learning slowly transitions from playgrounds like videogames into more challenging environments. On this path, it seems unavoidable that some of the well-established classical machinery, such as predictive control, can be made use of in RL. This is supported, in particular, by the attractive trend of fusion of MPC and RL. The current study was generally dedicated to this topic and considered a particular, yet fairly popular, control problem of parking of a mobile robot by MPC and RL. The influence of the prediction- and sampling-related hyperparameters,

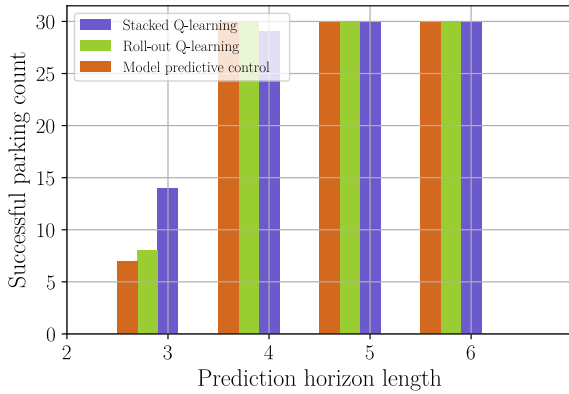


Fig. 7: Successful parking count tuning the prediction horizon length.

namely, the prediction horizon step and length sizes, and the sampling time, was investigated. It was generally observed that RL-based controllers appeared more efficient than MPC at shorter horizon lengths, where the learning-based elements dominated. Predictive RL, like the herein studied stacked Q-learning, may be considered a viable solution in terms of fusion of classical controllers and RL agents.

REFERENCES

- [1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [2] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.
- [3] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.
- [4] C. Li, "Deep reinforcement learning," in *Reinforcement Learning for Cyber-Physical Systems*. Chapman and Hall/CRC, 2019, pp. 125–154.
- [5] K. Krauth, S. Tu, and B. Recht, "Finite-time analysis of approximate policy iteration for the linear quadratic regulator," in *Advances in Neural Information Processing Systems*, 2019, pp. 8514–8524.
- [6] Z. Yang, Y. Chen, M. Hong, and Z. Wang, "Provably global convergence of actor-critic: A case for linear quadratic regulator with ergodic cost," in *Advances in Neural Information Processing Systems*, 2019, pp. 8353–8365.
- [7] Y. Park, R. Rossi, Z. Wen, G. Wu, and H. Zhao, "Structured policy iteration for linear quadratic regulator," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7521–7531.
- [8] S. Kakade, A. Krishnamurthy, K. Lowrey, M. Ohnishi, and W. Sun, "Information theoretic regret bounds for online nonlinear control," *arXiv preprint arXiv:2006.12466*, 2020.

- [9] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.
- [10] Y. Li, X. Chen, and N. Li, "Online optimal control with linear dynamics and predictions: Algorithms and regret analysis," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 887–14 899.
- [11] W. R. van Soest, Q. P. Chu, and J. A. Mulder, "Combined feedback linearization and constrained model predictive control for entry flight," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 2, pp. 427–434, 2006.
- [12] S. Kouro, P. Cortes, R. Vargas, U. Ammann, and J. Rodríguez, "Model predictive control—a simple and powerful method to control power converters," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1826–1838, 2009.
- [13] Y. Ma, F. Borrelli, B. Hencye, B. Coffey, S. Benghea, and P. Haves, "Model predictive control for the operation of building cooling systems," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 3, pp. 796–803, 2012.
- [14] S. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.
- [15] D. Hrovat, S. Di Cairano, H. E. Tseng, and I. V. Kolmanovsky, "The development of model predictive control in automotive industry: A survey," in *2012 IEEE International Conference on Control Applications*, 2012, pp. 295–302.
- [16] M. L. Darby and M. Nikolaou, "Mpc: Current practice and challenges," *Control Engineering Practice*, vol. 20, no. 4, pp. 328 – 342, 2012, special Section: IFAC Symposium on Advanced Control of Chemical Processes - ADCHEM 2009.
- [17] M. G. Forbes, R. S. Patwardhan, H. Hamadah, and R. B. Gopaluni, "Model predictive control in industry: Challenges and opportunities," *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 531 – 538, 2015, 9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015.
- [18] D. P. Bertsekas and D. A. Castanon, "Rollout algorithms for stochastic scheduling problems," *Journal of Heuristics*, vol. 5, no. 1, pp. 89–108, 1999.
- [19] D. P. Bertsekas, "Dynamic programming and suboptimal control: A survey from ADP to MPC," *European Journal of Control*, vol. 11, no. 4-5, pp. 310–334, 2005.
- [20] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *International Conference on Learning Representations*, 2020.
- [21] Z. Guo, A. B. Pires, G. M. Azar, B. Piot, F. Althé, J.-B. Grill, and R. Munos, "Bootstrap latent-predictive representations for multitask reinforcement learning," *International Conference on Machine Learning*, 2020.
- [22] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018, pp. 8289–8300.
- [23] D. Hoeller, F. Farshidian, and M. Hutter, "Deep value model predictive control," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100, 2020, pp. 990–1004.
- [24] S. East, M. Gallieri, J. Masci, J. Koutnik, and M. Cannon, "Infinite-horizon differentiable model predictive control," *International Conference on Learning Representations*, 2020.
- [25] S. Reddy, A. D. Dragan, S. Levine, S. Legg, and J. Leike, "Learning human objectives by evaluating hypothetical behavior," *International Conference on Machine Learning*, 2019.
- [26] N. Karnchanachari, M. de la Iglesia Valls, D. Hoeller, and M. Hutter, "Practical reinforcement learning for mpc: Learning from sparse objectives in under an hour on a real robot," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, ser. Proceedings of Machine Learning Research, A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, Eds., vol. 120. The Cloud: PMLR, 2020, pp. 211–224.
- [27] M. Cannon and B. Kouvaritakis, "Optimizing prediction dynamics for robust mpc," *IEEE Transactions on Automatic Control*, vol. 50, no. 11, pp. 1892–1897, 2005.
- [28] M. J. Tippett, C. K. Tan, and J. Bao, "Non-constant prediction-step mpc for processes with multi-scale dynamics," *IFAC Proceedings*

- Volumes, vol. 47, no. 3, pp. 3068 – 3073, 2014, 19th IFAC World Congress.
- [29] W. Wojsznis, J. Gudaz, T. Blevins, and A. Mehta, “Practical approach to tuning MPC,” *ISA Transactions*, vol. 42, no. 1, pp. 149–162, 2003.
- [30] K. Worthmann, “Estimates of the prediction horizon length in MPC: a numerical case study,” *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 232–237, 2012.
- [31] J. Sawma, F. Khatounian, E. Monmasson, R. Ghosn, and L. Idkhajine, “The effect of prediction horizons in MPC for first order linear systems,” in *2018 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2018.
- [32] C. A. Silva and J. I. Yuz, “On sampled-data models for model predictive control,” in *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2010.
- [33] X. Zhang, S. Grammatico, G. Schildbach, P. Goulart, and J. Lygeros, “On the sample size of randomized mpc for chance-constrained systems with application to building climate control,” in *2014 European Control Conference (ECC)*, 2014, pp. 478–483.
- [34] P. K. Khosla, “Effect of sampling rates on the performance of model-based control schemes,” in *Dynamics of Controlled Mechanical Systems*. Springer, 1989, pp. 271–284.
- [35] M. Łaskawski and M. Weislik, “Influence of sampling on the tuning of pid controller parameters,” *IFAC-PapersOnLine*, vol. 48, no. 4, pp. 430 – 435, 2015, 13th IFAC and IEEE Conference on Programmable Devices and Embedded Systems.
- [36] C. Tallec, L. Blier, and Y. Ollivier, “Making deep q-learning methods robust to time discretization,” in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97, Long Beach, California, USA, 2019, pp. 6096–6104.
- [37] S. R. Sinclair, T. Wang, G. Jain, S. Banerjee, and C. L. Yu, “Adaptive discretization for model-based reinforcement learning,” *arXiv*, 2020.
- [38] M. Malisoff and E. Sontag, “Asymptotic controllability and input-to-state stabilization: The effect of actuator errors,” in *Optimal control, stabilization and nonsmooth analysis*. Springer, 2004, pp. 155–171.
- [39] F. Clarke, “Discontinuous feedback and nonlinear systems,” *IFAC Proceedings Volumes*, vol. 43, no. 14, pp. 1–29, 2010.
- [40] P. Braun, L. Grüne, and C. M. Kellett, “Feedback design using nonsmooth control lyapunov functions: A numerical case study for the nonholonomic integrator,” in *IEEE Conf. Decis. Control*. IEEE, 2017, pp. 4890–4895.
- [41] S. I. Abbasi W., urRehman F., “Backstepping based nonlinear adaptive control for the extended nonholonomic double integrator,” *Kybernetika*, vol. 53, no. 4, pp. 578–594, 2017.
- [42] A. P. Aguiar and A. Pascoal, “Stabilization of the extended nonholonomic double integrator via logic-based hybrid control,” *IFAC Proceedings Volumes*, vol. 33, no. 27, pp. 351 – 356, 2000, 6th IFAC Symposium on Robot Control (SYROCO 2000), Vienna, Austria, 21–23 September 2000.
- [43] A. P. A. M. Pascoal and A. Aguiar, “Practical stabilization of the extended nonholonomic double integrator,” *Proc. 10th Mediterranean Conference on Control and Automation*, 2002.
- [44] F. ur Rehman, “Steering control of nonholonomic systems with drift: The extended nonholonomic double integrator example,” *Nonlinear Analysis: Theory, Methods & Applications*, vol. 62, no. 8, pp. 1498 – 1515, 2005, hybrid Systems and Applications.
- [45] D. DeVon and T. Bretl, “Kinematic and dynamic control of a wheeled mobile robot,” in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 4065–4070.
- [46] K. Watanabe, T. Yamamoto, K. Izumi, and S. Maeyama, “Under-actuated control for nonholonomic mobile robots by using double integrator model and invariant manifold theory,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2862–2867.
- [47] D. Nešić, A. R. Teel, and P. V. Kokotović, “Sufficient conditions for stabilization of sampled-data nonlinear systems via discrete-time approximations,” *Systems & Control Letters*, vol. 38, no. 4–5, pp. 259–270, 1999.
- [48] J. Primbs, V. Nevistić, and J. Doyle, “Nonlinear optimal control: A control lyapunov function and receding horizon perspective,” *Asian Journal of Control*, vol. 1, no. 1, pp. 14–24, 1999.
- [49] M. Nikolaou, “Model predictive controllers: A critical synthesis of theory and industrial needs,” in *Advances in Chemical Engineering*. Elsevier, 2001, pp. 131–204.
- [50] L. Grüne, “Nmpe without terminal constraints,” *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 1 – 13, 2012, 4th IFAC Conference on Nonlinear Model Predictive Control.
- [51] D. Q. Mayne, “Model predictive control: Recent developments and future promise,” *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.
- [52] H. Chen and F. Allgöwer, “A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability,” *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.
- [53] P. Scokaert, D. Mayne, and J. Rawlings, “Suboptimal model predictive control (feasibility implies stability),” *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 648–654, 1999.
- [54] L. Grüne and A. Rantzer, “On the infinite horizon performance of receding horizon controllers,” *IEEE Transactions on Automatic Control*, vol. 53, no. 9, pp. 2100–2111, 2008.
- [55] P. Osinenko, T. Göhr, G. Devadze, and S. Streif, “Stacked adaptive dynamic programming with unknown system model,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4150–4155, 2017.
- [56] L. Beckenbach, P. Osinenko, T. Gohrt, and S. Streif, “Constrained and stabilizing stacked adaptive dynamic programming and a comparison with model predictive control,” in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 1349–1354.