Multi-step Greedy Reinforcement Learning Based on Model Predictive Control

Yucheng Yang * **Sergio Lucia** **

* *Department of Electrical Engineering and Computer Science, TU Berlin, Einsteinufer 17, 10587 Berlin, Germany (e-mail: yucheng.yang@campus.tu-berlin.de).*
** *Laboratory of Process Automation Systems, TU Dortmund University, Emil-Figge-Str. 70, 44227 Dortmund, Germany (e-mail:sergio.lucia@tu-dortmund.de)*

Abstract: Reinforcement learning aims to compute optimal control policies with the help of data from closed-loop trajectories. Traditional model-free approaches need huge number of data points to achieve an acceptable performance, rendering them not applicable in most real situations, even if the data can be obtained from a detailed simulator. Model-based reinforcement learning approaches try to leverage model knowledge to drastically reduce the amount of data needed or to enforce important constraints to the closed-loop operation, which is another important drawback of model-free approaches. This paper proposes a novel model-based reinforcement learning approach. The main novelty is the fact that we exploit all the information of a model predictive control (MPC) computing step, and not only the first input that is actually applied to the plant, to efficiently learn a good approximation of the state value function. This approximation can be included into a model predictive control formulation as a terminal cost with a short prediction horizon, achieving a similar performance to an MPC with a very long prediction horizon. Simulation results of a discretized batch bioreactor illustrate the potential of the proposed methodology.

Keywords: Reinforcement learning, Model predictive control, Nonlinear system, Batch reactor

1. INTRODUCTION

Reinforcement Learning (RL) is an area of machine learning concerned with how agents have to take actions in Markov Decision Processes (MDP) in order to optimize a cumulative reward. In Sutton et al. (1992), one of the founding fathers of modern computational reinforcement learning, Richard S. Sutton, already pointed out that reinforcement learning is actually direct adaptive optimal control. *Direct* is to describe that most RL methods are model-free, which means that they do not need to depend on the model of the system dynamics; *Adaptive* is to describe the data-driven nature of RL. These ideas had been previously applied also in the field of process control, mostly under the keywords of adaptive optimal control and approximate dynamic programming, but it has recently gained and increased attention (Shin et al., 2019), (Spielberg et al., 2017), (Kim et al., 2020).

There are mainly two categories of reinforcement learning: value-based methods and policy-based methods. Value-based RL such as Q-learning (Mnih et al., 2013) typically try to approximate the optimal state-action function, $Q^*(x, a)$, which denotes the expected least cumulative cost of taking action a at state x. The optimal policy at a state x is then the minimizer of $Q^*(x, a)$ over all possible actions a. Policy-based methods try to approximate the optimal policy function $\pi(x)$ directly. They typically update their approximation based on the intuitive idea that if an action is followed by less cumulative cost, then the tendency of taking that action is strengthened (or *reinforced*). Most policy-based methods use policy gradient (Sutton et al., 2000) to update their parameters. Both categories have remarkable achievements in the field of of board games and modern computer games.

One major problem of RL is that most *direct* (model-free) RL methods usually apply black boxes methods (e.g. artificial neural networks) for function approximation, resulting in non-interpretable behaviours for which it is also hard to enforce safety critical constraints. In many application domains, such as process control, the lack of constraint handling, can prevent the application of RL methods. To enforce constraint satisfaction of the state and action trajectories generated by a reinforcement learning policy, some recent works have combined Model Predictive Control (MPC) with RL: (Gros and Zanon, 2019a), (Gros and Zanon, 2019b), (Zanon and Gros, 2020). By solving the constraint optimization problem of MPC, a safe policy can be obtained.

For RL, sample efficiency is another major problem, the learning process often needs a large amount of data. Some approaches of safe reinforcement learning (e.g. in Wabersich and Zeilinger (2018)) use a model predictive filter to project the unsafe actions generated by model-free RL controller to a safe action set. These approaches need the knowledge of model dynamics for the predictive filter, but this model knowledge is not efficiently exploited for the training of the model-free RL policy. Furthermore,

Gros et al. (2020) has shown that the projected policy might not be the optimal policy with safety constraint satisfaction. By using MPC, a relatively good initial policy at the beginning of the learning process can be obtained by exploiting the model knowledge. When the optimal value function can be exactly approximated, the policy generated by the constraint optimization of MPC would be optimal. The combination of MPC and RL can have, besides the rigorous consideration of constraints, other advantages. For example, since MPC is in practice implemented with a finite prediction horizon, which can lead to significant suboptimal performance, an RL approach can be used to approximate the infinite-horizon cost, leading to an improved performance.

In this paper, based on the interesting ideas proposed in Gros and Zanon (2019b), Zanon and Gros (2020) we aim to provide a novel contribution which improves previous methods by exploiting more information from the model knowledge and the MPC computation. Our proposed method, which assumes knowledge of the model, aims to learn the parametric state value function, which can be used as the terminal cost of MPC leading to an improved performance despite of a short prediction horizon. Different from the quasi-infinite horizon MPC methods, such as the one described in Chen and Allgöwer (1998), where the terminal cost that determined from an assumed linear state feedback is an upper bound of the optimal value function, our proposed approaches tries to learn the exact optimal value function from sampled data. In addition, our methods also work for cases with time varying references.

This work focuses on showing that the proposed method has good theoretical and computational properties that make it a good candidate to work in practice. The application to realistic case studies with uncertain models is part of our future work.

2. BACKGROUND

This section introduces some common concepts and notations related to reinforcement learning.

2.1 Markov Decision Process

Reinforcement learning usually considers that the dynamics of the real system are described by a Markov Decision Processes (MDP). A countable MDP is defined as a triplet $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathcal{P})$, where $\mathcal{X}$ is the nonempty set of states and the nonempty set of actions is denoted by $\mathcal{A}$. The state transition probability $\mathcal{P}$ assigns to each state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$ a probability over $\mathcal{X}$, which can be denoted by $\mathcal{P}(\cdot\,|x, a)$. For example, after taking decision a at state x, the probability of reaching a new state x' is $\mathcal{P}(x'|x, a)$.

After defining a stage cost function l that maps each state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$ to a value in $\mathbb{R}$, a transition step of MDP can be denoted by a tuple (x, a, x', l). This means that after taking decision a at state x, the system comes to a new state x' and gets a stage cost $l(x, a)$.

2.2 Concepts and notations

In this paper, we consider the learning goal of finding a policy which minimizes the expected cumulative stage cost. For clarity in the presentation, we summarize some common concepts in reinforcement learning and optimal control that will be used in the remainder of the paper.

(1) **Policy:** normally denoted as $\pi(\cdot\,|x)$. If policy π is a applied to a system, then at state x, the probability to take action a is $\pi(a|x)$.

(2) **V-value:** V-value is also known as state value function. The V-value for a given policy π is defined as

$$V^\pi(x) := \mathbb{E}_{\pi,\mathcal{P}}\Big[\sum_{k=0}^{\infty} \gamma^k l(x_k, a_k)\Big], \tag{1}$$

where γ is the discount factor and $x_0 = x$. The discount factor γ is a real number satisfying $0 < \gamma < 1$. Its purpose is to make this sum of stage costs bounded. $\mathbb{E}_{\pi,\mathcal{P}}$ means it is an expectation with respect to the policy π and the transition probability $\mathcal{P}$. The V-value $V^\pi(x)$ is also called the V-function.

(3) **Q-value:** Q-value is also known as state action value function. The Q-value for a given policy π is defined as

$$Q^\pi(x, a) := \mathbb{E}_{\pi,\mathcal{P}}\Big[\sum_{k=0}^{\infty} \gamma^k l(x_k, a_k)\Big], \tag{2}$$

where $x_0 = x$ and $a_0 = a$. The Q-value function $Q^\pi(x, a)$ is also called the Q-function.

(4) **Greedy policy:** The greedy policy of a Q-function Q is defined as

$$\pi(u|x) = 1, \text{ with } u = \underset{a}{\operatorname{argmin}}\, Q(x, a). \tag{3}$$

(5) **Bellman equation:** For a given V-function, the Bellman equation with respect to a given π is

$$\mathcal{T}^\pi V(x) = \mathbb{E}_{\pi,\mathcal{P}}\Big[l(x, a) + \gamma V(x')\Big], \tag{4}$$

where a and x' are random variables of $\pi, \mathcal{P}$. The notation $\mathcal{T}^\pi V$ means to apply $\mathcal{T}^\pi V(x)$ for all $x \in \mathcal{X}$. For the Q-function, one gets

$$\mathcal{T}^\pi Q(x, a) = l(x, a) + \gamma \mathbb{E}_{\pi,\mathcal{P}}[Q(x', a')]. \tag{5}$$

The notation $\mathcal{T}^\pi Q$ means to apply $\mathcal{T}^\pi Q(x, a)$ for all $x \in \mathcal{X}$ and all $a \in \mathcal{A}$.

(6) **Bellman optimality equation:** For the V-function, Bellman optimality equation is

$$\mathcal{T}^* V(x) = \min_a \Big[l(x, a) + \gamma \mathbb{E}_{\mathcal{P}}[V(x')|x, a]\Big], \tag{6}$$

where $\mathbb{E}_{\mathcal{P}}[\cdot\,|x, a]$ means an expectation of $V(x')$ with respect to the conditional probability $\mathcal{P}(x'|x, a)$. The notation $\mathcal{T}^* V$ means to apply $\mathcal{T}^* V(x)$ for all $x \in \mathcal{X}$. For the Q-function, one gets

$$\mathcal{T}^* Q(x, a) = l(x, a) + \gamma \mathbb{E}_{\mathcal{P}}[\min_{a'} Q(x', a')|x, a], \tag{7}$$

where $\mathbb{E}_{\mathcal{P}}[\cdot\,|x, a]$ means an expectation of $V(x')$ with respect to the conditional probability $\mathcal{P}(x'|x, a)$. The notation $\mathcal{T}^* Q$ means to apply $\mathcal{T}^* Q(x, a)$ for all $x \in \mathcal{X}$ and all $a \in \mathcal{A}$.

From Szepesvári (2010), we know that $\mathcal{T}^\pi$ and $\mathcal{T}^*$ are so-called contraction mappings. For a contraction mapping $\mathcal{T}$ in a Banach space $\mathbb{B}$, it holds that for all $V_1, V_2 \in \mathbb{B}$, $\|\mathcal{T}V_1 - \mathcal{T}V_2\| \leq \|V_1 - V_2\|$.

For a contraction mapping $\mathcal{T}$ and a sequence of functions V_k or Q_k the following Banach fixed-point theorem holds (Szepesvári, 2010).

Theorem 1 (Banach fixed-point theorem). *Let $(\mathcal{X}, d)$ be a non-empty complete metric space with a contraction mapping $\mathcal{T} : \mathcal{X} \to \mathcal{X}$. Then $\mathcal{T}$ admits a unique fixed-point X^* in $\mathcal{X}$ (i.e. $\mathcal{T}X^* = X^*$). Furthermore, X^* can be found as follows: start with an arbitrary element X_0 in $\mathcal{X}$ and define a sequence $\{X_n\}$ by $X_n = \mathcal{T}X_{n-1}$ for $n \geq 1$. Then $X_n \to X^*$.*

Proof. See (Szepesvári, 2010).

This means that one can recursively apply $V_{k+1} = \mathcal{T}V_k$ or $Q_{k+1} = \mathcal{T}Q_k$, and then when k goes to infinity, V_k or Q_k will converge to the fixed-point of the operator $\mathcal{T}$, which satisfies $V_k = \mathcal{T}V_k$ or $Q_k = \mathcal{T}Q_k$. The equation $Q_k = \mathcal{T}Q_k$ means $Q_k(x, a) = \mathcal{T}Q_k(x, a)$ for all $(x, a) \in \mathcal{X} \times \mathcal{A}$.

2.3 Value iteration and Q-learning

Value-based RL methods such as Q-learning are based on value iteration. Here we briefly introduce value iteration and Q-learning.

For our goal, the optimal policy π^* should be the one minimizing expected sum of stage costs.

$$\pi^* = \underset{\pi}{\operatorname{argmin}}\ \mathbb{E}_{\pi,\mathcal{P}}\Big[\sum_{k=0}^{\infty} \gamma^k l(x_k, a_k)\Big], \tag{8}$$

A Q-function Q^* satisfying $\mathcal{T}^*Q^* = Q^*$ is the fixed-point of the Bellman optimality equation and the greedy policy associated to Q^* would be π^*.

Value iteration is to learn Q^* by recursively applying the Bellman optimality equation. In a value iteration step, $V_{k+1} = \mathcal{T}^*V_k$ or $Q_{k+1} = \mathcal{T}^*Q_k$ is applied for all $(x, a) \in \mathcal{X} \times \mathcal{A}$. By Banach fixed point theorem, the value function will converge to the optimal value function V^* or Q^*, satisfying $V^* = \mathcal{T}^*V^*$ or $Q^* = \mathcal{T}^*Q^*$

Notice that for every iteration, the value over all input space should be updated, which is not feasible for large input spaces.

Q-learning is a modified version of value iteration, which is feasible for large state-action spaces. Instead of updating the Q-value of all state action pairs in one iteration, for every iteration it only samples one previously encountered transition tuple (x, a, x', l) and applies

$$Q_{k+1}(x, a) \leftarrow Q_{k+1}(x, a) + \delta, \tag{9}$$

$$\text{with: } \delta = l + \min_{a'} Q_k(x', a') - Q_k(x, a) \tag{10}$$

where δ is called the temporal difference error (TD error) and the left arrow $\leftarrow$ means to update the value $Q_{k+1}(x, a)$ towards the target value $l + \min_{a'} Q_k(x', a')$. When there is enough exploration, i.e. every state action pair can be sampled with positive probability, Q^* can be learned, enabling the learning of Q^* for larger state-action spaces.

As we have shown, value-based methods make use of the Bellman optimality equation to learn the optimal fixed point V^* or Q^*. The fact that the operator $\mathcal{T}^*$ contracts the value function at each iteration towards a unique fixed point is essential. Therefore, any novel methods that uses as update an operator different from $\mathcal{T}^*$, should prove that the new operator also contracts the value function to the optimal Q^*.

3. MPC PROBLEM FORMULATION

We consider discrete-time nonlinear systems. We assume that the full state can be measured at each sampling time and that we control the system using discrete-time MPC with a parametric terminal cost. The MPC problem to be solved at time t can be written as:

$$\begin{aligned}
U_\theta(x, a, t) :=& \\
&\min_{z} \sum_{k=0}^{N-1} \gamma^k l_{t+k}(z_k, u_k) + \gamma^N V_\theta(z_N, t+N) && \text{(11a)}\\
&\text{s.t. } z_0 = x, \quad u_0 = a, && \text{(11b)}\\
&\qquad z_{k+1} = f(z_k, u_k),\ k \in \mathbb{I}_0^{N-1} && \text{(11c)}\\
&\qquad g(z_k, u_k) \leq 0,\ k \in \mathbb{I}_0^{N-1} && \text{(11d)}\\
&\qquad g_T(z_N, u_N) \leq 0 && \text{(11e)}\\
\pi_\theta(x, t) :=& \underset{a'}{\operatorname{argmin}}\ U_\theta(x, a', t) && \text{(11f)}
\end{aligned}$$

where $\boldsymbol{z} := \{z_0, u_0, ..., u_{N-1}, z_N\}$, or more specifically $\boldsymbol{z}_\theta(x, t) := \{z_0, u_0, ..., u_{N-1}, z_N|(x, t)\}$ is the predicted trajectory at (x, t), l_τ is the stage loss at time τ, θ is the learnable parameter, f represents the dynamics, g is the constraints and g_T is the constraint for recursive feasibility.

The parameterized terminal cost is denoted by $V_\theta(\,\cdot\,,\,\cdot\,)$ and has two arguments, the state and the time step. The goal of the proposed reinforcement learning scheme is to learn the optimal V_θ, so that this MPC with finite prediction horizon N (potentially very short) has the same performance of an MPC with approximately infinite prediction horizon, while guaranteeing that the constrains are satisfied as enforced by (11d).

4. PROPOSED APPROACH

4.1 Update Description

At time step t with state x, after action a is taken, we get a stage cost $l_t(x, a)$ and reach a new state x'. We propose to update V_θ by using the obtained stage cost as well as the full result of the MPC computation defined by (11):

$$\begin{aligned}
V_{\theta_{\text{new}}}(x, t) \leftarrow\ & l_t(x, a) + \\
& \gamma U_\theta(x', \pi_\theta(x', t+1), t+1).
\end{aligned} \tag{12}$$

The term $U_\theta(x', \pi_\theta(x', t+1), t+1)$, in the right hand side of (12) is obtained from MPC computation at $(x', t+1)$.

Besides differences in terms of parameterization, this update has different theoretical basis from the method proposed in Zanon and Gros (2020), which is using Bellman optimality equation (6) for Q-learning. And it is also different from the method of Gros and Zanon (2019a), which is using Bellman equation (4) for value function evaluation. This update is neither using Bellman equation (4) nor Bellman optimality equation (6), so we should prove that it will lead to some fixed point and this fixed point can provide optimality. In the next subsection, we will show that (12) leads V_θ to the fixed point of Bellman optimality equation (6). In addition, unlike the previously mentioned methods, an important advantage of our proposed method is that it does not require complex sensitivity calculations, which means that more complex parameterizations can be used as function approximations (such as neural networks),

opening the door for the consideration of significantly more challenging problems.

4.2 Optimality of the proposed update

As introduced in Section 2.3, Q-learning learns the optimal state action value function Q^* because it takes advantage of the Bellman optimality operation $\mathcal{T}^*$, which has a fixed point at Q^*. Here we want to show that (12) also has similar effect and leads to the fixed point V^*, and by inserting $V_\theta = V^*$ into the MPC objective (34a), the MPC-computed cost (U_θ) becomes Q^*.

We define an operator that performs the proposed update for all $x \in \mathcal{X}$ and $t \in \mathcal{I}$, where $\mathcal{I}$ is the set of time indices, as:

$$\mathcal{T}^v V(x,t) = l_t(x, \pi_V(x,t)) + \gamma U_V(x', \pi_V(x', t+1), t+1). \tag{13}$$

Subscript V of π_V and U_V is to show that they are obtained by inserting a state value function V into the MPC formulation of (11).

Assumption 1. *A perfect model of the system dynamics is available and the terminal constraints describe a feasible control invariant set.*

Assumption 2. *V satisfies for all $x_t \in \mathcal{X}$ and $t \in \mathcal{I}$:*

$$V(x_t,t) \geq \min_{(x_t,a_t,x_{t+1})\in\mathcal{F}} \Big\{ l_t(x_t,a_t) + \gamma V(x_{t+1}, t+1) \Big\} \tag{14}$$

Where $\mathcal{F}$ denotes the set of feasible transitions.

We can first get a V satisfying this assumption by applying $\mathcal{T}^\pi$ of an arbitrary policy π to a initial V_{init}.

Theorem 2. *Under Assumption 1 and 2, a unique fixed point of $\mathcal{T}^v V = V$ exists and the fixed point can be reached by recursively updating V with $\mathcal{T}^v V$. Moreover, If $\mathcal{T}^v V = V$, then $\mathcal{T}^* U = U = Q^*$.*

Proof. First, we define for all $x_t \in \mathcal{X}$ and $t \in \mathcal{I}$:

$$\begin{aligned}
\mathcal{T}^{(N)} V(x_t,t) :=& \min_{(x_k,a_k,x_{k+1})\in\mathcal{F}} \Big\{ \sum_{k=t}^{t+N-1} \gamma^{k-t} l_k(x_k,a_k) + \\
& \gamma^N V(x_{t+N}, t+N) \Big\} \\
\mathcal{T}^{(N+1)} V(x_t,t) :=& \min_{(x_k,a_k,x_{k+1})\in\mathcal{F}} \Big\{ \sum_{k=t}^{t+N} \gamma^{k-t} l_k(x_k,a_k) + \\
& \gamma^{N+1} V(x_{t+N+1}, t+N+1) \Big\}.
\end{aligned} \tag{15}$$

When Assumption 2 holds for V we have:

$$\begin{aligned}
\mathcal{T}^{(N)} V(x_t,t) =& \min_{(x_k,a_k,x_{k+1})\in\mathcal{F}} \Big\{ \sum_{k=t}^{t+N-1} \gamma^{k-t} l_k(x_k,a_k) + \\
& \gamma^N V(x_{t+N}, t+N) \Big\} \\
\geq& \min_{(x_k,a_k,x_{k+1})\in\mathcal{F}} \Big\{ \sum_{k=t}^{t+N} \gamma^{k-t} l_k(x_k,a_k) + \\
& \gamma^{N+1} V(x_{t+N+1}, t+N+1) \Big\} \\
=& \min_{(x_k,a_k,x_{k+1})\in\mathcal{F}} \Big\{ l_t(x_t,a_t) + \\
& \gamma \mathcal{T}^{(N)} V(x_{t+1}, t+1) \Big\}
\end{aligned} \tag{16}$$

So when Assumption 2 holds for V, it also holds for $\mathcal{T}^{(N)}V$, and similarly it holds for $\mathcal{T}^{(N+1)}V$ as well.

Expanding $\mathcal{T}^{(N)} V(x_t,t)$, we get:

$$\begin{aligned}
& \mathcal{T}^{(N)} V(x_t,t) \\
=\ & l_t(x_t, \pi_V(x_t,t)) + \\
& \gamma \min_{(x_k,a_k,x_{k+1})\in\mathcal{F}} \Big\{ \sum_{k=t+1}^{t+N-1} \gamma^{k-(t+1)} l_k(x_k,a_k) + \\
& \gamma^{N-1} V(x_{t+N}, t+N) \Big\} \\
\geq\ & l_t(x_t, \pi_V(x_t,t)) + \\
& \gamma \min_{(x_k,a_k,x_{k+1})\in\mathcal{F}} \Big\{ \sum_{k=t+1}^{(t+1)+N-1} \gamma^{k-(t+1)} l_k(x_k,a_k) + \\
& \gamma^{N} V(x_{(t+1)+N}, (t+1)+N) \Big\} \\
=\ & l_t(x_t, \pi_V(x_t,t)) + \gamma U_V(x_{t+1}, \pi_V(x_{t+1}, t+1), t+1) \\
=\ & \mathcal{T}^v V(x_t,t)
\end{aligned} \tag{17}$$

Under Assumption 1, the model used for prediction is the same as the real dynamics, and by Assumption 2 the inequality holds, so the last equation of (17) holds.

By (17) we have $\mathcal{T}^{(N)} V \geq \mathcal{T}^v V$. Expanding $T^v V$ we have for all $x_t \in \mathcal{X}$ and $t \in \mathcal{I}$:

$$\begin{aligned}
\mathcal{T}^v V(x_t,t) =& l_t(x_t, \pi_V(x_t,t)) + \\
& \min_{(x_k,a_k,x_{k+1})\in\mathcal{F}} \Big\{ \sum_{k=t+1}^{t+N} \gamma^{k-t} l_k(x_k,a_k) + \\
& \gamma^{N+1} V(x_{t+N+1}, t+N+1) \Big\} \\
\geq& \mathcal{T}^{(N+1)} V(x_t,t) \\
=& \min_{(x_t,a_t,x_{t+1})\in\mathcal{F}} \Big\{ l_t(x_t,a_t) + \gamma \mathcal{T}^N V(x_{t+1}, t+1) \Big\} \\
\geq& \min_{(x_t,a_t,x_{t+1})\in\mathcal{F}} \Big\{ l_t(x_t,a_t) + \gamma \mathcal{T}^v V(x_{t+1}, t+1) \Big\}
\end{aligned} \tag{18}$$

The equation in (18) also holds because of Assumption 1, the second inequality holds by (17)

By (16) and (18) we have when Assumption 2 holds for a initial V, when recursively updating V with $\mathcal{T}^v V$, $\mathcal{T}^N V$ or $\mathcal{T}^{N+1} V$, the resulting V always satisfies Assumption 2. Therefore in value iteration setting, we only need to start with a initial V satisfying Assumption 2.

By (17) and (18):

$$\mathcal{T}^{(N)} V \geq \mathcal{T}^v V \geq \mathcal{T}^{(N+1)} V \tag{19}$$

Next, we prove $\mathcal{T}^{(N)}$ and $\mathcal{T}^{(N+1)}$ have the same fixed point as $\mathcal{T}^*$.

From (Szepesvári, 2010, Thm. 2 and Eq. (42)), we know that

$$\mathcal{T}^* V(x,t) = \min_{(x,a,x')\in\mathcal{F}} \{ l_t(x,a) + \gamma V(x', t+1) \}$$

is a contraction mapping of γ-contraction and its fixed point is the optimal state value function. $\mathcal{T}^{(N)} V$ and $\mathcal{T}^{(N+1)} V$ can be proven to be contraction mappings of γ^N-contraction and γ^{N+1}-contraction respectively using the same way of proof for $\mathcal{T}^*$ in Szepesvári (2010). By the

Banach fixed-point Theorem, $\mathcal{T}^{(N)}V$ and $\mathcal{T}^{(N+1)}V$ have unique fixed points. When V^* is a solution of $\mathcal{T}^*V = V$, we have for all $x_t \in \mathcal{S}$ and $t \in \mathcal{I}$:

$$\begin{aligned}\mathcal{T}^{(N+1)}V^*(x_t,t) =& \min_{(x_k,a_k,x_{k+1})\in\mathcal{F}}\Big\{\sum_{k=t}^{t+N}\gamma^{k-t}l_k(x_k,a_k)+\\ &\gamma^{N+1}V^*(x_{t+N+1},t+N+1)\Big\}\\ =& \min_{(x_k,a_k,x_{k+1})\in\mathcal{F}}\Big\{\sum_{k=t}^{t+\infty}\gamma^{k-t}l_k(x_k,a_k)\Big\}\\ =&V^*(x_t,t).\end{aligned} \tag{20}$$

So V^* is also a fixed point of $\mathcal{T}^{(N+1)}$. Similarly, we can prove $\mathcal{T}^{(N)}V^* = V^*$. Because $\mathcal{T}^{(N)}$ and $\mathcal{T}^{(N+1)}$ both have unique fixed points, they have the same fixed point as $\mathcal{T}^*$.

Now we prove that V^* is also a fixed point of $\mathcal{T}^v$. V^* satisfies Assumption 2 by the definition, so we can apply inequality (19), for all $x_t \in \mathcal{X}$ and $t \in \mathcal{I}$:

$$\begin{aligned}\mathcal{T}^{(N+1)}V^*(x_t,t) \le& \mathcal{T}^vV^*(x_t,t) \le \mathcal{T}^{(N)}V^*(x_t,t)\\ V^*(x_t,t) \le& \mathcal{T}^vV^*(x_t,t) \le V^*(x_t,t)\\ &\mathcal{T}^vV^*(x_t,t) = V^*(x_t,t)\end{aligned} \tag{21}$$

We have already proved that if $V = V^*$, then V is a fixed point of $\mathcal{T}^v$. In order to prove uniqueness, we need to prove that for all V^v, which is a fixed point of $\mathcal{T}^v$, they are the same as the unique fixed point V^* of $\mathcal{T}^*$.

For every V^v satisfying $\mathcal{T}^vV^v = V^v$, we have for all $x_t \in \mathcal{X}$ and $t \in \mathcal{I}$:

$$\begin{aligned}\mathcal{T}^vV^v(x_t,t) &= l_t(x_t,a_t) + \mathcal{T}^vV^v(x_{t+1},t+1)\\ &\ge \min_{(x_k,a_k,x_{k+1})\in\mathcal{F}}\Big\{l_t(x_t,a_t) + \mathcal{T}^vV^v(x_{t+1},t+1)\Big\}\end{aligned} \tag{22}$$

(22) means Assumption 2 holds for every V^v. Then we can get $\mathcal{T}^{(N)}V^v \le \mathcal{T}^*V^v \le V^v$ for all $x_t \in \mathcal{X}$ and $t \in \mathcal{I}$:

$$\begin{aligned}\mathcal{T}^{(N)}V(x_t,t) =& \min_{(x_k,a_k,x_{k+1})\in\mathcal{F}}\Big\{\sum_{k=t}^{t+N-1}\gamma^{k-t}l_k(x_k,a_k)+\\ &\gamma^{t+N}V^v(x_{t+N},t+N)\Big\}\\ \le& \min_{(x_t,a,x_{t+1})\in\mathcal{F}}\{l_t(x_t,a)+\gamma V^v(x_{t+1},t+1)\}\\ =&\mathcal{T}^*V^v(x_t,t) \le V^v(x_t,t)\end{aligned} \tag{23}$$

By inequality (19) and (23), :

$$\begin{aligned}V^v = \mathcal{T}^vV^v \le& \mathcal{T}^{(N)}V^v \le \mathcal{T}^*V^v \le V^v\\ V^v \le& \mathcal{T}^*V^v \le V^v\\ &\mathcal{T}^*V^v = V^v\end{aligned} \tag{24}$$

So every fixed point of $\mathcal{T}^v$ is also the unique fixed point of $\mathcal{T}^*$.

Then we prove that for a sequence V_n, $n \in \mathcal{N}$, V_0 satisfying Assumption 2, by iteratively applying $V_{n+1} = \mathcal{T}^vV_n$, the limit of this sequence is V^*.

For all V and all $x_t \in \mathcal{X}$, $t \in \mathcal{I}$, by the fact that V^* is also the unique fixed-point of $\mathcal{T}^v$:

$$|\mathcal{T}^vV(x_t,t) - \mathcal{T}^vV^*(x_t,t)| = |\mathcal{T}^vV(x_t,t) - V^*(x_t,t)| \tag{25}$$

By (19), when $\mathcal{T}^vV(x_t,t) - V^*(x_t,t) \ge 0$:

$$\begin{aligned}|\mathcal{T}^vV(x_t,t) - V^*(x_t,t)| &= \mathcal{T}^vV(x_t,t) - V^*(x_t,t)\\ &\le \mathcal{T}^{(N)}V(x_t,t) - V^*(x_t,t)\\ &\le |\mathcal{T}^{(N)}V(x_t,t) - V^*(x_t,t)|\\ &\le \|\mathcal{T}^{(N)}V - V^*\|_\infty\\ &\le \gamma^N\|V - V^*\|_\infty\end{aligned} \tag{26}$$

The last inequality of (26) is from the fact that $\mathcal{T}^{(N)}$ is γ^N-contraction. When $\mathcal{T}^vV(x_t,t) - V^*(x_t,t) < 0$, also by (19) and the fact that $\mathcal{T}^{(N+1)}$ is γ^{N+1}-contraction:

$$\begin{aligned}|\mathcal{T}^vV(x_t,t) - V^*(x_t,t)| &= V^*(x_t,t) - \mathcal{T}^vV(x_t,t)\\ &\le V^*(x_t,t) - \mathcal{T}^{(N+1)}V(x_t,t)\\ &\le |\mathcal{T}^{(N+1)}V(x_t,t) - V^*(x_t,t)|\\ &\le \|\mathcal{T}^{(N+1)}V - V^*\|_\infty\\ &\le \gamma^{N+1}\|V - V^*\|_\infty\\ &\le \gamma^N\|V - V^*\|_\infty\end{aligned} \tag{27}$$

Combining (26) and (27), we know that for all V and all $x_t \in \mathcal{S}$, $t \in \mathcal{I}$:

$$|\mathcal{T}^vV(x_t,t) - V^*(x_t,t)| \le \gamma^N\|V - V^*\|_\infty \tag{28}$$

So:

$$\|\mathcal{T}^vV - V^*\|_\infty \le \gamma^N\|V - V^*\|_\infty \tag{29}$$

By (29) we know that under Assumption 1 by iteratively applying (12), we can get V^*.

Finally, we show that inserting V^* into the MPC under Assumption 1 results in the optimal policy.

If V^* is inserted in the MPC problem under Assumption 1, the objective of the MPC at x, t becomes:

$$\begin{aligned}\min_{(x_k,a_k,x_{k+1})\in\mathcal{F}}\Big\{&\sum_{k=t}^{t+N-1}\gamma^{k-t}l_k(x_k,a_k)+\\ &\gamma^NV^*(x_{t+N},t+N)\Big\},\end{aligned} \tag{30}$$

which is equivalent to:

$$\min_{(s,a,s')\in\mathcal{F}}\{l_t(x,a)+\gamma V^*(x',t+1)\} = \min_a\{Q^*(x,a,t)\}. \tag{31}$$

The value Q^* in (31) is the solution of $\mathcal{T}^*Q = Q$, so the policy π from the MPC is the greedy policy of Q^*. □

With Theorem 2, we show that an optimal policy can be learned when the model is known and enough exploration of the state action space occurs. In practice, with a nominal model, which does not exactly describe the real dynamics, a robust MPC scheme such as (Mayne et al., 2011; Lucia et al., 2013) can be used to ensure that the constraints are satisfied also in the presence of uncertainty. Here we focus on the theoretical optimality of the proposed method and future work will tackle the case of uncertain models as well as the removal of Assumption 2. In the following section, a simulation study shows the advantages of the proposed update.

5. SIMULATION RESULTS

We use a control task of a nonlinear system to demonstrate that the proposed method can obtain the performance of

MPC with a large prediction horizon by solving a short-horizon MPC with the proposed reinforcement learning strategy.

The considered model of the batch bioreactor is continuous and has 4 states and 1 control input. The four states are concentration of the biomass X_s, the concentration of the substrate S_s, the concentration of the product P_s and the volume V_s. The control input u_{inp} is the feed flow rate of S_s. The system model is described by the ordinary differential equations:

$$\begin{aligned}\dot{X}_s &= \mu(S_s)X_s - \frac{u_{inp}}{V_s}X_s,\\ \dot{S}_s &= -\frac{\mu(S_s)X_s}{Y_x} - \frac{vX_s}{Y_p} + \frac{u_{inp}}{V_s}(S_{in} - S_s),\\ \dot{P}_s &= vX_s - \frac{u_{inp}}{V_s}P_s,\\ \dot{V}_s &= u_{inp},\end{aligned} \tag{32}$$

where:

$$\mu(S_s) = \frac{\mu_m S_s}{K_m + S_s + (S_s^2/K_i)}, \tag{33}$$

The values for the different parameters, such that the inlet substrate concentration S_{in}, the kinetic parameters μ_m, K_m, K_i , v and the yield coefficients Y_x, Y_p as well as more information about the process can be found in Srinivasan et al. (2003). The parameters Y_x and S_{in} are set to be the nominal values ($Y_x = 0.5$ and $S_{in} = 200$).

The objective is to maximize the concentration of product after 150 hours of production. The constraints are 0 l/h $\leq u_{inp} \leq 1$ l/h and $X_s \leq 3.7$ g/l.

The model described by (32) is discretized using Euler method with a sampling time of 1 hour. We collect all the states X_s, S_s, P_s, V_s in the vector x. Using f to denote the Euler discretized system, the optimization problem to be solved at each sampling time for the parametric MPC can be written as:

$$U_\theta(x, a, t) :=$$

$$\min_z \sum_{k=0}^{N-1} \gamma^k l_{t+k}(z_k, u_k, u_{k-1}) + \gamma^N V_\theta(z_N, t+N) \tag{34a}$$

$$\text{s.t.} \quad z_0 = x, \quad u_0 = a, u_{-1} = u_{last} \tag{34b}$$

$$z_{k+1} = f(z_k, u_k), \ k \in \mathbb{I}_0^{N-1} \tag{34c}$$

$$Gz_k \leq 3.7, \ k \in \mathbb{I}_0^N \tag{34d}$$

$$0 \leq u_k \leq 1, \ k \in \mathbb{I}_0^N \tag{34e}$$

$$\pi_\theta(x, t) := \underset{a'}{\text{argmin}}\ Q_\theta(x, a', t) \tag{34f}$$

where u_{last} is the input a at the last time step $t-1$, and

$$l_{\tau+k}(z_k, u_k, u_{k-1}) \tag{35}$$

$$= 0.18 + z^T Q z + (u_k - u_{k-1})^T R (u_k - u_{k-1}) \tag{36}$$

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -0.1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \ R = 0.5, \tag{37}$$

$$\tag{38}$$

and

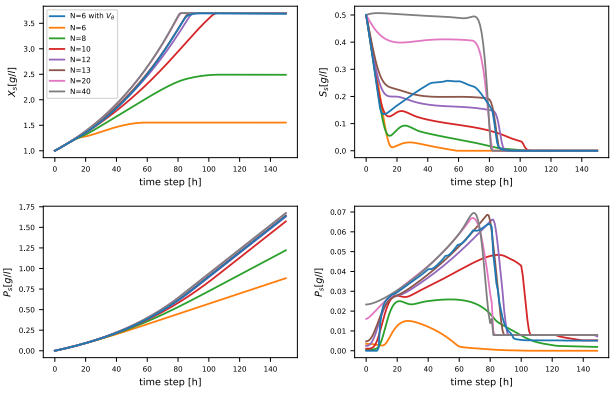

Fig. 1. Comparison of states and inputs

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{39}$$

We run the method proposed in Section 4 for 50 episodes, with initial state is $x_0 = [1\ , 0.5\ , 0\ , 120]^T$ and prediction horizon $N = 6$. We choose the parametric terminal cost V_θ to be a neural network with *tanh* activation, 5 inputs, 1 output and two hidden layers of 20 and 30 neurons respectively. Common strategies for value evaluation in reinforcement learning like Schulman et al. (2015) can be used for faster learning. Once the training has been performed, we compare the performance of our method to MPC schemes with different prediction horizons without terminal cost. The results can be seen in Fig. 1. It can be seen that the proposed method with an RL-based terminal cost and a horizon $N = 6$ achieves a significantly better performance than the standard MPC scheme with $N = 6$ or even higher.

Table 1 shows the comparisons of total losses and average warm start computation time (ACT) in ms between MPC trials with (marked in bold) and without learned terminal cost, showing the computational and performance advantages of the proposed method.

Table 1. Comparison of computation time and total loss

N	6	7	8	9	10	11
ACT(ms)	6.36	6.70	7.12	7.57	7.97	8.82
Total loss	23.35	22.25	20.84	19.29	17.95	17.40
N	12	13	14	20	40	6+V_θ
ACT(ms)	9.40	9.74	10.35	12.83	29.09	**7.14**
Total loss	17.06	16.84	16.69	16.38	16.33	**16.94**

6. CONCLUSION

We proposed a method for combining model predictive control and reinforcement learning. The method effectively utilizes the full computation of the MPC for the learning process. The theoretical optimality of this method is analysed and proved. An experiment to control a discrete nonlinear system is evaluated, and the results show that the parametric terminal cost learned by this method increases the performance of a finite horizon MPC without adding significant computational time.

The theoretical analysis and simulations of this paper are both dealing with nominal trajectories, where the

model can be considered to be exactly known. Future work will focus on the study the performance of the approach when the assumed model is uncertain and a detailed performance comparison with other recently proposed reinforcement learning schemes based on MPC.

REFERENCES

Chen, H. and Allgöwer, F. (1998). A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10), 1205–1217.

Gros, S. and Zanon, M. (2019a). Towards safe reinforcement learning using nmpc and policy gradients: Part i-stochastic case. *arXiv preprint arXiv:1906.04057*.

Gros, S. and Zanon, M. (2019b). Towards safe reinforcement learning using nmpc and policy gradients: Part ii-deterministic case. *arXiv preprint arXiv:1906.04034*.

Gros, S., Zanon, M., and Bemporad, A. (2020). Safe reinforcement learning via projection on a safe set: How to achieve optimality? *arXiv preprint arXiv:2004.00915*.

Kim, J.W., Park, B.J., Yoo, H., Oh, T.H., Lee, J.H., and Lee, J.M. (2020). A model-based deep reinforcement learning method applied to finite-horizon optimal control of nonlinear control-affine system. *Journal of Process Control*, 87, 166–178.

Lucia, S., Finkler, T., and Engell, S. (2013). Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23(9), 1306–1319.

Mayne, D.Q., Kerrigan, E.C., Van Wyk, E., and Falugi, P. (2011). Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, 21(11), 1341–1353.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Shin, J., Badgwell, T.A., Liu, K.H., and Lee, J.H. (2019). Reinforcement learning–overview of recent progress and implications for process control. *Computers & Chemical Engineering*, 127, 282–294.

Spielberg, S.P.K., Gopaluni, R.B., and Loewen, P.D. (2017). Deep reinforcement learning approaches for process control. In *6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, 201–206.

Srinivasan, B., Bonvin, D., Visser, E., and Palanki, S. (2003). Dynamic optimization of batch processes: Ii. role of measurements in handling uncertainty. *Computers & chemical engineering*, 27(1), 27–44.

Sutton, R.S., Barto, A.G., and Williams, R.J. (1992). Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, 12(2), 19–22.

Sutton, R.S., McAllester, D.A., Singh, S.P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1), 1–103.

Wabersich, K.P. and Zeilinger, M.N. (2018). Safe exploration of nonlinear dynamical systems: A predictive safety filter for reinforcement learning. *arXiv preprint arXiv:1812.05506*.

Zanon, M. and Gros, S. (2020). Safe reinforcement learning using robust mpc. *IEEE Transactions on Automatic Control*, Available Online.