Model Predictive Control Guided Reinforcement Learning Control Scheme

Huimin Xie^1, Xinghai Xu^1, Yuling Li^2, Wenjing Hong^1, Jia Shi^{1*}
^{1}Department of Chemical and Biochemical Engineering, Xiamen University, Xiamen, China
^{2}Department of Earth Science and Engineering, Imperial College London, London, UK
{onewarmhear, xinghai_xu}@163.com, yuling.li19@imperial.ic.uk, {whong, jshi}@xmu.edu.cn

Abstract—**Deep Reinforcement Learning (DRL) is an artificial intelligence technology that can complete decision-making tasks by interaction. It has been successfully applied to various games. However, there are still many challenges when this technique is applied to the industrial process control due to the low sample efficiency and the inability to deal with large time delay. In this paper, a novel Model Predictive Control (MPC) guided Reinforcement Learning Control (MP-RLC) scheme is proposed for the process control. In this scheme, Model predictive control is directly combined with Reinforcement Learning (RL) to guide the training process, thus greatly improving the sample efficiency of reinforcement learning and effectively solving the problem of time delay. The simulation results on both a third-order linear system and a nonlinear continuous stirred tank reactor (CSTR) system with large time delay demonstrate that this scheme can not only accelerate the training process but also improve the control performance, which is superior to both standalone RL and MPC schemes. The proposed approach may help to pave the way for DRL applied to industrial processes.**

Keywords—Deep reinforcement learning, Model predictive control, Time delay, Process control.

I. Introduction

Reinforcement Learning (RL) is an artificial intelligence technology that has been studied in different fields for a long time, such as process control [1], robotics [2] and power systems [3], etc., and it has attracted widespread attention in recent years. RL can learn an optimal closed-loop control only through interacting with the environment, and it possesses two features superior to the conventional feedback control. One is RL can learn an optimal control almost without any prior knowledge required. The other is RL can work as a direct adaptive optimal control for nonlinear systems [4]. In recent years, with the rapid development of Deep Learning (DL) methods, Deep Reinforcement Learning (DRL) leverages Deep Neural Networks (DNNs) as its function approximator [5] which brings RL with the capability of controlling not only systems with high dimensional input and output but also complex nonlinear systems. Recently, DRL has attained great success in computer games and board games. For instance, human-level control has been achieved in video games [5], and even human experts were defeated by DRL in GO game [6].

Researchers have done some pioneering works trying to apply DRL or similar methods in chemical process control. In the 1990s, Hoskins et al. [1] used improved AHC (Adaptive Heuristic Critic), essentially a RL algorithm combined with neural networks, to deal with the process control problem. Lee et al. [7-9] developed a series of ADP (Approximate Dynamic Programming or Adaptive Dynamic Programming) algorithms for process control tasks. ADP is a type of RL algorithm often studied in the control system community [10], and it leverages function approximators like neural networks to overcome the shortcomings of dynamic programming. Recently, Spielberg et al. [11] developed a DRL control scheme, called Deep Deterministic Policy Gradient (DDPG) [12], and gave the test results on the numerical linear and nonlinear processes to demonstrate the feasibility and effectiveness of DRL, and then Ma et al. [13] extended the application of a similar method to a numerical polymerization process. More recently, Petsagkourakis et al. [14] managed to use DRL for a real batch bioprocess. All of these studies show that the development and application of RL technology in process control are attracting wide attention.

However, there are still some challenges in applying RL to actual industrial process control. Originated from trial-and-error learning, so DRL usually needs a lot of exploratory interactions with the environment, which may result in low sample efficiency and high consuption of time or resources that may not be allowed in practices [15]. Also, RL is an optimization algorithm based on Markov Decision Processes (MDP). But for most of the actual industrial processes, due to the inevitable time delay and slow response, the Markov property of the process dynamics cannot be guaranteed, that is, the current action not only influences the next state (and reward) but also the later ones. Although this problem can be solved by extending the state with historical data [11, 13], it also leads to slower convergence with a high dimension.

To improve the sample efficiency of RL, several different ways has been proposed [15], and for process control tasks, one obvious and appropriate way is to employ a conventional feedback control scheme to guide the sample and training processes of RL. For instance, Guide Policy Search (GPS) developed by Levine et al.[16] uses the optimal control to guide the policy search of RL, which greatly improves the sample efficiency of the RL. For the time delay problem, a predictive model is usually helpful by estimating the delayed time. Meanwhile, the predictive model usually can make RL more data-efficient because richer information besides reward is used. Comprehensively, considering the two solutions above, Model Predictive Control (MPC), which is widely used in process industries, could be used to guide RL's training process in this scenario.

* Corresponding author. This work is supported by NSFC(61573295), National Key R&D Program of China (2017YFA0204902) and Tencent AI Lab Rhino-Bird Focused Research Program.

In this paper, a novel Model Predictive Control guided Reinforcement Learning Control (MP-RLC) scheme is proposed. In this control scheme, the action of the DRL is directly guided by the MPC control law in a simple but very efficient way. Through the numerical simulations on a linear system and the nonlinear continuous stirred tank reactor (CSTR) system with large time delay, it is demonstrated that the proposed MP-RLC scheme has advantages in two aspects: First, the sample efficiency of RL is improved significantly. Second, MP-RLC can achieve better control performance even on a non-linear system with a large time delay.

Despite that the basic idea of this work is closely related to some model-based RL algorithm combined with model predictive control [17-19], the way of the guidance proposed in this paper is quite different. For example, in MPC-Guided Policy Search [18], the guidance is offline and based on the MPC runtime data, however, the guiding method proposed in this paper is online and directly based on the action of MPC, which makes the implementation of the algorithm simpler and results in the parallel control structure of MPC and RL during the training phase. Besides, PILCO [17] and GP-MPC [19] leverage Gaussian Processes as the predictive model and especially for GP-MPC, it explicitly uses MPC. They both achieve good sample (data) efficiency. However, our method is not limited to MPC, it is a framework offering a potential way to combine any conventional control scheme with the RL, therefore it is more practical in the scenario of industrial process control.

The rest of the paper is organized as follows: Section II provides the preliminaries of RL and MPC. Section III presents the MP-RLC algorithm. Tests on numerical systems and discussions are presented in Section IV, aiming to verify the effectiveness of the scheme. Section V gives conclusions and perspectives.

II. Preliminaries

In this paper, MPC is employed to guide RL to complete control tasks, thus there are two control methods concerned in our scheme. In this section, the brief introductions about RL and MPC together with the corresponding algorithms used are given.

A. Reinforcement Learning(RL)

RL is a data-driven artificial intelligence algorithm, originated from trial-and-error learning. It realizes optimal decision-making through continuous interactions with the environment. From the viewpoint of control theory, RL can be regarded as a direct adaptive optimal control [4]. Fig. 1 shows the basic framework of RL.

There are two core elements in the RL algorithm. One is the agent which is working as a self-optimized controller, another is the environment E which is everything except for the agent. At every timestep t, the agent executes the control action $a(t)$ according to the states $s(t)$ observed from the environment, then the environment responses to the action and provides the scalar reward $r(t)$ based on the performance. The policy $\pi : S \to A$, a mapping from state space $S \in \mathbb{R}^d$ to action space $A \in \mathbb{R}^m$, represents how the agent acts in the environment. The aim of the agent is to learn an optimal policy π^* to maximize the cumulative rewards.

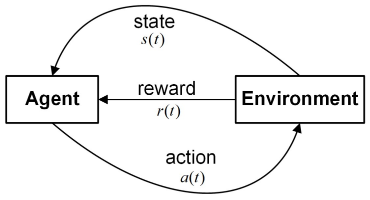

Fig. 1. The framework of Reinforcement Learning

In this paper, a DRL algorithm, Deep Deterministic Policy Gradient (DDPG) [12], is employed, which is developed from the Deterministic Policy Gradient (DPG) method [20]. It offers a deterministic policy and has shown excellent performance in the control tasks with continuous action spaces.

DDPG inherits the framework of the Actor-Critic algorithm [21], and it adopts two important techniques of Deep Q-Network (DQN) [22]. One is a replay buffer used to minimize the correlations of the training data, and the other is the separate target networks which are helpful for a stable and robust training process. Thus, there are four networks, including a policy network (Actor), a value network (Critic) and their own target networks.

Mathematically, the training process of DDPG is to update the value network and policy network alternatively based on the optimality. The final aim of DDPG is to train an optimal deterministic policy network π^*, which can maximize the following expected discounted return [23].

$$J(\theta) = \mathbb{E}_\pi \left[G(1) \right] \tag{1}$$

where $G(t) = \sum_{k=t}^{T} \gamma^{k-t} r(k)$. $G(t)$, called return, refers to the (discounted) cumulative rewards from timestep t to T (for continous control tasks $T = \infty$), γ is the discount factor and $\mathbb{E}_\pi[\cdot]$ refers to the expected random variable given that the agent follows policy π. Silver et al. [20] has proved that the deterministic policy gradient $\nabla_{\theta^\pi} J$ is the following expected gradient of action-value function Q parameterized by θ^Q.

$$\begin{aligned} \nabla_{\theta^\pi} J &\approx \mathbb{E}_{s(t)\sim\rho} \, [\nabla_{\theta^\pi} Q(s, a \mid \theta^Q) \,|_{s=s(t), a=\pi(s(t)|\theta^\pi)}] \\ &= \mathbb{E}_{s(t)\sim\rho} \, [\nabla_a Q(s, a \mid \theta^Q) \,|_{s=s(t), a=\pi(s_t)} \, \nabla_{\theta^\pi} \pi(s(t) \mid \theta^\pi) \,|_{s=s(t)}] \end{aligned} \tag{2}$$

where π indicates the policy network parameterized by θ^π. ρ is the state distribution, which can be determined by a behavior policy different with π because DDPG is off-policy. The value network Q is a mapping from state $s(t)$ and action $a(t)$ to the cumulative rewards defined by:

$$\begin{aligned} Q^\pi(s(t), a(t) \mid \theta^Q) &= \mathbb{E}_\pi \left[G(t) \right] \\ &= \mathbb{E}_\pi \left[\sum_{k=t}^{T} \gamma^{k-t} r(k) \right] \end{aligned} \tag{3}$$

The loss of value network is as follow:

$$L(\theta^Q) = \mathbb{E}_{s(t)\sim\rho^\beta, a(t)\sim\beta, r(t)\sim E}[(Q(s(t),a(t)\mid\theta^Q) - y(t))^2] \quad (4)$$

where

$$y(t) = r(s(t),a(t)) + \gamma Q'(s(t+1),a(t+1)\mid\theta^{Q'}) \quad (5)$$

The target value function is denoted by Q'. To ensure the stability of the training process, the target networks' parameters $\theta^{Q'}$ and $\theta^{\pi'}$ are updated by exponential moving average.

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \quad (6)$$

$$\theta^{\pi'} \leftarrow \tau\theta^\pi + (1-\tau)\theta^{\pi'} \quad (7)$$

where $\tau \in [0,1]$ is an update rate.

The pseudo-code of a simplified DDPG algorithm is summarized in Algorithm 1. More details refer to Lillicrap et al. [12]

Algorithm 1. DDPG

```
1. Randomly initialize the weights $\theta^Q$ of the critic
   network and $\theta^\pi$ of the policy network
2. Copy $\theta^Q$ to $\theta^{Q'}$ and $\theta^\pi$ to $\theta^{\pi'}$
3. Reset a replay buffer $R$
4. for the episode $k$ from 1 to $P$, do
5.    Get the initial state from the environment
6.    Initialize a noise $\mathcal{N}(t)$ for exploration
7.    for the step $t$ from 1 to $T$, do
8.      Get the action $a(t) = \pi(s(t)\mid\theta^\pi) + \mathcal{N}(t)$
9.      Execute the action to the environment
10.     Observe the new state $s(t+1)$ and reward $r_t$
11.     Store $(s(t),a(t),r(t),s(t+1))$ in $R$
12.     Randomly sample a batch data from $R$
13.     Set label $y(t)$ for the value network $Q$
14.     Update the value network $Q$ by minimizing the
          loss function (4)
15.     Update policy network $\pi$ based on the gradient
          $\nabla_{\theta^\pi} J$ given in (2)
16.     Update the two target networks by (6) and (7)
17.  end for
18. end for
```

B. Model Predictive Control (MPC)

MPC is an advanced control technique widely used in process industries. The basic idea of the MPC is to use the process model to predict the output of the process at every timestep, then MPC determines the optimal control sequence that minimizes an objective function defined over a receding horizon and executes only the first step of the sequence. MPC makes the closed-loop control system have better robustness than the traditional control schemes.

In this paper, we choose Generalized Predictive Control (GPC) which is a typical MPC scheme proposed by Clarke et al. [24]. For simplicity, it is assumed that a process is a single-input-single-output (SISO) system described by the following Controller Auto-Regressive Integrated Moving Average (CARIMA) model :

$$A(z^{-1})y(t) = B(z^{-1})\Delta u(t) + w(t) \quad (8)$$

where z^{-1} represents the unit backward shifting operator, A and B are the following polynomials of z^{-1} determining the dynamics of the process, $w(t)$ represents the noise or uncertainty of the model.

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n} \quad (9)$$

$$B(z^{-1}) = b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m} \quad (10)$$

We use the following notations for convenience:

$$f(t_1 : t_2) = \{f(k)\}_{k=t_1,t_1+1,\cdots,t_2}$$

$$f(|_{t_2}^{t_1}) = \begin{pmatrix} f(t_1) & f(t_1+1) & \cdots & f(t_1+t_2)\end{pmatrix}^{\mathrm{T}}$$

The objective function to minimize in GPC is a quadratic function of predictive performance over a receding horizon:

$$\begin{aligned} &J(t, u(t:t+n_2-1)) \\ &= \sum_{i=1}^{n_1} \lambda(i)(y_r(t+i) - \hat{y}(t+i\mid t))^2 + \sum_{i=1}^{n_2} \xi(i)\Delta u^2(t+i-1) \end{aligned} \quad (11)$$

where $\hat{y}(t+i\mid t)$ is the i-step ahead predictive output of the system at time step t, and y_r is the set-points, n_1 and n_2 are the prediction horizon and the control horizon, respectively. $\lambda(i)$ and $\xi(i)$ denote weighting factors.

To get the control law, we need to construct the predictive outputs first. It can be computed recursively by model (8):

$$\begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{pmatrix}\begin{pmatrix} y(|_t^{t-n+1}) \\ y(|_{t+n_1}^{t+1}) \end{pmatrix} = \begin{pmatrix}\mathbf{B}_1 & \mathbf{B}_2\end{pmatrix}\begin{pmatrix} \Delta u(|_t^{t-m+1}) \\ \Delta u(|_{t+n_1-1}^{t}) \end{pmatrix} + w(|_{t+n_1}^{t+1}) \quad (12)$$

where

$$\begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{pmatrix} = \left(\begin{array}{ccccc|ccccc} a_n & a_{n-1} & a_{n-2} & \cdots & a_1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & a_n & a_{n-1} & \cdots & a_2 & a_1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & a_n & \cdots & a_3 & a_2 & a_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & * & * & * & \cdots & a_1 & 1 \end{array}\right) \quad (13)$$

$$\begin{pmatrix}\mathbf{B}_1 & \mathbf{B}_2\end{pmatrix} = \left(\begin{array}{ccccc|ccccc} b_m & b_{m-1} & b_{m-2} & \cdots & b_2 & b_1 & 0 & \cdots & 0 & 0 \\ 0 & b_m & b_{m-1} & \cdots & b_3 & b_2 & b_1 & \cdots & 0 & 0 \\ 0 & 0 & b_m & \cdots & b_4 & b_3 & b_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & * & * & * & \cdots & b_2 & b_1 \end{array}\right) \quad (14)$$

Note that $\mathbf{A}_2$ is a non-singular matrix, then it results in

$$\mathbf{y}(|_{t+n_1}^{t+1}) = \mathbf{A}_2^{-1}\mathbf{B}_2\Delta\mathbf{u}(|_{t+n_1-1}^{t}) + \mathbf{F}(t) + \mathbf{A}_2^{-1}\mathbf{w}(|_{t+n_1}^{t+1}) \quad (15)$$

where

$$\mathbf{F}(t) = \mathbf{A}_2^{-1}\mathbf{B}_1\Delta\mathbf{u}(|_t^{t-m+1}) - \mathbf{A}_2^{-1}\mathbf{A}_1\mathbf{y}(|_t^{t-n+1}) \quad (16)$$

$\mathbf{w}(|_{k+n_1}^{k+1})$ consists of disturbance signals in the future, which is an unknown variable and here we assume it is the Gaussian white noise. Now the predictive model is defined as follows:

$$\hat{\mathbf{y}}(|_{t+n_1}^{t+1} \mid t) = \mathbf{G}\Delta\mathbf{u}(|_{t-1}^{t}) + \mathbf{F}(t) \tag{17}$$

where $\mathbf{G} = \mathbf{A}_2^{-1}\mathbf{B}_2$, $\mathbf{F}(t)$ represents the free-response when both outside disturbance signals and control increment of predictive control are zero. Equation (17) is the predictive model when n_1 equals to n_2, but if $n_1 > n_2$, $\mathbf{G}$ should be modified and the predictive model changes to

$$\hat{\mathbf{y}}(|_{t+n_1}^{t+1} \mid t) = \mathbf{G}\Delta\mathbf{u}(|_{t+n_2-1}^{t}) + \mathbf{F}(t) \tag{18}$$

The predictive model $\hat{\mathbf{y}}(|_{t+n_1}^{t+1} | t)$ is substituted into the objective function and written in matrix form,

$$\begin{aligned} &\mathbf{J}(t, u(t : t+n_2-1)) \\ &= \hat{\mathbf{e}}(|_{t+n_1}^{t+1})\mathbf{Q}\hat{\mathbf{e}}(|_{t+n_1}^{t+1} \mid t) + \Delta\mathbf{u}^T(|_{t+n_2-1}^{t})\mathbf{R}\Delta\mathbf{u}(|_{t+n_2-1}^{t}) \end{aligned} \tag{19}$$

where,

$$\hat{\mathbf{e}}(|_{t+n_1}^{t+1} \mid t) = \mathbf{y}_r(|_{t+n_1}^{t+1}) - \hat{\mathbf{y}}(|_{t+n_1}^{t+1} \mid t) \tag{20}$$

$$\mathbf{Q} = \mathrm{diag}(\lambda(1), \lambda(2), \ldots, \lambda(n_1)) \tag{21}$$

$$\mathbf{R} = \mathrm{diag}(\xi(1), \xi(2), \ldots, \xi(n_2)) \tag{22}$$

Then the control law that makes objective function reach minimum can be derived,

$$\Delta\mathbf{u}(|_{t+n_2-1}^{t}) = (\mathbf{G}^{\mathrm{T}}\mathbf{QG} + \mathbf{R})^{-1}\mathbf{G}^{\mathrm{T}}\mathbf{Q}(\mathbf{y}_r(|_{t+n_1}^{t+1}) - \mathbf{F}(t)) \tag{23}$$

In GPC, at every timestep only the first value of $\Delta\mathbf{u}(|_{t+n_2-1}^{t})$ is used. So the control law can be written as:

$$\Delta u(t) = \mathbf{K}(\mathbf{y}_r(|_{t+n_1}^{t+1}) - \mathbf{F}(t)) \tag{24}$$

where $\mathbf{K}$ refers to the first row of $(\mathbf{G}^{\mathrm{T}}\mathbf{QG} + \mathbf{R})^{-1}\mathbf{G}^{\mathrm{T}}\mathbf{Q}$.

III. MPC Guided RL Algorithm

In order to solve the problems of low sample efficiency of DRL for process control especially on systems with large time delay. We proposed MPC guided RL control scheme in this section. By using this approach, we expect to get an RL control scheme with better sample efficiency and excellent control performance even applied to the nonlinear processes with time delay.

A. Basic elements

Several basic elements, including state, action, reward and two main neural networks of the MP-RLC, should be carefully designed according to the specific task.

1) State:

For process control, the state used by RL usually consists of the real-time output information $y(t)$ and the set-point information $y_r(t)$ [25] as follows:

$$s(t) = (y(t),\ y_r(t)) \tag{25}$$

$$s(t) = (\ y(t),\ (y_r(t) - y(t))\ ) \tag{26}$$

However, when applied RL to a process with time delay, the task turns to a Partially Observed Markov Decision Process (POMDP) [26]. To satisfy the Markov property, the following extended state should be used:

$$\begin{aligned} s(t) = (&(y(t-T_y),\ y(t-T_y+1), \cdots, y(t-1)), \\ &y_r(t),\ (u(t-T_a),\ u(t-T_a+1), \ldots u(t-1))) \end{aligned} \tag{27}$$

where $u(t)$ is the control action executed on the process. T_{y} and T_a are, respectively, the extended timestep lengths of the process output and the control action.

2) Action:

The action of the DDPG is the output of the policy network, denoted by $a(t)$, while the action of GPC, determined by the control law (24), is denoted by $u^{MPC}(t)$. The action executed on the process of MP-RLC in the training phase is the combination of $a(t)$ and $u^{MPC}(t)$, which will be illustrated later. In the testing phase, the action is the output of the policy without exploration added.

3) Reward:

The reward function of RL is the key factor for the control performance of the closed-loop system. It is usually designed based on the control error. Inspired by the quadratic cost function of GPC, the reward function is designed as a truncated quadratic function as follows:

$$r(t) = \begin{cases} b(-\dfrac{e(t)^2}{\varepsilon^2} + 1), & |e(t)| \le \varepsilon \\ -c|e(t)|, & |e(t)| \ge \varepsilon \end{cases} \tag{28}$$

where $e(t) = y_r(t) - y(t)$ indicates the control error and ε is a tolerance.

4) Policy and value networks:

Both policy and value networks are 3-layer neural networks with ReLU [27] as the activation function of the hidden layer. The policy networks leverage tanh (case 1) or ReLU (case 2) as the activation function of the output layer, while the value networks' output layers have no activation function. Optimization algorithm Adam is used to train parameters of the networks and smooth L1 loss [28] is used to modify the loss of the value network in (4). The structures of the policy network and value network are shown in Fig. 2.

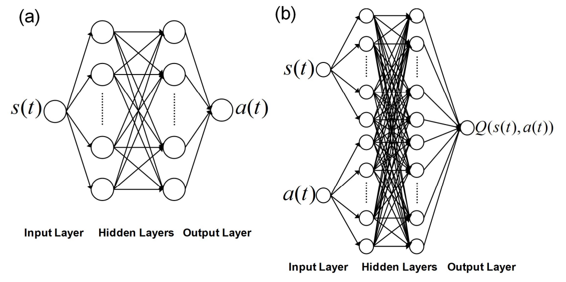

Fig. 2. The structures of the policy network (a) and the value network (b).

B. MP-RLC scheme

To guide the optimization of RL in the training phase, the control output of GPC is combined with the action output of the policy network leading to the guiding control law as follows:

$$u(t)=\beta\times u^{MPC}(t)+(1-\beta)\times a(t) \tag{29}$$

where $u(t)$ is the control action of MP-RLC and β is the weighting factor, which is a hyperparameter selected by trials.

The block diagram of the MP-RLC scheme is shown in Fig. 3. It can be seen from the figure that the whole control system consists of two parallel control loops, one is the GPC loop and the other is the DDPG control loop. In this control scheme, GPC plays a guiding role by applying its control action directly to the process to obtain more effective state and reward information used for DDPG training.

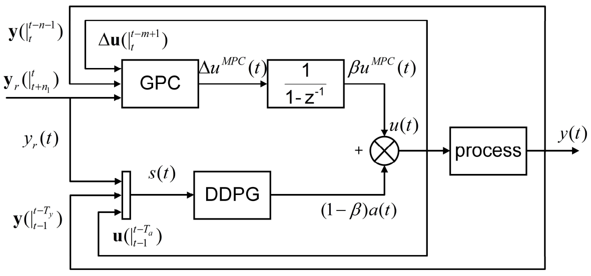
$\mathbf{y}(|_t^{t-n-1})$
$\Delta\mathbf{u}(|_t^{t-m+1})$
$\mathbf{y}_r(|_{t+n_1}^{t})$
GPC
$\Delta u^{MPC}(t)$
$\frac{1}{1-z^{-1}}$
$\beta u^{MPC}(t)$
$u(t)$
$y_r(t)$
+
process
$y(t)$
$s(t)$
DDPG
$(1-\beta)a(t)$
$\mathbf{y}(|_{t-1}^{t-T_y})$
$\mathbf{u}(|_{t-1}^{t-T_a})$

Fig. 3. Block diagram of the MP-RLC scheme.

The implementation of the MP-RLC scheme is divided into the training phase and testing phase. In the training phase, the main task is to optimize the policy network and value network by the DDPG algorithm, while in the test phase, the policy network, as the process controller, is evaluated.

The training phase consists of two parts. The first is the interaction part, and the second is the updating part. The interaction part mainly focuses on obtaining the training data through interaction with the process. During the interaction part, the GPC and the DDPG's policy network are implemented as controllers according to control law (29), and the process then gives the state transition and reward, which are then stored as tuples in replay buffer. In the updating part, experience in replay buffer are sampled to train the value network and policy network according to the optimization algorithm described in the last section. The pseudo-code of the algorithm at the training phase is summarized in Algorithm 2.

Algorithm 2. MP-RLC

```
1. Randomly initialize the value network $Q$ and policy
   network $\pi$
2. Initialize $Q'$ and $\pi'$ with $\theta^Q$ and $\theta^\pi$, respectively.
3. Set a replay buffer $R$
4. Set the initial values $\Delta\mathbf{u}(|_0^{-m+1})$ and $\mathbf{y}(|_0^{-n+1})$ for GPC
5. for the episode $k$ from 1 to $P$, do
Interaction part:
6.    Give the set-point $\mathbf{y}_r$ for the episode $k$
7.    Initialize a queue with the size of $T_{RS}+1$
8.    Initialize a Gaussian noise $\mathcal{N}(t)$ for exploration
9.    for the step $t$ from 1 to $T$, do
10.     Sample $s(t)$ from the environment
11.     Get action from the output of the policy network:
          $a(t)=\pi\left(s(t)\mid\theta^\pi\right)+\mathcal{N}(t)$
12.     Get $\Delta\mathbf{u}(|_t^{t-m+1})$ by the control law (24)
13.     Get the output of GPC $u^{MPC}(t)=u(t-1)+\Delta u^{MPC}(t)$
14.     Calculate the control $u(t)$ by control law (29)
15.     Execute $u(t)$ to the process, and observe $y(t)$
        and $r(t)$
16.     Generate extend state as follows:
          $s(t+1)=((y(t-T_y+1), y(t-T_y+2),\cdots,y(t)),$
             $y_r(t+1),(u(t-T_a+1),u(t-T_a+2),\ldots u(t)))$
17.     Push the element $(s(t),u(t),s(t+1))$ into the queue
18.     Take out the element in the head of the queue and
        insert $r(t)$ into the element to form a sample data
        $(s(t-T_{RS}),u(t-T_{RS}),r(t),s(t-T_{RS}+1))$, and then
        store it into the buffer $R$
Updating part:
19.     Randomly sample $N$ tuples from the buffer $R$
20.     Update the weights of the value network $Q$ by
        minimizing the loss function (4)
21.     Update the policy network $\pi$ according to the
        gradient calculated in (2)
22.     Update the two target networks by (6) and (7)
23.   end for
24. end for
```

Remark: Step 16 is used to extend the state with historical observations and actions, where the lengths T_y and T_a should be determined according to the time constant of the predictive model. Step 7, 17 and 18 are related to the "reward shifting" (RS) operation, in which the shifting step T_{RS} is determined according to the estimated time delay of the predictive model. When the time delay of the model is significant, the over-extended state can be avoided by using the reward shifting operation. When there is a large model mismatch or uncertainty, the Markov property can be guaranteed by selecting a relatively large T_y and T_a.

IV. Numerical Illustrations

In order to demonstrate the feasibility and effectiveness of the proposed algorithm, the MP-RLC scheme is implemented on two types of numerical systems in this section. One is a third-order linear system, and the other is a CSTR system, which is a typical nonlinear system commonly used in the chemical industry [29].

A. Case 1: Linear system

To verify the effectiveness and advantages of our approach, the proposed MP-RLC scheme, standalone GPC scheme and standalone DDPG algorithm are all implemented and tested on a discrete-time linear system with the 3rd-order dynamics and pure time delay, which is modeled by the following discrete time transfer function:

$$G(z)=\frac{2.651z^{-1}+5.298z^{-2}+0.5805z^{-3}}{1-1.454z^{-1}+0.5285z^{-2}-0.04736z^{-3}}z^{-6} \tag{30}$$

Considering the inevitable model mismatch in practical applications, it is assumed that the predictive model used for controller design is the following simplified first-order plus time-delay (FOPD) model obtained by a simple model identification algorithm which is omitted for clarity.

$$G_1(z)=\frac{32.19z^{-1}}{1-0.889z^{-1}}z^{-7} \tag{31}$$

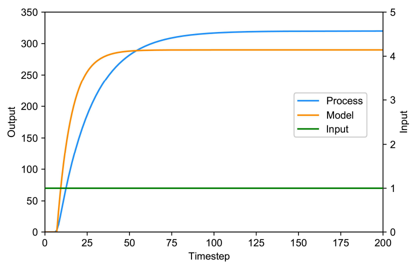

Fig. 4. Step responses of the process and the simplified model.

The step responses of the model (31) and the process (30) are shown in Fig. 4 which illustrates the model mismatch and time delay. The parameters used in the MP-RLC and DDPG are given in Table 1.

Table 1. Parameters and hyperparameters for MR-RLC algorithm

Notation	Description	Case 1	Case 2
P	Number of episodes	200	2000
T	Length of per episode	300	100
β	Combination factor in MP-RLC	0.8	0.5
n_1	Prediction horizon	30	10
n_2	Control horizon	30	10
λ	Weighting factor of GPC.	0.000001	0.173
ξ	Weighting factor of GPC.	1	1
N	Minibatch size	128	128
T_y	Output history step in the state	5	20
T_a	Action history step in the state	20	0
T_{RS}	Reward shift step	7	0
A	Action space	[-1,1]	[273,500]
ε	Tolerance in reward function	5	0.3
b	Scale factor in reward function	1	5
c	Scale factor in reward function	1	5
γ	Discount factor	0.99	0.99
τ	Update rate of target networks	0.001	0.001
-	Length of steps for testing	600	600
-	Learning rate of Critic	0.001	0.001
-	Learning rate of Actor	0.001	0.001
-	Buffer size	1000000	1000000
-	Initial standard deviation of noise	6	20
-	Decay factor of noise per step	0.99999	0.9995

To compare the efficiency and control performance of the three control schemes, the set-point tracking tests are conducted. Cumulative Rewards (CR) of one episode defined by (32) and the Sum of Absolute Errors (SAE) of one episode defined by (33) are used to evaluate the control performance of the three algorithms. CR matches the objective of RL and is suitable for evaluating the sample efficiency, while SAE is more comprehensive for the control performance.

$$CR=\sum_{t=1}^{T}r(t) \tag{32}$$

$$SAE=\sum_{t=1}^{T}\left|y(t)-y_r(t)\right| \tag{33}$$

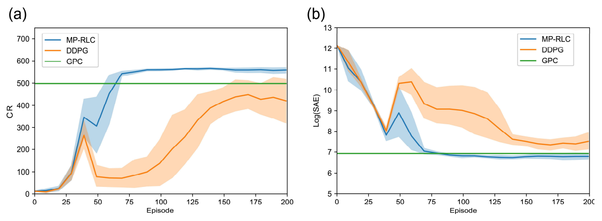

Fig. 5. Learning curves of MP-RLC, DDPG, and GPC evaluated by CR(a) and SAE(b) on the linear system.

Fig. 5 shows the CR and SAE curves of the three control algorithms, which are evaluated at an interval of 10 episodes during the training phase. And it lasted 200 episodes in total. The shaded areas represent 95% confidential intervals around the mean value based on the 7 experiments with different random seeds for training. It can be seen from the figures that both CR and SAE values of GPC are constants, indicating that the GPC does not have any learning performance during the training phase. Besides, to achieve the same CR or SAE value, MP-RLC required less episodes of training than the standalone DDPG, which illustrates that MP-RLC does have better sample efficiency than the DDPG algorithm. In addition, MP-RLC leads to a better control performance than GPC and DDPG after 200 episodes of training. The reason that MP-RLC out-performed GPC might be the former optimizes the control performance under a longer horizon than the later. Table 2 shows the mean values of CR and SAE for the three control algorithms evaluated after 200 episodes. It is shown that the CR value of MP-RLC is about 46% greater than that of DDPG, while the SAE value is about 62% lower than that of DDPG. Fig. 6 shows a comparison of the control performances of MP-RLC and DDPG after 200 episodes of training. From the output responses (Fig. 6(a)(c)) and the control input (Fig. 6(b)(d)), it can be seen that the control performance obtained by MP-RLC is much better than that of DDPG because of the smaller control error and more stability at the steady state.

Table 2. Mean value of metrics after training.

Metrics	MP-RLC	DDPG
CR	**574.713**	393.505
SAE	**765.503**	2060.577

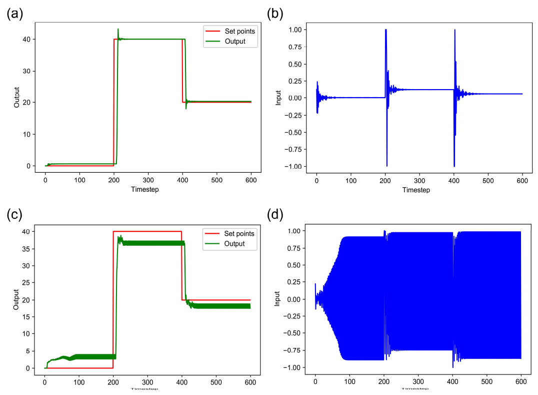
Fig. 6. Comparison of the control performances of MP-RLC and DDPG: (a) and (b) are, respectively, output response and control input of MP-RLC; (c) and (d) are, respectively, output response and control input of the DDPG.

B. Case 2: Continuous Stirred Tank Reactor (CSTR)

CSTR is a typical chemical process widely used in the modern chemical industry. To keep the safety, stability and efficiency of the reaction, the key variables of the process should be controlled effectively. For the complicated chemical reaction, the temperature control is especially challenging because of the complicated heat-releasing or heat-absorption dynamics. Thus the temperature control of the CSTR has become a benchmark to the control schemes.

There is an irreversible exothermic reaction from the reactant A to product B in the CSTR. According to the law of material balance and energy conservation, the continuous-time model of CSTR is as follows [29]:

$$\dot{C}_A = \frac{q}{v}(C_{Af} - C_A) - k_0 \exp(-\frac{E/R}{T_r})C_A \tag{34}$$

$$\dot{T}_r = \frac{q}{V}(T_f - T_r) + \frac{-\Delta H}{\rho C_p} k_0 \exp(\frac{-E/R}{T_r})C_A + \frac{UA}{V\rho C_p}(T_c - T_r) \tag{35}$$

where T_C is the only manipulated variable, referring to the temperature of the coolant. T_r and C_A are the controlled variable, representing the temperature of CSTR and concentration of the residual reactant A, respectively. C_{Af} is the feed concentration and T_f is the feed temperature, both of which are disturbing variables. The initial value of C_A is 0.5mol/L and T_f is 350K, which is a steady-state operation point. Other parameters we used in the simulation are given in Table 3 [29]. To demonstrate the proposed MP-RLC scheme has the ability to deal with time delay, it is assumed there is a pure input time delay in the process.

To demonstrate the applicability and effectiveness of proposed scheme, MP-RLC, standalone DDPG and standalone GPC are applied to the temperature control of a numerical CSTR system. The design of GPC is still based on a following simplified discrete-time model obtained by a simple model identification algorithm.

$$G_2(z) = \frac{0.1754}{z - 0.8154} z^{-3} \tag{36}$$

Table 3. Parameters of CSTR

Parameter	Description	Value
q	Volumetric flowrate	100L/min
C_{Af}	Feed concentration	1mol/L
T_f	Feed temperature	350K
V	Volume of CSTR	100L
ρ	Density of A-B mixture	1000g/L
E/R	Activation energy/Universal gas constant	8750K
ΔH	Heat of reaction	-1.2×10^4J/mol
k_0	Pre-exponential factor	7.2×10^{10}/min
C_p	Heat capacity	0.239J/(g·K)
UA	Heat transfer coefficient times area	5×10^4J/(min·K)
T_s	Sampling time	0.1s
T_d	Pure time delay	0.3s

In the testing phase, CR(with the same reward $r(t)$ as case 1) and SAE are used to evaluate three algorithms. Hyper-parameters and some parameters used in the MP-RLC and DDPG are given in Table 1.

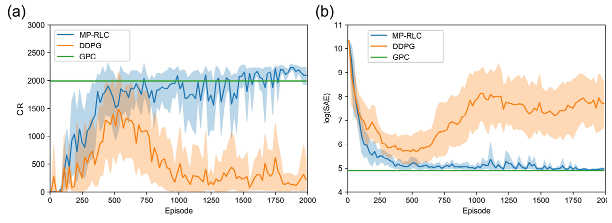
Fig. 7. Learning curves of MP-RLC, DDPG, and GPC evaluated by CR (a) and SAE (b) on the CSTR system.

Fig. 7 shows the CR and SAE curves of the three control algorithms, which are evaluated at an interval of 10 episodes during the training phase. MP-RLC can result in higher CR values and lower SAE than standalone DDPG, which shows that MP-RLC has better sample efficiency than standalone DDPG. Moreover, MP-RLC get slightly higher CR value than standalone GPC. But Fig. 7(b) shows that MP-RLC get almost same SAE value as GPC, which means the optimization ability of RL are somewhat limited by GPC. One might wonder why we need to use RL, the reason is that we expect RL can gurantee the continuous improvement of the control performance and finally leads to a better control control performance than GPC. Table 4 indicates MP-RLC outperforms DDPG at the end of training by about 13 times on CR and about 94% on SAE because of DDPG's collapse after the 500th episode. The simulation results demonstrate that the proposed MP-RLC scheme can be used for the process control of the nonlinear system with a significant time delay.

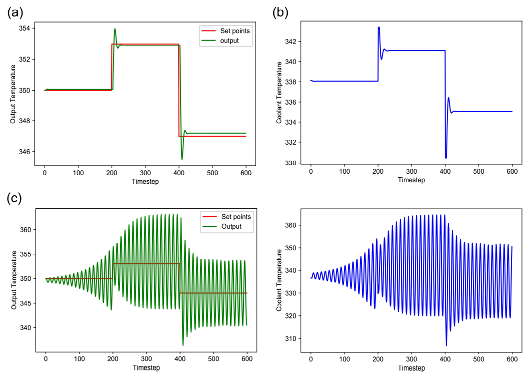

Fig. 8. Comparison of the control performances of MP-RLC and DDPG: (a) and (b) are, respectively, the output response and control input of MP-RLC; (c) and (d) are, respectively, the output response and control input of DDPG.

Table 4. Control performance of MP-RLC and DDPG

Metrics	MP-RLC	DDPG
CR	**2293.706**	155.384
SAE	**124.321**	2312.689

V. Conclusions

For the complicated process control problem, RL algorithm cannot find the optimal policy quickly due to the low sample efficiency. To solve this problem, an MPC guided RL scheme is proposed in this paper. The proposed scheme executes actions by directly combining the outputs of MPC with the actions of the DDPG in the training phase, which results in the more effective training data. Two typical simulation cases are conducted on a third-order linear system and the CSTR nonlinear system both with time delay. The simulation results show that our proposed scheme possesses higher sample efficiency than RL scheme without guidance and can achieve better control performance. The future work can focus on that how to free RL from the limitation of guiding algorithm along with iterations. This paper provides a new method to apply RL technique to the process control.

References

[1] J. Hoskins and D. Himmelblau, "Process control via artificial neural networks and reinforcement learning," *Computers & chemical engineering,* vol. 16, no. 4, pp. 241-251, 1992.

[2] H. Benbrahim and J. A. Franklin, "Biped dynamic walking using reinforcement learning," *Robotics and Autonomous Systems,* vol. 22, no. 3-4, pp. 283-302, 1997.

[3] D. Ernst, M. Glavic, and L. Wehenkel, "Power systems stability control: reinforcement learning framework," *IEEE transactions on power systems,* vol. 19, no. 1, pp. 427-435, 2004.

[4] R. S. Sutton, A. G. Barto, and R. J. Williams, "Reinforcement learning is direct adaptive optimal control," *IEEE Control Systems Magazine,* vol. 12, no. 2, pp. 19-22, 1992.

[5] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature,* vol. 518, no. 7540, p. 529, 2015.

[6] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature,* vol. 550, no. 7676, pp. 354-359, 2017.

[7] J. M. Lee and J. H. Lee, "Neuro-dynamic programming method for mpc," *IFAC Proceedings Volumes,* vol. 34, no. 25, pp. 143-148, 2001.

[8] J. M. Lee and J. H. Lee, "Approximate dynamic programming-based approaches for input–output data-driven control of nonlinear processes," *Automatica,* vol. 41, no. 7, pp. 1281-1288, 2005.

[9] J. M. Lee, N. S. Kaisare, and J. H. Lee, "Choice of approximator and design of penalty function for an approximate dynamic programming based control approach," *Journal of process control,* vol. 16, no. 2, pp. 135-156, 2006.

[10] 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.

[11] S. Spielberg, A. Tulsyan, N. P. Lawrence, P. D. Loewen, and R. B. Gopaluni, "Towards Self-Driving Processes: A Deep Reinforcement Learning Approach to Control," *AIChE Journal,* 2019.

[12] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015.

[13] Y. Ma, W. Zhu, M. G. Benton, and J. Romagnoli, "Continuous control of a polymerization system with deep reinforcement learning," *Journal of Process Control,* vol. 75, pp. 40-47, 2019.

[14] P. Petsagkourakis, I. O. Sandoval, E. Bradford, D. Zhang, and E. A. del Rio-Chanona, "Reinforcement learning for batch bioprocess optimization," *Computers & Chemical Engineering,* vol. 133, p. 106649, 2020.

[15] T. A. Badgwell, J. H. Lee, and K.-H. Liu, "Reinforcement learning–overview of recent progress and implications for process control," in *Computer Aided Chemical Engineering*, vol. 44: Elsevier, 2018, pp. 71-85.

[16] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1-9.

[17] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465-472.

[18] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *2016 IEEE international conference on robotics and automation (ICRA)*, 2016, pp. 528-535: IEEE.

[19] S. Kamthe and M. P. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," *arXiv preprint arXiv:1706.06491,* 2017.

[20] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," 2014.

[21] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008-1014.

[22] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature,* vol. 518, p. 529, 02/25/online 2015.

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

[24] D. W. Clarke, C. Mohtadi, and P. Tuffs, "Generalized predictive control—Part I. The basic algorithm," *Automatica,* vol. 23, no. 2, pp. 137-148, 1987.

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

[26] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 AAAI Fall Symposium Series*, 2015.

[27] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807-814.

[28] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440-1448.

[29] J. D. Hedengren, "A nonlinear model library for dynamics and control," *Yeast,* vol. 7, p. 24, 2008.