2020 IEEE 9th Data Driven Control and Learning Systems Conference
November 20-22, 2020, Liuzhou, China

Iterative Learning Control (ILC) Guided Reinforcement Learning Control (RLC) Scheme for Batch Processes

Xinghai Xu[1], Huimin Xie[2], Jia Shi*
Department of Chemical & Biochemical Engineering, School of Chemistry & Chemical Engineering
Xiamen University Xiamen, Fujian 361005, P. R. China

Abstract: Iterative learning control (ILC) is a kind of effective learning control scheme which is mainly designed to solve the problems in controlling a batch or repetitive process. Although the control performances of ILC systems can improved from batch to batch, it still strongly depends on the repeatability of the process and control target. Reinforcement learning (RL) is another learning based optimization algorithm which can be applied to many complicated decision-making scenarios. Data-driven based RL algorithms have good robustness due to the generalization of the policy neural network, however, it is low-data efficiency in network training. In this paper, for batch process control we propose a new reinforcement learning control (RLC) scheme which is guided by classical iterative leaning control. On the one hand, this RLC scheme has capability to optimize the policy network faster than RL algorithm without guidance, on the other hand, the generalization of deep policy network improves the robustness of the control system. Based on the numerical simulations, the effectiveness of the proposed control scheme is demonstrated by comparing with the conventional reinforcement learning algorithm and the P-type iterative learning control scheme. This paper provides a new way for the application of reinforcement learning algorithm to batch process control.

Key Words: Iterative learning control (ILC), Deep reinforcement learning, Batch/repetitive processes

1 Introduction

Batch process are widespread in modern manufacturing. Almost half of petroleum processing and most of fine chemical production can be categorized as a batch process. Iterative learning control (ILC) is an effective control scheme for batch, repetitive or cyclical processes[1]. The basic idea of the ILC scheme is to improve the control performances by using the repeatability of the process to update the control behavior from batch to batch. Since 1984, Arimoto[2] first proposed this control scheme and successfully applied it to the manipulation of robotic arm, the research work on ILC system has attracted wide attention. As the control of the ILC systems updated from cycle to cycle, the control performances strongly depend on the repetitive natures of the process, including the repeatability of initial conditions, dynamic characteristics and control target of the processes. When these repetitive natures cannot be guaranteed, the control performances of the ILC systems will be deteriorated. Therefore, how to ensure the performances of ILC systems under the non-repetitive conditions has become the key problems in the current research for ILC.

With the fast development of Artificial Intelligent (AI), reinforcement learning (RL) plays an important role in intelligent decision-making. RL is not only showing the outstanding performance in electronic games[3], but also widely used in recommendation systems[4]. Since Alpha Go defeat master Li in Go programs, RL come into public view and appears lots of researchers. With computer powering being enhanced, RL combined with deep neural networks can do more and more complex work than human-being. RL algorithm derives from dynamic programming which is an optimal control algorithm. Because of this, researchers try to implement it into industrial processes, hoping it can deal with nonlinear control problem effectively. Pioneers like Hoskins[5] successfully applies RL to controlling CSTR system. Besides, F.L. Lewis[6]. demonstrates that RL can learn to optimize control policies by exploring environment episode by episode in multi-variability system.

RL acquires large number of data through the interaction with environment, and then optimizes its control policies episode by episode till it results in the best control policy. Aiming to apply RL method to industrial processes, researchers have done some interesting works. Spielberg[7] et al. has illustrated that RL can control HAVC[8](Heating, Ventilating and Air Conditioning) system excellently. Yan Ma[9] implemented RL to controlling polymerization system. However, there still exit some challenges to implement RL into industrial process. RL is totally a model-free and data-driven based method which needs quantity of data form the interaction with the environments, and it also needs trial and error, i.e. exploration, to search for a better policy. However, most industrial processes can't afford any random exploration which might cause equipment damage or economic loss. To solve this problem, an algorithm called guided policy search (GPS) proposed by Sergey Levine[10] [11] et al. which can effectively enhance the efficiency of searching the best policy. GPS is also capable to shorten the training time of RL. However, the algorithm of GPS is complex, and it will impose large computational burden for process control.

In order to enhance the ability of ILC scheme to deal with the non-repetitive natures of the batch processes, we combine a RL algorithm with a simple ILC scheme, forming an iterative learning control guided reinforcement learning control (ILC-RLC) scheme. Compared with ILC algorithm and the RL algorithm without guidance, the proposed control scheme has two advantages. First, benefited from the generalization of the policy obtained by reinforcement learning, the closed-loop control system has better robustness than the classical ILC scheme, especially when the control

*This work is supported by National Natural Science Foundation (NSFC) of China under Grant 00000000.

978-1-7281-5922-5/20/$31.00 ©2020 IEEE

target varying from batch to batch. Second, under the guidance of ILC, RL algorithm can obtain more effective information to improve the efficiency of searching optimal policy. Although the systematic analysis for the proposed control scheme is still a hard work, based on the numerical simulations, we demonstrated the superior learning performance and robustness of the control system.

The rest of the paper are organized as follows: in Section 2, we give the preliminaries of deep deterministic policy gradient (DDPG), which is a deep RL algorithm widely used in deterministic continuous-time processes, and P-type ILC scheme for batch processes, and then we present the ILC guided RLC (ILC-RLC) algorithm in Section 3. In Section 4, we validate our algorithm by implementing it in numerical batch processes. Finally, we make the conclusions about our work.

2 Preliminaries

In order to introduce the iterative learning control guided reinforcement learning control algorithm in the next section, in this section we need to review the ILC scheme and the RL algorithm that to be used in our algorithm.

2.1 ILC scheme for batch process

For simplicity, it is assumed in this paper that a batch process is a single-input-single-output (SISO) system described by the following discrete-time model:

$$\begin{cases} x_k(t+1) = f(x_k(t), u_k(t), t) \\ y_k(t) = g(x_k(t), u_k(t), t) \end{cases} \quad t = 0,1,...,T; \quad k = 1,2,... \tag{1}$$

where t and k represent, respectively, the discrete-time and batch index, T is the finite time duration of each batch, $u_k(t)$, $y_k(t)$ and $x_k(t)$ are, respectively, the input, output and state of the process at time t of the kth batch, $f(\bullet,\bullet,\bullet)$ and $g(\bullet,\bullet,\bullet)$ represent, respectively, the state and output function determining the dynamic of the process within the batch. In order to implement an ILC scheme to batch process, it is necessary to assume that the initial state of each batch is the same, i.e. $x_k(0) \equiv x_0$, the time-wise dynamics of the processes are consistent from batch to batch, and the control target for each batch, i.e. the reference trajectories, $y_r(t), t = 1,2,\cdots,T$, are exactly the same.

At any time t of the kth batch, we define the control error as:

$$e_k(t) = y_r(t) - y_k(t) \tag{2}$$

Then the idea of ILC is to determine the control $u_k(t)$ according to the control $u_{k-1}(t)$ and all available information at time t of the kth batch, so as to obtain the control performance improved from batch to batch. Generally, the control law of ILC can be formulated as follows:

$$u_{k+1}(t) = u_k(t) + U(\boldsymbol{u}_k, \boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{y}_r) \tag{3}$$

where $\boldsymbol{u}_k, \boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{y}_r$ represent, respectively, the input, state, output and reference sequence available at time t of the kth batch, and $U(\boldsymbol{u}_k, \boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{y}_r)$ represents the learning function to be designed.

Among all of the ILC schemes[12], the following linear P-type ILC law is the simplest:

$$u_{k+1}(t) = u_k(t) + Le_k(t) \tag{4}$$

where L is a constant termed as *learning rate*.

The main advantages of the P-type iterative learning law are that the design of learning rate is very easy and the learning convergence of the control system can be guaranteed. For the ILC-RLC scheme to be developed in this paper, the above P-type ILC law will be adopted.

2.2 Reinforcement learning (RL) algorithm

Reinforcement learning (RL) is a data-driven based self-optimization learning algorithm which usually applied to Markov decision processes (MDP).

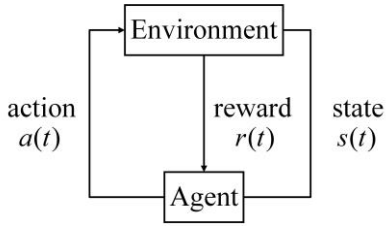

Fig. 1: The framework of reinforcement learning

The framework of RL is shown in Fig. 1. There are two major elements in a RL system, one is environment, another is agent. Agent and environment interact online through three type of information, including the state $s(t)$, the action $a(t)$ and the reward $r(t)$. The state $s(t)$ is the feedback from environment to agent, indicating the dynamic response of the environment, the action $a(t)$ is determined by control policy of the agent, and the reward $r(t)$ is usually defined by environment based on the action and the corresponding state response. Compared with a closed-loop control system, environment corresponds to the plant of the control system, agent corresponds to the controller of the system, action is essentially same as the control input of the environment determined by the policy network of the agent according to the state, state is the online information observed by agent, and the reward is usually regarded as the credits given by a supervisor of the system used to help the agent to find the good action. Except for the instant reward, we also need a value function $Q(s(t), a(t))$ to estimate the accumulated reward that can be obtained in the future based the current policy. The main objective of RL algorithm is to optimize the policy network through an appropriate learning method such that the value function at every time instant has a maximized value. According to the optimization theory, the optimal value function should satisfy the following Bellman equation:

$$\begin{aligned} &Q^{\mu}\left(s(t), a(t)\right) \\ &= \mathbf{E}\left[r\left(s(t), a(t)\right) + \gamma Q^{\mu}\left(s(t+1), \mu\left(s(t+1)\right)\right)\right] \end{aligned} \tag{5}$$

where $s(t)$ and $a(t)$ represent, respectively, the state and the action at time t, $r\left(s(t), a(t)\right)$ is the reward function, $\mu\left(s(t)\right)$ is the policy to be optimized, $Q^{\mu}\left(s(t), a(t)\right)$ indicates the value function, E is the expectation.

In this paper, we take deep deterministic policy gradient (DDPG)[13], a kind of deep RL algorithm, as the RL algorithm that will combined with ILC scheme in the next section. DDPG derives from deterministic policy gradient (DPG) [14],

which is widely used for solving the decision-making problem of system with continuous-time action space and deterministic dynamic.

2.3 Deep deterministic policy gradient (DDPG) algorithm

2.3.1 state

RL use the state of the environment as input of the policy and value function[15]. In order to ensure that the agent gets sufficient information for decision-making, the state are usually composed of the control target $y_r(t)$ and the state $x(t+1)$, which is defined as follows:

$$s(t)=(y_r(t),x(t+1)) \tag{6}$$

For the DDPG algorithm, the action of the agent is determined by the policy network according to the state of the environment, which can be described as follows:

$$a(t)=\mu(s(t)|\theta^{\mu}) \tag{7}$$

where θ^{μ} represents the weights of the neural network.

Except the state information, agent also need a reward to take an action. Reward function is very important to RL. In this paper we designed a reward function using the truncated quadratic function of the control error, which is defined as follows:

$$r\left(s(t),a(t)\right)=\begin{cases}-\dfrac{e(t)^2}{\varepsilon^2}+1, & |e(t)|\le\varepsilon\\ 0, & |e(t)|>\varepsilon\end{cases} \tag{8}$$

where $e(t)=y_r(t)-y(t)$ indicates the control error. When $|e(t)|\le\varepsilon$, the profit of reward function is shown in Fig. 2.

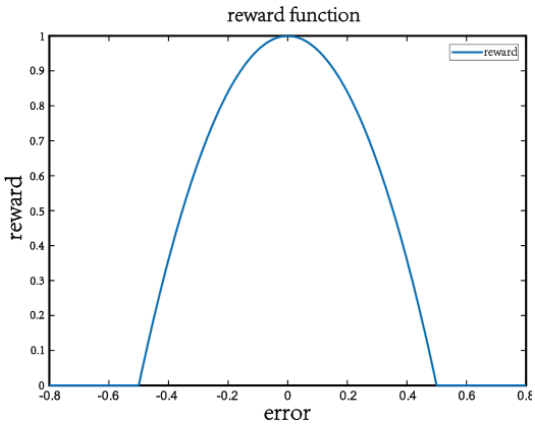

Fig. 2: Reward Function.

DDPG use a framework of learning named 'actor-critic'[13], and it implement deterministic policy of DPG[14] into actor section. In actor-critic framework, actor determines the action $a(t)$ based on the policy network expressed by (7). Critic is aimed to judge whether the policy network θ^{μ} is good enough. For the policy network represented by a neural network, DDPG use the back propagation algorithm to train the policy network based on the gradient of value function $Q(s(t),a(t)|\theta^{Q})$ as follows[13]:

$$\begin{aligned}&\nabla_{\theta^{\mu}}J\\ &\approx \mathbf{E}_{s(t)\sim\rho^{\beta}}\left[\nabla_a Q(s(t),a(t)|\theta^{Q})|_{s=s(t),a=\mu(s(t)|\theta^{\mu})}\right]\\ &=\mathbf{E}_{s(t)\sim\rho^{\beta}}\left[\nabla_a Q(s(t),a(t)|\theta^{Q})|_{s=s(t),a=\mu(s(t)|\theta^{\mu})}\nabla_{\theta^{\mu}}\mu\left(s(t)|\theta^{\mu}\right)|_{s=s(t)}\right]\end{aligned} \tag{9}$$

where θ^{Q} represents the weights of the value network, ∇ represent the gradient. For the critic, it take a method called 'TD error' to update the value network θ^{Q} with the loss function defined by[13]:

$$\begin{aligned}&L(\theta^{Q})\\ &=\mathbf{E}_{s(t)\sim\rho^{\beta},a(t)\sim\beta,r(t)\sim E}[(Q(s(t),a(t)|\theta^{Q})-r(s(t),a(t))\\ &\qquad -\gamma Q'(s(t+1),a(t+1)|\theta^{Q'}))^2]\end{aligned} \tag{10}$$

At last it will use the followed function to update parameters of two target networks[13]:

$$\begin{cases}\theta^{Q'}\leftarrow\tau\theta^{Q}+(1-\tau)\theta^{Q'}\\ \theta^{\mu'}\leftarrow\tau\theta^{\mu}+(1-\tau)\theta^{\mu'}\end{cases} \tag{11}$$

where $0<\tau<1$ is the update rate.

The pseudo-code for the DDPG algorithm is summarized as follows, more details can be found in reference[13].

Algorithm 1: **DDPG algorithm**

1 Randomly initialize the weights of critic network θ^{Q} and policy network θ^{μ}, respectively
2 Set the weights of the target network Q' and μ' same as Q and μ, respectively
3 Reset a replay buffer R
4 **for** episode index k from 1 to K, do:
5 Get the initial state $s(0)$ from environment
6 **for** step index t from 1 to T, do:
7 Generate a Gaussian noise $\mathcal{N}_t$ for exploration
8 Choose action: $a(t)=\mu\left(s(t)|\theta^{\mu}\right)+\mathcal{N}_t$
9 Output the action to environment to get the state $s(t+1)$ and reward $r\left(s(t),a(t)\right)$
10 Store $\langle s(t),a(t),r(s,a),s(t+1)\rangle$ in R
11 Sample a stochastic batch data from R
12 Update critic network Q according to the loss function (10)
13 Update policy network μ based on the gradient of value function (9)
14 Update target networks Q' and μ' by law (11)
15 **end for**
16 **end for**

3 Iterative learning control guided reinforcement learning control (ILC-RLC) scheme

For the batch process, the following control law can be obtained by directly combining the control of the ILC scheme with the action of the policy network in RL:

$$u_k(t)=\beta u_k^{ILC}(t)+(1-\beta)a_k^{RL}(t) \tag{12}$$

where $a_k^{RL}(t)$ is the action determined by policy network of the agent, and $u_k^{ILC}(t)$ is the control output calculated by ILC law, β is the weight factor.

Obviously, the above control law A simply combines the output of the ILC with the actions of the RL. Although it may result in the generalization of the policy network working in the control loop and improve the robustness of the control system, it cannot guide the DDPG algorithm by the iterative learning control strategy. For complex systems, if the learning efficiency of the DDPG algorithm is low, the DDPG control actions may not ensure good control performance, and its exploration may also disturb the control performance of iterative learning control.

In order to guide the policy optimization of the RL algorithm, we need to submit the control action $a_k^{RL}(t)$ to the ILC algorithm to ensure the control action is updated batch to batch according to the control performance, at the same time the final control will be used by DDPG algorithm to update policy and value network. That is, in the ILC-RLC algorithm the P-type ILC law (4) need to be rewritten as:

$$u_{k+1}^{ILC}(t) = a_k^{RL}(t) + Le_k(t) \tag{13}$$

The block diagram of the proposed ILC-RLC scheme is shown in Fig.3. It should be noted that the guiding method used in ILC-RLC scheme is more easier to implement than the way in GPS[11] and Path-integral method[16].

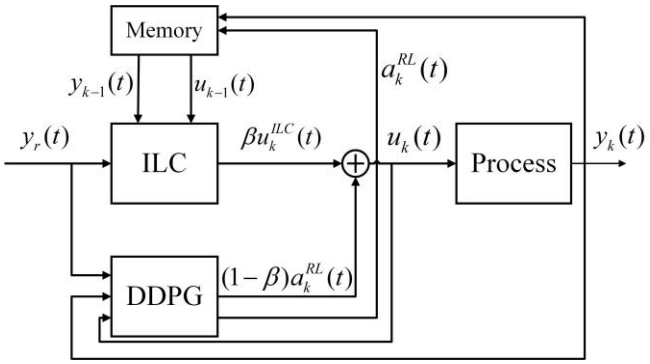

Fig. 3: Block diagram of the ILC-RLC scheme

In summary, the ILC-RLC scheme can be described by the following algorithm:

Algorithm 2: ILC-RLC algorithm

Initialize the control output $u_0^{ILC}(t), t = 0,1,2,\cdots,T$ for P-type ILC law (4)
Random initialize policy network $\mu(s \mid \theta^{\mu})$ and value network $Q(s,a \mid \theta^{Q})$ for DDPG
Initialize target network μ' and Q' by using the same weights of μ and Q, respectively
Set a replay buffer R for DDPG
Set a memory buffer M for ILC
Set the control target of the batch: $y_r(t), t = 0,1,2\cdots,T$
Initialize the action $a_0^{RL}(t)$ and the state $x_0(t)$
Generate the initial state information $s_0(t)$ according to (6)
for batch index k from 0 to K, do:
  for step index t form 0 to T, do:
Interaction for DDPG:
    Sample the state information $s_k(t)$ from the environment
    Generate a Gaussian noise $\mathcal{N}_k(t)$ for exploration
    Get the action from the output of the policy network: $a_k^{RL}(t) = \mu\left(s_k(t) \mid \theta^{\mu}\right) + \mathcal{N}_k(t)$
    Calculate the output of ILC $u_k^{ILC}(t)$ according to control law (4)
    Calculate the guided control $u_k(t)$ according to control law (13)
    Output the $u_k(t)$ to process to get state response, $x_k(t+1)$, output $y_k(t)$ and reward $r_k(t)$.
    Store $\langle s_k(t), a_k(t), s_k(t+1), r_k(t)\rangle$ into R
    Store $\langle a_k^{RL}(t), y_k(t)\rangle$ into M
Network Update for DDPG:
    Random select a mini-batch data from R
    Update the weights of value network θ^{Q} by minimizing loss function (10)
    Update the policy network θ^{μ} based on the gradient of value function (9)
    Update the weights of two target networks $\theta^{Q'}$ and $\theta^{\mu'}$ by using update laws (11)
  end for
Calculate $u_{k+1}^{ILC}(t)$ for next batch :
  Replace the ILC output $u_k^{ILC}(t)$ with the action $a_k^{RL}(t)$
  Calculate ILC control output $u_{k+1}^{ILC}(t)$ for the next batch by equation (4) used to guide the control action of DDPG
end for

4 Numerical simulations

To illustrate that our algorithm can search optimal policy more efficient than DDPG without guidance and has the better robustness than ILC, we implemented the proposed ILC-RLC scheme into two type of batch processes: a first-order process and a second-order process. We demonstrate that our algorithm performs very well in the two typical processes.

Case 1: the first-order process

In order to verify the feasibility of the control algorithm, we implement the ILC-RLC scheme into a first-order system defined by the following discrete-time transfer function:

$$G(z) = \frac{0.4877z^{-1}}{1-0.9512z^{-1}} \tag{14}$$

To provide the state information to DDPG algorithm, we also construct the corresponding state space model as follows:

$$\begin{cases} x_k(t+1) = 0.9512x_k(t) + 1 \cdot u_k(t) \\ y_k(t) = 0.4877x_k(t) + 0 \cdot u_k(t) \end{cases} \tag{15}$$

where $x_k(t)$ is the state observed by agent, $y_k(t)$ and $u_k(t)$ are, respectively, the output and the control input of the process.

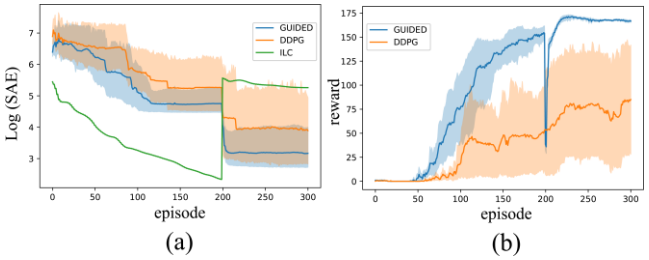

Fig. 4: Control and learning performances of P-type ILC, DDPG and ILC-RLC: (a) the log value of cumulated control error of P-type ILC, DDPG and ILC-RLC. (b) the cumulated reward of DDPG and ILC-RLC.

We compared the learning performances of the P-type ILC, the DDPG and the proposed ILC-RLC scheme from the convergence of the Sum of Absolute Control Error (SAE) of the batch, as shown in Fig. 4(a). It demonstrates that the ILC scheme results the fastest convergence, followed by ILC-RLC algorithm and DDPG. In order to verify the robustness of ILC-RLC for the non-repetitive variation of the batch process, we changed the control targets at the 201th batch. As shown in Fig. 4a, the control performance of ILC deteriorates significantly, but ILC-RLC scheme keeps the convergence. Compare the time-wise accumulated rewards obtained by DDPG and ILC-RLC during the learning procedure, as shown in figure 4b, ILC-RLC algorithm shows the ability to get rewards faster, indicating that the learning efficiency of ILC-RLC is significantly improved under the guidance of ILC.

Fig. 5 shows the corresponding output responses and control input signals of ILC, DDPG and ILC-RLC algorithms for the 201th batch where the control target changed significantly. It shows that ILC is unable to ensure the tracking performance, while DDPG and ILC-RLC can ensure the good tracking performance.

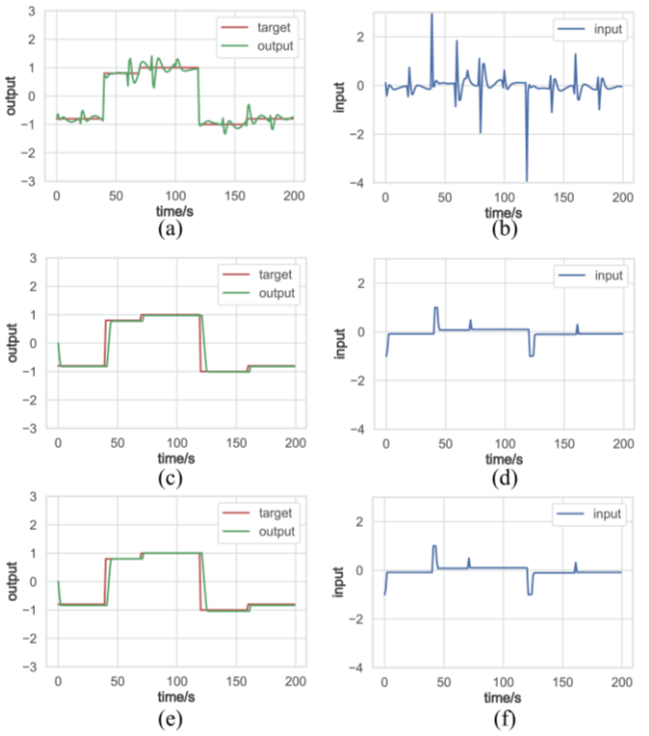

Fig. 5: Output responses and control input of ILC, ILC-RLC and DDPG schemes for the 201th batch: (a) Output response of ILC; (b) Control input of ILC; (c) Output response of ILC-RLC; (d) Control input of ILC-RLC; (e) Output response of DDPG without guidance; (f) Control input of DDPG without guidance.

It can be concluded from the above simulation results that the learning efficiency of RL is significantly improved with the guidance of ILC. On the other hand, combining RL control with traditional ILC can improving the robustness of the control system to against non-repetitive variation.

Case 2: the second-order process

In order to verify the effectiveness of the control scheme further, we tested the control scheme on the following system with a second-order dynamic.

$$G(z)=\frac{2.642z^{-1}+1.353z^{-2}}{1-0.7358z^{-1}+0.1353z^{-2}} \tag{16}$$

We also formulate it into state space model as follows:

$$\begin{cases} x(t+1)=Ax(t)+Bu(t) \\ y(t)=Cx(t)+Du(t) \end{cases} \tag{17}$$

where

$$A=\begin{bmatrix} 0.7358 & -1.1353 \\ 1 & 0 \end{bmatrix},\quad B=\begin{bmatrix} 1 & 0 \end{bmatrix}$$
$$C=\begin{bmatrix} 2.6424 & 1.3534 \end{bmatrix}, D=0 \tag{18}$$

We also compared the convergence of the SAE under the ILC, DDPG and ILC_RLC scheme, as shown in Fig. 6(a), and the time-wise accumulated rewards obtained by DDPG and ILC-RLC in the learning process, as shown in Fig. 6(b). We can draw the same conclusions as that in Case 1.

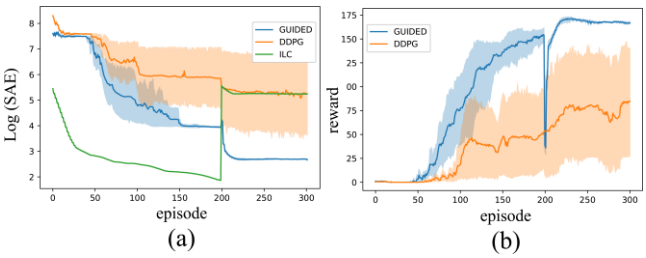

Fig. 6: Control and learning performances of P-type ILC, DDPG and ILC-RLC: (a) the log value of cumulated control error of P-type ILC, DDPG and ILC-RLC. (b) the cumulated reward of DDPG and ILC-RLC.

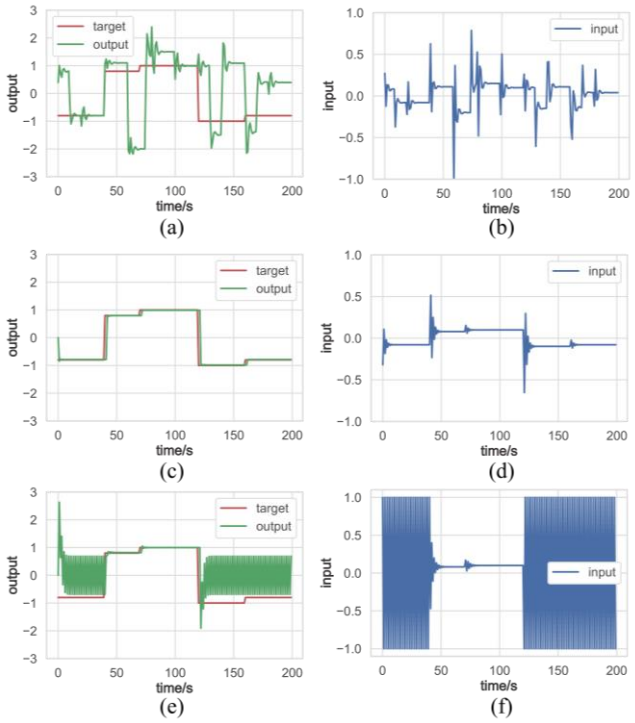

Fig. 7: Output responses and control input of ILC, ILC-RLC and DDPG schemes for the 201th batch: (a) Output response of ILC; (b) Control input of ILC; (c) Output response of ILC-RLC; (d) Control input of ILC-RLC; (e) Output response of DDPG without guidance; (f) Control input of DDPG without guidance.

Figure 7 shows the system responses and control signals of the 201th batch where the control target changed significantly for the three kinds of control scheme. It can be seen from these figures that ILC system cannot guarantee the tracking performance when there is non-repetitive variation, the DDPG scheme without guidance cannot guarantee the good control performance, only ILC-RLC scheme keeps a good control performance.

5 Conclusions

For ILC system can't handle the disturbance caused by non-repeatability of the batch process, we propose new method to combine the reinforcement learning with iterative learning control scheme in this paper, forming the ILC guided RLC scheme for the batch processes. The simulation results on the first-order and the second-order systems demonstrate that the proposed control scheme can guarantee a good control performance when the control target dramatically changed. This is mainly benefit from the generalization of policy network optimized by RL algorithm. In addition, under the guidance of ILC, the learning efficiency of reinforcement learning is also significantly improved. Simulation results show that the proposed control scheme can effectively improve the robustness of the iterative learning control system.

References

[[1] J. R. Mike Barker, Steve Mackay, *1 - Introduction*, **2005**.

[2] S. K. Suguru Arimoto, and Fumio Miyazak, *Journal of Robotic Systems* **1984**, *1*, 123-140.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, *Nature* **2017**, *550*, 354-359.

[4] S. L. Xinshi Chen, Hui Li, Shaohua Jiang, Yuan Qi, Le Song, *arXiv preprint* **2018**, *arXiv:1812.10613*.

[5] D. M. H. J. C. Hoskins, *Computers chem.Engng* **1992**, *16*, 241-251.

[6] F. L. Lewis, D. Vrabie, *IEEE Circuits and Systems Magazine* **2009**, *9*, 32-50.

[7] S. Spielberg, A. Tulsyan, N. P. Lawrence, P. D. Loewen, R. Bhushan Gopaluni, *AIChE Journal* **2019**, *65*.

[8] E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, M. Gibescu, J. G. Slootweg, *IEEE Transactions on Smart Grid* **2019**, *10*, 3698-3708.

[9] Y. Ma, W. Zhu, M. G. Benton, J. Romagnoli, *Journal of Process Control* **2019**, *75*, 40-47.

[10] S. K. Levine, V, *International Conference on Machine Learning* **2013**, 1.

[11] S. W. Levine, Nolan, Abbeel, Pieter, *IEEE Robotics and Automation Society* **2015**.

[12] H.-S. Ahn, Y. Chen, K. L. Moore, *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* **2007**, *37*, 1099-1121.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, *Computer Science* **2015**, *8*, A187.

[14] G. L. David Silver, Nicolas Heess, Thomas Degris, Daan Wierstra, Martin Riedmiller, **2014**.

[15] A. G. B. Richard S. Sutton, *Reinforcement Learning: An Introduction*, MIT, **2018**.

[16] M. K. Yevgen Chebotar, Ali Yahya, Adrian Li, Stefan Schaal, Sergey Levine, *IEEE Robotics and Automation Society* **2016**.