2021 IEEE 10th Data Driven Control and Learning Systems Conference
May 14-16, 2021, Suzhou, China

Design of Nonlinear Iterative Learning Control Based on Deep Reinforcement Learning Algorithm

Jia Shi[1], Kechao Wen[1], Xinghai Xu[1], Xiongzhe Hu[1], Dijun Luo[2]

1. Department of Chemical and Biochemical Engineering, School of Chemistry and Chemical Engineering
Xiamen University, Xiamen, Fujian, China 361005
E-mail: jshi@xmu.edu.cn
2. Tencent AI Lab, Shenzhen, Guangdong, China
E-mail: dijunluo@tencent.com

Abstract: Iterative learning control (ILC) is an advanced control method which has been studied and widely used in periodical, repetitive or batch processes. However, there is still a lack of an effective method for the design of nonlinear iterative learning control (NILC). In view of the excellent performance of deep reinforcement learning (DRL) in dealing with the decision-making problems for the complex dynamical processes, in this paper, we propose an intelligent design method for NILC system by using deep deterministic policy gradient (DDPG), a typical DRL algorithm. By properly designing the state information and the instant reward, the design algorithm can gradually realize the optimal NILC law through the interaction without any requirement of prior knowledge of the processes. The numerical simulation based on a nonlinear model illustrates the effectiveness and applicability of the proposed design method.

Key Words: iterative learning control, reinforcement learning, deep neural network, policy gradient

1 Introduction

Iterative learning control (ILC) is an advanced control method for manufacturing or operating processes with repetitive nature. This method can make full use of the repeatability of the processes to achieve the high quality control performance. ILC was firstly proposed for the motion control of the manipulator[1]. Due to less reliance on prior knowledge of the process, the design and implementation of the ILC scheme is relatively simple, and at the same time, it can achieve good tracking performance, so it has attracted extensive attention of the researchers and engineers in the field of process control[2]. In recent years, the in-depth research on ILC system has become more and more important with the wide applications of periodic, repetitive or batch manufacturing in modern industry, such as integrated circuit manufacturing[3], fine chemical engineering[4,5], robot control[6], bio-pharmaceutical engineering[7] and industrial extruder plant[8] etc..

The original motivation of ILC is to improve the control performance of the system by iteratively updating the control signals along the repetitive index of the process. Therefore, no matter what control scheme is, the control law of ILC can generally be described by the following formula[9]:

$$u_k(t) = u_{k-1}(t) + L\left(I_k(t)\right) \tag{1}$$

where, k indicates the cycle (or repetitive, batch) index, t is the time index within the cycles, $I_k(t)$ represents any available information at the time t of the k th cycle, $L(\bullet)$ represents an updating function to be designed. The above formula shows that the control actions over each cycle are determined by the control actions of the previous cycle and the updating function $L(\bullet)$ iteratively, so $L(\bullet)$ is called the *iterative updating law* (IUL). According to the control tasks, a lot of design and analysis methods for the ILC systems have been developed, such as the P-type and PD-type ILC[10], robust ILC [11], which is developed to improve the robustness of the system, as well as two-dimensional (2D) ILC[12] that contains both feedforward ILC law and real-time feedback control law, etc.. All of these control schemes essentially can be described by formula (1). Theoretically, for a complex processes with nonlinearity, the optimal IUL should be a nonlinear function. However, in order to facilitate the design and analysis, almost all of the existing research works assumed that IUL is a linear function of the process information[9]. How to design a nonlinear IUL is still a key problem in both theoretical research and practical application.

Reinforcement learning (RL), which was first proposed in the 1970s[13], is an optimal decision-making algorithm for the dynamic processes based on the optimization principle. In recent years, with the rapid improvement of computer hardware, deep reinforcement learning (DRL), which combines the RL alogrithm with the deep neural network (DNN) and machine learning (ML), has been developed and become a powerful tool to solve decision-making and optimization problems of the complex dynamic processes. It not only has shown the superior intelligence in chess and computer games, but also has been widely applied to intelligent transportation[14], business recommendation[15] and quantitative trading system[16] etc.. The prominent advantage of DRL is that it can realize optimal decision-making through the interaction between the agent and the environment without any requirments of prior knowledge of the processes. In addition, DRL based on the DNN approximator essentially leads to an optimal nonlinear decision-making system. Therefore, it is natural to consider applying this technology to design or optimize a nonlinear control system automatically for the complex nonlinear process.

Aiming at the situation that there is no systematic and effective method in the design of the nonlinear ILC, a novel design method is proposed in this paper based on the DRL

*This work is supported by Tencent AI Lab Rhino-Bird Focused Research Program (No.JR202003) and National Natural Science Foundation (NNSF) of China (61573295).

algorithm. This design method applies a mature DRL algorithm, which is called deep deterministic strategy gradient (DDPG), to automatically optimize a nonlinear IUL without any requirement of process modeling except for the interaction with the process and properly choosing the state information and designing the instant reward function. Through the numerical simulation on a nonlinear repetitive model, we demonstrate the effectiveness and applicability of the proposed design methods. We believe that the proposed design scheme in this paper provides a feasible solution to the control and decision-making problems of the complex process based on the artificial intelligence (AI) method.

2 Problem description

Consider a complex periodic/repetitive/batch process that can be described by the following nonlinear state space model:

$$\begin{cases} \dot{x}_k(t) = f\left(x_k(t), u_k(t), t\right) \\ y_k(t) = g\left(x_k(t), u_k(t), t\right) \end{cases}, \quad 0 \le t \le T, k = 1, 2, \cdots \tag{2}$$

where t indicates the continuous-time, k is the cycle (repetitive or batch) index, T represents the constant time duration of the dynamic within the cycle, $u_k(t), x_k(t), y_k(t)$ represent, respectively, the input, state and output of the process at time t of the k th cycle, nonlinear functions $f(\bullet,\bullet,\bullet)$ and $g(\bullet,\bullet,\bullet)$ describe the dynamics of the state and the output of the process, respectively.

For repetitive process (2), we consider the following discrete-time ILC law:

$$u_k(i) = u_{k-1}(i) + L\left(\mathbf{u}_k(i), \mathbf{x}_k(i), \mathbf{y}_k(i)\right) \tag{3}$$

where, i indicates the discrete-time index under the sampling period T_s, $\mathbf{u}_k(i), \mathbf{x}_k(i), \mathbf{y}_k(i)$ represents, respectively, all accessible historical input, state and output information at the time t of the k th cycle, and $L(\bullet,\bullet,\bullet)$ is a nonlinear IUL to be designed.

For the ILC system composed of process (2) and ILC law (3), the following cost function is generally considered to ensure the closed-loop control performance

$$J\left(\mathbf{u}_k\right) = \sum_{i=0}^{I} \alpha_i \left(e_k(i)\right)^2 + \sum_{i=0}^{I} \beta_i \left(u_k(i) - u_k(i-1)\right)^2 + \sum_{i=0}^{I} \eta_i \left(u_k(i) - u_{k-1}(i)\right)^2 \tag{4}$$

where, $e_k(i)$ represents the control error at the time i of the k th cycle, i.e. $e_k(i) = y_r(i) - y_k(i)$, where $y_r(i)$ represents the setpoint at the time i, and $\alpha_i, \beta_i, \eta_i > 0$ are the weighting factors of the terms. Obviously, the first term on the right-hand side indicates the tracking performance of the outputs to the setpoints within a cycle, while the second and third terms represent, respectively, the weighted quadratic summation of the control increments along the time and cycle indexes. According to the above cost function, the design objectives of the ILC system can be described as follows:

Design objective: For the periodic/repetitive/batch processes described by model (2), an ILC law (3) is designed to ensure that the cost (4) converges to a minimum value along with the cycle index as quickly as possible.

3 Deep Deterministic Policy Gradient (DDPG)

Reinforcement learning (RL) algorithm is proposed to solve the optimal decision-making problems of the dynamic processes. An RL system is mainly composed of an agent and an environment with architecture as shown in Fig. 1, where the agent can not only give the real-time control action to the environment according to the state feedback information, at the same time, it also can optimize its own control policy according to the assessment information (reward) given by the environment, which leads to the optimal decision-making gradually. Due to the capability of learning from interactive information and realizing self-optimization online, RL algorithm generally applied to solve complex decision-making problems.

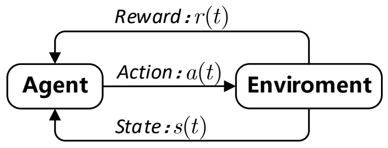


Fig. 1. Architecture of the reinforcement learning system.

In addition to the agent and the environment, the elements as follows is also required to describe an RL system:

- State $s(t)$: the real-time information feedback from the environment to the agent, defined by $s(t) \in S$, where S notes the admissible sets of the state.
- Policy $\pi(s)$: defining the decision-making policy of the agent, that is, the actions $a(t) \in A$ given by the agent according to the state information, where A represents the admissible actions of the environment. From a control system point of view, the policy is equivalent to a feedback controller described by $a(t) = \pi(s)$.
- Reward $r(t)$: defining the optimization task of the RL system. The instant reward is obtained from the real-time interaction. The task of the RL algorithm is to maximize the cumulative discount reward (CR) at the step:
$$R(t) = r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + \cdots \tag{5}$$
where γ is the discount factor.
- Value $V(t)$: indicating the maximum CR that can be obtained in the subsequent at time t. According to the optimization algorithm, value can be defined as the state value $V(s)$ or the state-action value $Q(s,a)$. Obviously, based on the accurate estimation of the state-action value $Q(s,a)$, the optimal policy can be determined according to the greed policy as follows:
$$a^*(t) = \pi^*(s) = \arg\max_{a \in A} Q(s,a) \tag{6}$$
- Model $s' = P(s,a)$: describing the dynamic response of the environment to the actions of the agent. It can be seen from this description that the application scenario of the RL is generally Markov decision process (MDP).

It can be seen from the above description that RL system is essentially a closed-loop feedback control system. The environment is equivalent to the plant to be controlled, while the agent mainly plays the role of state feedback controller. Compared with the traditional feedback controller, the agent can not only act as the online feedback controller, but also realize the self-optimization only based on the on-line interaction with the environment. It can be seen from the algorithms that RL is a data-driven optimization method. Therefore, for the complex processes, it is reasonable to to design a closed-loop control law by using RL algorithm, so as to realize intelligent and optimal design of the control system.

According to the dynamic characteristics of the process and the optimization methods used in RL, a variety of RL algorithms have been proposed[17]. Since the process considered in this paper is a deterministic dynamic process with continuous state and control input, the RL algorithm, which is called deep deterministic policy gradient (DDPG) [18], is considered in this paper. DDPG algorithm has the main fetures as follows:

- DDPG is suitable for the complex deterministic processes with continuous-time state and action. Obviously, model (2) describes a nonlinear system with deterministic dynamics and continuous-time input, state and output. Therefore, DDPG algorithm can be used for the design of optimal control law.
- The policy and value function are represented by DNN, so it essentially results in a nonlinear control scheme. Based on the self-optimization algorithm, it is may results in a optimal control performance.
- This algorithm uses the tricks of experience replay and the separation of the target network and online network to improve the stability of the training. Many successful applications have demonstrated the good stability of DDPG.

The framework of DDPG algorithm is shown in Fig. 2, where it is seen that the actor-critic framework is adopted in algorithm, and the experience replay buffer and the separated target and online networks are used in the algorithm to garantee the data efficiency and the stability of training.

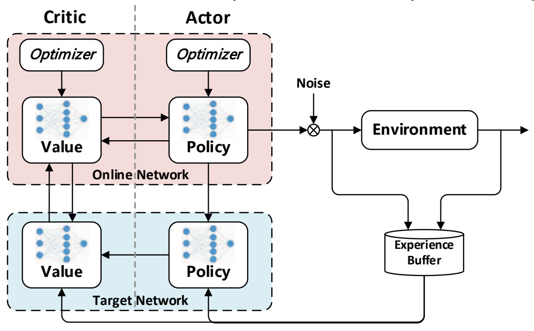


Fig. 2: the block diagram of DDPG algorithm

Algorithm 1 gives the pseudocode of the DDPG algorithm. In the next section, it will be used to design the nonlinear ILC system.

Algorithm 1: DDPG algorithm[18]

```
1. Initialize the experience buffer with data capacity $N$ .
2. Initialize the online value network $Q(s,a;\theta_Q)$ with random
   parameter values $\theta_Q$ , and initialize the target value network
   $Q'(s,a;\theta_{Q'})$ with the same parameter values: $\theta_{Q'}=\theta_Q$ .
3. Initialize the online actor network $\pi(s;\theta_\pi)$ with random
   parameter values $\theta_\pi$ , and initialize the target actor network
   $\pi'(s;\theta_{\pi'})$ with the same parameter values: $\theta_\pi=\theta_{\pi'}$ .
4. For each training step $i$ , do:
4-1. For the current state $s_i$ , choose the action $a_i$ as follows:
        $a_i=\pi(s_i;\theta_\pi)+\mathrm{N}$     (7)
     where N is stochastic noise configured for exploration.
4-2. Execute action $a_i$ , and then observe the state $s_{i+1}$ and
     instant reward $r_i$ at the next sampling step.
4-3. Store the experience $(s_i,a_i,r_i,s_{i+1})$ in the replay buffer.
4-4. Sample a random mini-batch of $M$ experiences from the
     replay buffer and compute the target values $y_i$ as follows:
        $y_i=\begin{cases} r_i & s_i \text{ is end} \\ r_i+\gamma Q'_{\pi'}(s_{i+1},\pi'(s_{i+1};\theta_{\pi'});\theta_{Q'}) & \text{otherwise} \end{cases}$     (8)
     where $\gamma$ is the discount factor.
4-5. Update the parameters of the online value network by
     minimizing the following loss across the mini-batch data:
        $L(\theta_Q)=\frac{1}{M}\sum_i\left(y_i-Q(s_i,a_i;\theta_Q)\right)^2$     (9)
4-6. Update the parameters of the online actor network by using
     the following sampled policy gradient to maximize the
     expected discounted reward:
        $J(\theta_\pi)=\frac{1}{M}\sum_i Q(s_i,a_i;\theta_Q)$     (10)
        $\nabla_{\theta_\pi}J(\theta_\pi)\approx\frac{1}{M}\sum_i G_{ai}G_{\pi i}$     (11)
     where $G_{ai}=\nabla_{a_i}Q(s_i,a_i;\theta_Q)$, $a_i=\pi(s_i;\theta_\pi)$ is the gradient
     of the output of value network with respect to the action
     computed by the actor network, and $G_{\pi i}=\nabla_{\theta_\pi}\pi(s_i;\theta_\pi)$ is
     the gradient of the output of the actor network with respect
     to the actor parameters.
4-7. Update the parameters of the target value network
     $Q'(s,a;\theta_{Q'})$ and target actor network $\pi'(s;\theta_{\pi'})$ as follows
     at every step using smoothing factor $\tau$ :
        $\begin{cases}\theta_{Q'}=\tau\theta_Q+(1-\tau)\theta_{Q'} \\ \theta_{\pi'}=\tau\theta_\pi+(1-\tau)\theta_{\pi'}\end{cases}$     (12)
```

4 Design scheme of the ILC system

It can be seen from the Algorithm 1 that the state, policy and instant reward must be designed properly in order to realize the optimal control.

4.1. Design of the state

In principle, all observable information of the process can be used as the state of the RL. For the repetitive processes, the closed-loop control system is essentially becoming a two-dimensional (2D) dynamic system with the ILC law (3) implemented. In order to ensure the Markov properties of the environment for the RL algorithm, it is needed to select sufficient historical data of the current cycle as well as the data information of the previous cycle around the current time as the state information, namely

$$s_k(i)=\begin{pmatrix}\cdots & p_k(i-1) & p_k(i) & & \\ \cdots & p_{k-1}(i-1) & p_{k-1}(i) & p_{k-1}(i+1) & \cdots\end{pmatrix} \tag{13}$$

where, super vector $p_k(i)$ indicates any observable process information at the sampling time i of the k th cycle. The first row of the above state represents the observable historical information of the current cycle, while the second row represents the observable process information of the previous cycle around the current time. The first principle for designing the state is to ensure the Markov dynamic transition from state to state, which ensures the convergence of RL algorithm. Theoretically, Markov dynamic can be guaranteed when the sufficient 2D information of the process is included in the state $s_k(i)$, but too large state dimension will dramatically increase the computational burden of the RL algorithm. Therefore, in order to ensure the training efficiency of RL algorithm, the dimensions of state information should be reasonably designed according to the dynamic complexity of the process.

4.2. Design of the policy

For the ILC law (3), it is essentially the nonlinear function $L(\bullet,\bullet,\bullet)$ to be designed and optimized. Therefore, the policy of RL algorithm should be defined as follows:

$$a_k(i) = u_k(i) - u_{k-1}(i) = \pi\left(s_k(i);\theta_\pi\right) = L\left(\mathbf{u}_k(i),\mathbf{x}_k(i),\mathbf{y}_k(i)\right) \quad (14)$$

The parameters of the neural networks are optimized by RL algorithm. The structure of the policy network need to be seted appropriately according to the complexity of the system. Generally, for the systems with more complex dynamic, neural networks with more hidden layers and more neuron nodes in each layer should be configured to ensure the approximation of the neural network to the optimal nonlinear control law.

4.3. Design of the instant reward

The design of the instant reward is not only important to ensure the convergence of RL algorithm, but also directly determines the control performance of the closed-loop ILC system. The design of the instant reward needs to be considered according to the cost function of the ILC. For the cost function (4), the instant reward is generally determined by the following quadratic function:

$$r_k(i) = -\alpha_i\left(e_k(i)\right)^2 - \beta_i\left(u_k(i)-u_k(i-1)\right)^2 - \eta_i\left(u_k(i)-u_{k-1}(i)\right)^2 \quad (15)$$

Considering the constant time duration of the cycle, the discount factor is generally set by $\gamma=1$. The disadvantage of the above reward is that the obtained reward is always negative. In practical applications, in order to improve the learning rate of the system, appropriate positive rewards should be given when the control peromance is good enough. For example, the following instant reward can be adopted:

$$r_k(i) = \begin{cases} -\alpha_i\left(e_k(i)\right)^2 - \beta_i\left(u_k(i)-u_k(i-1)\right)^2 - \eta_i\left(u_k(i)-u_{k-1}(i)\right)^2, & \left(e_k(i)\right)^2 > \varepsilon \\ C - \beta_i\left(u_k(i)-u_k(i-1)\right)^2 - \eta_i\left(u_k(i)-u_{k-1}(i)\right)^2 & ,\left(e_k(i)\right)^2 \le \varepsilon \end{cases} \quad (16)$$

where $C>0$ is an appropriate positive constant.

Based on the above designs, the following RL algorithm is used to design the nonlinear IUL:

Algorithm 2: IUL design based on DDPG algorithm:

```
1. Initialize the experience buffer with data capacity $N$.
2. Initialize the online value network $Q\left(s,a;\theta_Q\right)$ with random parameter values $\theta_Q$, and initialize the target value network $Q'\left(s,a;\theta_{Q'}\right)$ with the same parameter values: $\theta_Q = \theta_{Q'}$.
3. Initialize the online actor-network $\pi\left(s;\theta_\pi\right)$ with random parameter values $\theta_\pi$, and initialize the target actor-network $\pi'\left(s;\theta_{\pi'}\right)$ with the same parameter values: $\theta_\pi = \theta_{\pi'}$.
4. For each training step $k$, do:
  4-1. Initialize the state of the process.
  4-2. For each training step $i$, do:
    4-2-1. For the current state $s_k(i)$, determine the action $a_k(i)$ as follows:
           $$a_k(i) = \pi\left(s_k(i);\theta_\pi\right)+\mathrm{N} \quad (17)$$
           where N is stochastic noise configured for exploration.
    4-2-2. Compute the output of the ILC law $u_k(i) = u_{k-1}(i) + a_k(i)$ and output the control $u_k(i)$ to the process, then observe the process information $p_k(i+1)$ and computer the instant reward $r_k(i)$ according to (15) or (16) at the next sampling step.
    4-2-3. Store the experien $\left(s_k(i),a_k(i),r_k(i),s_k(i+1)\right)$ in the experience buffer.
    4-2-4. Sample a random mini-batch of $M$ experiences from the experience buffer and compute the target values $q_k(i)$ as follows:
           $$q_k(i) = \begin{cases} r_k(i) & i \text{ is the end} \\ r_k(i)+\gamma Q'_{\pi'}\left(s_k(i+1),\pi'\left(s_k(i+1);\theta_{\pi'}\right);\theta_{Q'}\right) & \text{otherwise} \end{cases} \quad (18)$$
           where $\gamma$ is the discounted factor.
    4-2-5. Update the parameters of the online value network by minimizing the following loss across all sampled experiences:
           $$L\left(\theta_Q\right) = \frac{1}{M}\sum_i\left(q_k(i) - Q\left(s_k(i),a_k(i);\theta_Q\right)\right)^2 \quad (19)$$
    4-2-6. Update the parameters of the online actor-network by using the following sampled policy gradient to maximize the expected discounted reward:
           $$J\left(\theta_\pi\right) = \frac{1}{M}\sum_i Q\left(s_k(i),a_k(i);\theta_Q\right) \quad (20)$$
           $$\nabla_{\theta_\pi}J\left(\theta_\pi\right) \approx \frac{1}{M}\sum_i G_{ak}(i)G_{\pi k}(i) \quad (21)$$
           where $G_{ak}(i) = \nabla_{a_k(i)}Q\left(s_k(i),a_k(i);\theta_Q\right), a_k(i)=\pi\left(s_k(i);\theta_\pi\right)$ is the gradient of the output of value network with respect to the action computed by the actor-network, and $G_{\pi i} = \nabla_{\theta_\pi}\pi\left(s_i;\theta_\pi\right)$ is the gradient of the output of the actor-network with respect to the actor parameters.
    4-2-7. Update the parameters of the target value network $Q'(s,a;\theta_{Q'})$ and target actor-network $\pi'\left(s;\theta_{\pi'}\right)$ as follows at every step using smoothing factor:
           $$\begin{cases} \theta_{Q'}=\tau\theta_Q+(1-\tau)\theta_{Q'} \\ \theta_{\pi'}=\tau\theta_\pi+(1-\tau)\theta_{\pi'} \end{cases} \quad (22)$$
  4-3. Calculate the time-wise control performance index $J\left(\mathbf{u}_k\right)$ according to cost(4). If $\left|J\left(\mathbf{u}_k\right)-J\left(\mathbf{u}_{k-1}\right)\right|$ is small enough, define the following IUL function and stop the cycle loop:
       $$L\left(\mathbf{u}_k(i),\mathbf{x}_k(i),\mathbf{y}_k(i)\right) = \pi\left(s_k(i);\theta_\pi\right) \quad (23)$$
```

5 Numerical simulation

In order to verify the effectiveness of the proposed design scheme, a nonlinear repetitive process is considered and modeled as follows:

$$\begin{cases} \dfrac{dy_k(t)}{dt}+\sin\left(y_k(t)\right) = a(t)u_k(t) & 0 \le t \le 200 \\ y_k(0) = 0 \end{cases} \quad (24)$$

where $a(t)=0.5+0.3\sin(0.1t)$ is a nonlinear signal, and the second term on the left-hand side leads to the nonlinearity of the process.

For the above repetitive process, we set the sampling period $T_s = 1$, and then applied Algorithm 2 to design the ILC law. The super state vector is designed as follows:

$$s_k(i) = \begin{pmatrix} e_k(i-2) & e_k(i-1) & e_k(i) & u_k(i-2) & u_k(i-1) & u_k(i) & y_k(i-2) & y_k(i-1) & y_k(i) \\ e_{k-1}(i-2) & e_{k-1}(i-1) & e_{k-1}(i) & u_{k-1}(i-2) & u_{k-1}(i-1) & u_{k-1}(i) & y_{k-1}(i-2) & y_{k-1}(i-1) & y_{k-1}(i) \\ y_r(i) \end{pmatrix} \tag{25}$$

The above state is composed of 19 entries, where the entries in the first row are the process information of the current cycle, and the entries in the second row are the process information of the previous cycle. Based on the above state, the IUL obtained by RL algorithm will be a nonlinear function approximated by a ANN with 2D feedback information as the input, which leads to the closed-loop ILC system potentially composed of a real-time feedback control along time within the cycle and a feedforward ILC along the cycle. This nonlinear 2D control scheme can guarantee both the time-wise and cycle-wise control performances.

The instant reward in the simulation is defined as follows:

$$r_k(i) = \begin{cases} -\left(e_k(i)\right)^2 - 0.2\left(u_k(i) - u_k(i-1)\right)^2 - 0.1\left(u_k(i) - u_{k-1}(i)\right)^2, & \left(e_k(i)\right)^2 > 0.01 \\ 1 - 0.2\left(u_k(i) - u_k(i-1)\right)^2 - 0.1\left(u_k(i) - u_{k-1}(i)\right)^2 & ,\left(e_k(i)\right)^2 \le 0.01 \end{cases} \tag{26}$$

Based on the above designs, the design for the ILC is conducted by using Algorithm 2. Fig. 3 shows the trend of the cumulative rewards (CR) obtained during the training. It can be seen that the CR values gradually increases with training cycles, which demonstrates the effectiveness and convergence of the proposed design algorithm.

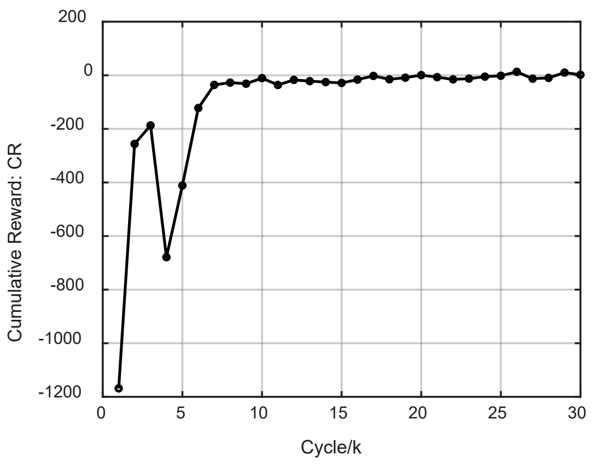
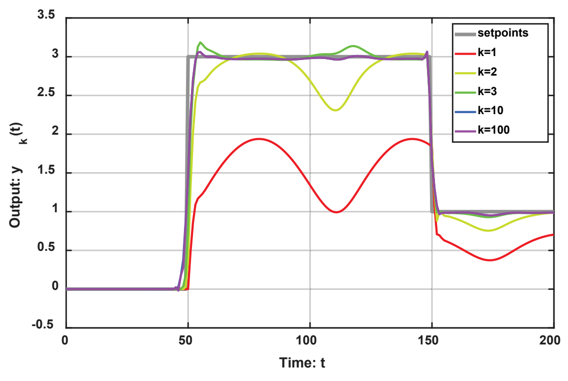


Fig. 3: Cumulative reward curve of the training

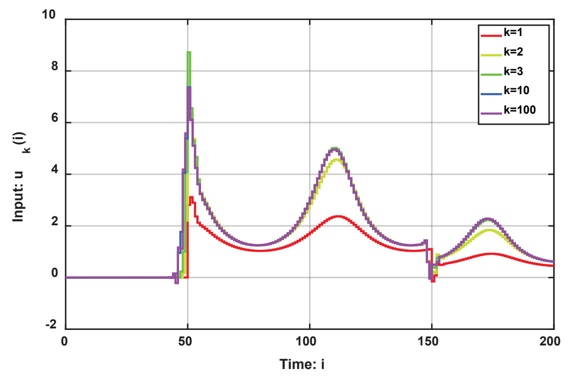
After 30 cycles of training, the obtained ILC law is implemented, and the simulation of the ILC system is conducted. Fig. 4 shows the output responses and the corresponding control inputs of the ILC system in the 1st, 2nd, 3rd, 10th and 100th cycles. It can be seen that the control performance gradually improves alone the cycle index. In order to assess the convergence of the ILC system along the cycle index, we defined the sum of absolute error (SAE) as the cycle-wise performance index:

$$\text{SAE}\left(\mathbf{u}_k\right) = \sum_{i}^{I} \left|e_k(i)\right| \tag{27}$$

Fig. 5 shows the variation of the SAE values along the cycle, which demonstrates that a good convergence along the cycle index is obtained. The simulation results illustrate the effectiveness of the proposed design method.



(a)



(b)

Fig. 4: Simulation results: (a) output responses ,(b) control inputs

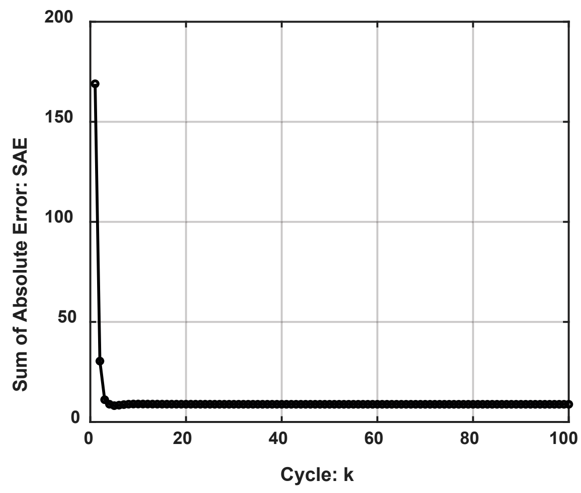


Fig. 5: Control performance along the cycle

6 Conclusions

Aiming at complex periodic/repetitve/batch processes, this paper proposes a nonlinear ILC design scheme based on DRL algorithm. This design scheme can automatically realize the optimal design of nonlinear IUL by simply designing the state information and instant reward used by RL algorithm. The numerical simulation results based on the nonlinear model illustrate the effectiveness and feasibility of the proposed design method. The proposal of this design scheme provides a feasible solution for the design of nonlinear ILC based on the artificial intelligence technology. The disadvantage of the proposed design scheme is that the implementation of the algorithm needs to interact with the process, which may have the risk for some industry applications. How to ensure the applicability of the proposed design method in more practical applications is a crucial problem that needs a further study in the future.

References

[1] Arimoto S, Kawamura S, Miyazak F. Bettering Operation of Robots by Learning[J]. *Journal of Robotic Systems*, 1984, 1(2): 123-140.

[2] Lee J H, Lee K S. Iterative learning control applied to batch processes: An overview[J]. *Control Engineering Practice*, 2007, 15(10): 1306-1318.

[3] Koo P-H, Moon D H. A Review on Control Strategies of Batch Processing Machines in Semiconductor Manufacturing[J]. *IFAC Proceedings Volumes*, 2013, 46(9): 1690-1695.

[4] Wang Y-C, Chien C-J, Chi R, et al. A Fuzzy-Neural Adaptive Terminal Iterative Learning Control for Fed-Batch Fermentation Processes[J]. *International Journal of Fuzzy Systems*, 2015, 17(3): 423-433.

[5] Lee K S, Lee J H. Iterative learning control-based batch process control technique for integrated control of end product properties and transient profiles of process variables[J]. *Journal of Process Control*, 2003, 13(7): 607-621.

[6] Jiang P, Unbehauen R. Robot visual servoing with iterative learning control[J]. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 2002, 32(2): 281-287.

[7] Hermanto M W, Braatz R D, Min-Sen C. A run-to-run control strategy for polymorphic transformation in pharmaceutical crystallization[C]. *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, 2006: 2121-2126.

[8] Pandit M, Buchheit K. Optimizing iterative learning control of cyclic production processes with application to extruders[J]. *IEEE Transactions on Control Systems Technology*, 1999, 7(3): 382-390.

[9] Wang Y, Gao F, Doyle F J. Survey on iterative learning control, repetitive control, and run-to-run control[J]. *Journal of Process Control*, 2009, 19(10): 1589-1600.

[10] Hara S, Yamamoto Y, Omata T, et al. Repetitive control system: a new type servo system for periodic exogenous signals[J]. *IEEE Transactions on Automatic Control*, 1988, 33(7): 659-668.

[11] Harte T J, Hätönen J, Owens D H. Discrete-time inverse model-based iterative learning control: stability, monotonicity and robustness[J]. *International Journal of Control*, 2005, 78(8): 577-586.

[12] Shi J, Yang B, Cao Z, et al. Two-dimensional generalized predictive control (2D-GPC) scheme for the batch processes with two-dimensional (2D) dynamics[J]. *Multidimensional Systems and Signal Processing*, 2015, 26(4): 941-966.

[13] Nian R, Liu J, Huang B. A review On reinforcement learning: Introduction and applications in industrial process control[J]. *Computers & Chemical Engineering*, 2020, 139: 106886.

[14] Xiaohui D, Chi-Kwong L, Rad A B. An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control[J]. *IEEE Transactions on Intelligent Transportation Systems*, 2005, 6(3): 285-293.

[15] Zhao X, Xia L, Zhang L, et al. Deep reinforcement learning for page-wise recommendations[C]. *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018: 95–103.

[16] Tsantekidis A, Passalis N, Toufa A S, et al. Price Trailing for Financial Trading Using Deep Reinforcement Learning[J]. *IEEE Transactions on Neural Networks and Learning Systems*, 2020: 1-10.

[17] Sutton R S, Barto A G. Reinforcement Learning: An Introduction[J]. *IEEE Transactions on Neural Networks*, 1998, 9(5): 1054-1054.

[18] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. *Computer Science*, 2015.