Reinforcement learning based iterative learning control for nonlinear batch process with non-repetitive uncertainty via Koopman operator

Hongfeng Tao[*], Yuan Huang [*], Tao Liu[*][†], Wojciech Paszke[‡]

08.08.2024

Abstract

To tackle the time and batchwise uncertainty involved in nonlinear batch process, this paper proposes a deep reinforcement learning (DRL) based ILC scheme via Koopman operator. Using the Koopman operator, the original nonlinear system is reformulated into a high-dimensional linear space form. Then, a DRL agent with neural network is introduced into the 2D ILC framework to compensate for non-repetitive uncertainty. Meanwhile, the convergence conditions of the proposed ILC scheme are analyzed with a proof through the linear matrix inequality. The proposed method is subsequently applied to a continuous stirring tank reactor (CSTR), and the output of the high-dimensional linear model is found to be almost identical to that of the original nonlinear model, with an output error of less than 5%. In contrast to the linearized ILC near the equilibrium point, the control performance of the ILC under the high-dimensional model based on Koopman operator is conspicuously augmented. Eventually, the utilisation of deep reinforcement learning to address non-repetitive uncertainty and model mismatch significantly enhances the control performance in comparison to dynamic iterative linearisation and PD-type classical ILC approach.

[*]Key Laboratory of Advanced Process Control for Light Industry of Ministry of Education, Jiangnan University, Wuxi 214000, PR China, CONTACT Hongfeng Tao. Email: taohongfeng@jiangnan.edu.cn

[†]Institute of Advanced Control Technology, Dalian University of Technology, Dalian 116024, PR China, CONTACT Tao Liu. Email: liurouter@ieee.org

[‡]Institute of Automation, Electronic and Electrical Engineering, University of Zielona Góra, ul. Szafrana 2, 65-246 Zielona Góra, Poland

Keywords: Iterative learning control, Nonlinear batch process, Koopman operator, Deep reinforcement learning, Non-repetitive uncertainty

1 Introduction

Various batch processes have been constructed in industrial production, encompassing automatic coal mining [1], metal rolling [2], robotic assembly [3, 4], fine chemical manufacturing [5] and so on. ILC is an advanced algorithm designed specifically for performance optimization, enabling continuous learning and adjustment of control parameters based on historical performance. By capitalizing on the inherent periodic data of batch processes, ILC systematically evaluates control efficacy and revises the input signals iteratively, thereby enhancing control performance from batch to batch [6]. Despite advancements in ILC, there are still some challenges to be overcome. The existing ILC methods were mainly designed for strictly repetitive processes [7], subject to the control performance degradation in variable production environments and the inevitable errors in machinery [8]. Although a 2D ILC scheme, which fully takes into account the 2D system nature of ILC and can quickly converge to the desired output in both time and batch directions, has been developed to handle non-repetitive model mismatches and can achieve good output performance with a finite model perturbation[9], there are modified approaches integrating 2D ILC with predictive control [10], adaptive control [11], machine learning [12], reinforcement learning [13] for dealing with non-repetitive uncertainty. Note that the developed 2D ILC framework is generally based on a linear system description. In fact, there are more and more industrial batch processes involved with complex nonlinearity, which causes difficulties in applying such a 2D ILC scheme.

To deal with nonlinear batch process dynamics, Chi et al. proposed an iterative dynamic linearization scheme to update the ILC controller parameters in real time using the linearized model, and then convergence proof and numerical simulations were performed to verify the effectiveness of the algorithm [14, 15]. Subsequently, Chi extended their proposed iterative dynamic linearization ILC to high order and proposed the method of data-driven high order optimal iterative learning control (DDHOILC), which achieved better tracking performance [16]. Meng proposed a new optimization-based design for adaptive ILC that was subject to unknown time-varying nonlinearities and non-repetitive uncertainties, and introduced a new robust convergence analysis, finally the validity of the method was verified by two numerical simulations [17]. Xing took the non-repetitive uncertainties of initial shifts, reference trajectories, trial lengths into consideration in the ILC design, and proposed a 2D adaptive ILC approach, finally a convergence proof was completed using Lyapunov stability principle and the excellent simulation control results on MIMO nonlinear systems were given [18]. Some scholars have also combined artificial intelligence with ILC, Chen utilized neural networks as a compensator for ILC controllers, and finally achieved good control results on mobile robot which was nonlinear with non-repetitive uncertainties[19]. Liu et al. used a deep reinforcement learning agent as a control compensator and a 2D ILC as a master controller, which basically achieved zero-error tracking of the desired output after training the agent [13].

To address non-repetitive uncertainty in nonlinear industrial batch processes, researchers have proposed some effective solutions through extensive experiments and achieved good control performance. However, most control schemes were based on the 2D ILC and the controller parameters were obtained based on the equilibrium point linearization. Furthermore, it was difficult to ensure the accuracy of the control performance of the ILC controller transplanted to nonlinear systems. In addition, other ILC schemes for nonlinear objects such as the iterative dynamic linearization scheme, needs to satisfy the Lipschitz condition. Therefore, a 2D ILC control scheme based on Koopman operator theory is proposed, which can lift the nonlinear system to high dimensional linear space and achieve the exact approximation of the original nonlinear system.

Koopman operator has found wide applications in fields such as fluid dynamics [20], meteorology [21], and power grids [22]. The core idea of Koopman theory is to lift a nonlinear system into an infinite-dimensional space. Theoretically, as the dimension of this lifted space approaches infinity, it enables an exact approximation of the original nonlinear system and achieves global linearization [23]. Currently, the main Koopman approximation methods for nonlinear systems include dynamic mode decomposition (DMD), extended dynamic mode decomposition (EDMD), and deep extended dynamic mode decomposition (Deep-EDMD). The DMD method is primarily used for autonomous systems, while EDMD and Deep-EDMD methods are mainly applied to controlled systems. However, the DMD method relies on manual selection of the eigenfunctions. In contrast, the Deep-EDMD method employs artificial neural networks to replace the eigenfunctions. Artificial neural network has been extensively employed for the forecasting of agricultural [24], energy commodity [25], carbon emission allowance [26] prices and futures trading volume [27]. These applications have yielded exemplary prediction outcomes, which substantiate the capacity of neural networks to process intricate nonlinear and extensive high-dimensional datasets. Overall, the use of deep neural networks instead of eigenfunctions exploits the high expressiveness of neural networks and eliminates the uncertain process of manually selecting eigenfunctions, resulting in more accurate prediction results.

However, the high dimensionality and batch-invariant uncertainty of the system make it challenging for 2D ILC to achieve rapid convergence of the system output to the desired output. With the development and application of artificial intelligence, reinforcement learning (RL) has become a crucial research area for decision-making and optimization. Deep reinforcement learning (DRL) combines deep neural networks with RL and is widely used in fields such as strategy games, complex machinery control [28], and network resource provisioning [29]. DRL utilizes networks to approximate value and policy functions, addressing the continuous state space problem that traditional reinforcement learning cannot solve. Additionally, with the advancements in DRL, the Soft Actor-Critic(SAC) algorithm introduced strategy entropy to the traditional Bellman Equation [30], significantly enhancing the agent's ability to explore the optimal strategy randomly. Furthermore, the SAC algorithm characterises the actions in accordance with its environment as a Gaussian process, which is highly regarded for its capacity to address uncertainty in quantitative prediction. Gaussian process has demonstrated remarkable performance in a range of domains, including coffee prices [31], thermal coal prices [32], steel and coking coal futures price index [33, 34] regression. However, the actual production process is often subject to unpredictable Gaussian noise, which can impede the efficacy of the SAC algorithm. In such cases, the use of compensated control signals that adhere to a Gaussian distribution can prove advantageous in mitigating the impact of uncertainty disturbances.

To cope with the high dimensionality, time and batch-invariant uncertainty of the system, this paper proposes a DRL-compensated ILC scheme based on Koopman operator. Firstly, the nonlinear differential equation model of batch process is extended to a high-dimensional space model by Deep-EDMD. The linear matrix inequality conditions for the convergence of the nominal system are derived based on the 2D ILC , then the ILC serves as the primary controller to ensure the control performance of the batch run, while the DRL agent acts as the secondary controller to mitigate the effects of model mismatch and non-repetitive uncertainty. The main contributions of this paper are summarized as follows:

1. Deep Koopman operator networks are employed to substitute for the eigenfunction of the nonlinear batch process and the system matrix of the high-dimensional state space model. This enables the high-dimensional linearisation and precise approximation of the original nonlinear model to be achieved.

2. Reinforcement learning is employed to address the issue of non-repetitive model mismatch and interference. Additionally, Gaussian noise is added to the non-repetitive perturbation. The efficacy of reinforcement learning in addressing Gaussian noise is substantiated by experimental evidence.

3. In comparison to the findings of [13], this paper puts forth an enhanced PD-type ILC control law incorporating state error, and modifies the reward structure. The ILC control methodology based on the Koopman operator demonstrates superior precision. These enable the agent to identify the optimal compensation control signal with greater alacrity, thereby accelerating the convergence.

The remainder of the paper will be organized in the following way: Section 2 introduces the control object of this paper. Section 3 illustrates the Koopman operator theory and EDMD, presenting the structure of the network in Deep-EDMD as well as the loss function. Section 4 introduces the 2D ILC-DRL control algorithm. Section 5 applies the proposed control scheme to a nonlinear continuously stirred tank reactor. Finally, conclusions will be illustrated in Section 6.

2 Control object

Consider a nonlinear batch process as generally described below

$$\begin{cases} x(t+1) = f(x(t), u(t)) + d(k,t) \\ y(t+1) = g(x(t+1)) \\ x(0) = x_0 \end{cases} . \tag{1}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^r$, $y(t) \in \mathbb{R}^m$ are respectively the process state, input and output vectors, $f(\cdot)$ and $g(\cdot)$ represent the nonlinear vector functions, while the dimension matches with the other vectors; $t = 0, 1, 2, \ldots, T$, and T is the total time for each trial of the system; x_0 denotes the initial state for each trial; $d(k,t)$ represents non-repetitive uncertainty perturbations related to time t and batch k, and $k = 0, 1, 2 \ldots$.

Control objective: for the above nonlinear batch process with non-repetitive uncertainty, a deep Koopman operator will be designed to construct a linearized description, by introducing a DRL agent with neural network to compensate for the nonlinear process dynamics. Based on such system description, a 2D ILC-RL based on Koopman operator is developed to achieve fast and smooth tracking of the desired output trajectory.

3 Data-driven Koopman operator based system modeling

3.1 Koopman operator

Herein, the Koopman operator is utilized to establish an expression for a nonlinear system in a high-dimensional linear space, the general framework of which is shown in Fig. 1, which illustrates the transformation between state space and lifting space for a nonlinear system by the Koopman operator. In Fig. 1, the uncontrolled dynamical system is denoted by

$$x(t+1) = G(x(t)), \tag{2}$$

$G(\cdot) : \mathcal{M} \rightarrow \mathcal{M}$ is a nonlinear function and the Koopman operator $\mathcal{K} : \mathcal{F} \rightarrow \mathcal{F}$ is defined as

$$(\mathcal{K}\varPsi)(x) = \varPsi(G(x)), \tag{3}$$

$\varPsi : \mathbb{R}^n \rightarrow \mathbb{R}$ is called the Koopman eigenfunction, which is the space of lifting functions represented by the symbol $\mathcal{F}$. The Koopman operator $\mathcal{K}$ acts on the eigenfunction $\varPsi$ and transforms it on the function space, so that the transformation result can be obtained as the state value of the next moment through the modal ν. However, the Koopman operator in (3) is an infinite-dimensional linear operator. In practice, a finite-dimensional operator is approximated in place of the Koopman operator, and the approximated Koopman operator $\hat{\mathcal{K}}\psi_i$ is defined as

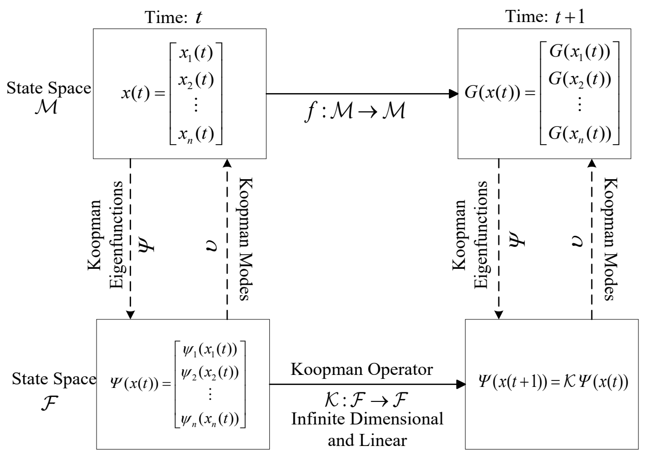
$$(\hat{\mathcal{K}}\psi_i)(x_i) = \mu_i\psi_i(x_i), \tag{4}$$

$\hat{\mathcal{K}} \in \mathbb{R}^{n\times n}$, μ_i denotes the ith eigenvalue of this linear operator, $i = 1, 2, ..., N$ and $N \gg n$. ψ_i denotes the eigenfunction corresponding to the ith eigenvalue. The state vector $\varPsi$ consists of the eigenfunctions that are in the lifting space $\mathcal{F}$, which is mapped back to the original state space $\mathcal{M}$ through the modal ν by

$$x(t) = \sum_{i=1}^{N} \nu_i\psi_i(x(t)), \tag{5}$$

ν_i denotes the ith mode, based on (4) and (5), the nonlinear system based on Koopman operator representation is written as

$$\begin{aligned} f(x(t)) &= \hat{\mathcal{K}} \sum_{i=1}^{N} \nu_i\psi_i(x(t)) \\ &= \sum_{i=1}^{N} \nu_i(\hat{\mathcal{K}}\varPsi)(x). \end{aligned}$$



Time: t
Time: $t+1$
State Space $\mathcal{M}$
$x(t)=\begin{bmatrix}x_1(t)\\x_2(t)\\\vdots\\x_n(t)\end{bmatrix}$
$f:\mathcal{M}\to\mathcal{M}$
$G(x(t))=\begin{bmatrix}G(x_1(t))\\G(x_2(t))\\\vdots\\G(x_n(t))\end{bmatrix}$
Koopman Eigenfunctions Ψ
υ Koopman Modes
State Space $\mathcal{F}$
$\Psi(x(t))=\begin{bmatrix}\psi_1(x_1(t))\\\psi_2(x_2(t))\\\vdots\\\psi_n(x_n(t))\end{bmatrix}$
Koopman Operator
$\mathcal{K}:\mathcal{F}\to\mathcal{F}$
Infinite Dimensional and Linear
$\Psi(x(t+1))=\mathcal{K}\Psi(x(t))$

Fig. 1. Illustration on the structure of Koopman operator.

Note that, the Koopman operator can also be used to describe a controlled nonlinear system. Correspondingly, the state vector of the system (1) is extended as

$$X(t)=\begin{bmatrix}x(t)\\u(t)\end{bmatrix}.$$

The extended state vector is the splice of the original state vector and the control vector, then the extended state vector at the next moment is

$$X(t+1)=F(X)=\begin{bmatrix}f(x(t),u(t))\\\mathcal{S}u(t)\end{bmatrix},$$

$\mathcal{S}$ is the forward time-shift operator and $\mathcal{S}u(t)=u(t+1)$. Define the eigenfunction in (3) in the extended form

$$\phi_i(x(t),u(t))=\psi_i(x(t))+\zeta u(t), \tag{6}$$

$\zeta:\mathbb{R}^r\to\mathbb{R}^r$ denotes a linear transformation, ψ_i is the ith eigenfunction of $x(t)$, and ϕ_i is the ith eigenfunction of the extended state vector $X(t)$. Define the Koopman operator under the extended state vector as

$$\phi_i(X(t+1))=\mathcal{K}\phi_i(X(t)), \tag{7}$$

where the Koopman lifting function is

$$\begin{aligned}\phi(X(t))&=\begin{bmatrix}\phi_1 & \phi_2 & \cdots & \phi_N\end{bmatrix}\\&=\begin{bmatrix}\psi_1(x(t)) & \psi_2(x(t)) & \cdots & \psi_N(x(t)) & u^T\end{bmatrix}^T.\end{aligned} \tag{8}$$

Define a finite-dimensional approximation of Koopman operator of (7) as

$$\phi(X(t+1)) = \hat{\mathcal{K}}\phi(X(t)), \tag{9}$$

where $\hat{\mathcal{K}} \in \mathbb{R}^{(N+r)\times(N+r)}$, and construct the lifted vector

$$\Phi(X(t))) = \begin{bmatrix}\psi_1(x(t)) & \psi_2(x(t)) & \cdots & \psi_N(x(t))\end{bmatrix},$$

then (9) can be reduced to

$$\Phi(x(t+1))) = \begin{bmatrix}A & B\end{bmatrix}\begin{bmatrix}\Phi(x(t)) \\ u(t)\end{bmatrix}, \tag{10}$$

$\mathcal{S} \in \mathbb{R}^{r\times r}$, $A \in \mathbb{R}^{N\times N}$, consists of the first N rows and N columns of matrix $\hat{\mathcal{K}}$, $B \in \mathbb{R}^{N\times r}$ consists of the first N rows and $N+1$ to $N+r$ columns of matrix $\hat{\mathcal{K}}$. Retaining the original state variables in the elevated state space ensures the continuity of the elevated state variables. Therefore, a new elevated state vector can be defined as

$$\chi(t) = \begin{bmatrix}x^T(t) & \Phi(x(t))\end{bmatrix},$$

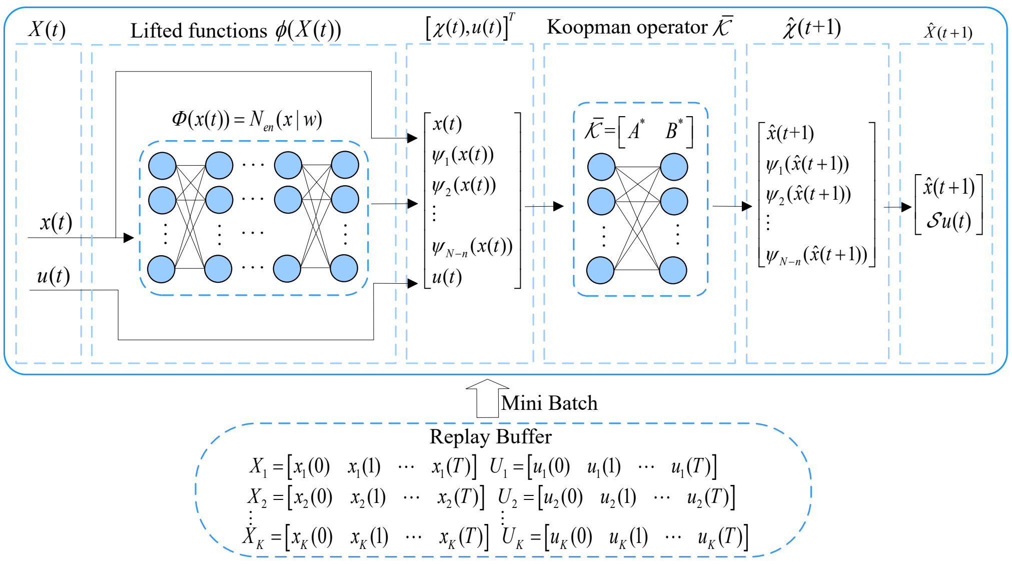
then the linear Koopman operator based model in the lifting space can be obtained as follow:

$$\begin{cases}\chi(t+1) = \begin{bmatrix}A^* & B^*\end{bmatrix}\begin{bmatrix}\chi(t) \\ u(t)\end{bmatrix} \\ y(t+1) = C^*\chi(t+1)\end{cases}, \tag{11}$$

where $A^* \in \mathbb{R}^{(n+N)\times(n+N)}$, $B^* \in \mathbb{R}^{(n+N)}$, $C^* \in \mathbb{R}^{m\times(n+N)}$, the new Koopman operator can be obtained as

$$\bar{\mathcal{K}} = \begin{bmatrix}A^* & B^*\end{bmatrix}.$$

According to the system structure shown in (11), Data-driven model based on the Koopman operator and neural network can be constructed as shown in Fig. 2. The Koopman operator neural network employs network $N_{en}(x|\omega)$ as an eigenfunction, elevating the nonlinear state to a high-dimensional linear space and combining it with the original nonlinear state variables and control variables to form an extension vector. This vector then passes through a two-layer network to yield the $\chi(t+1)$ at the subsequent moment.



$X(t)$
Lifted functions $\phi(X(t))$
$[\chi(t),u(t)]^T$
Koopman operator $\bar{\mathcal{K}}$
$\hat{\chi}(t+1)$
$\hat{X}(t+1)$
$\Phi(x(t)) = N_{en}(x|w)$
$x(t)$
$u(t)$
$x(t)$, $\psi_1(x(t))$, $\psi_2(x(t))$, $\psi_{N-n}(x(t))$, $u(t)$
$\bar{\mathcal{K}} = [A^* \quad B^*]$
$\hat{x}(t+1)$, $\psi_1(\hat{x}(t+1))$, $\psi_2(\hat{x}(t+1))$, $\psi_{N-n}(\hat{x}(t+1))$
$\hat{x}(t+1)$, $\mathcal{S}u(t)$
Mini Batch
Replay Buffer
$X_1 = [x_1(0) \; x_1(1) \; \cdots \; x_1(T)]$ $U_1 = [u_1(0) \; u_1(1) \; \cdots \; u_1(T)]$
$X_2 = [x_2(0) \; x_2(1) \; \cdots \; x_2(T)]$ $U_2 = [u_2(0) \; u_2(1) \; \cdots \; u_2(T)]$
$X_K = [x_K(0) \; x_K(1) \; \cdots \; x_K(T)]$ $U_K = [u_K(0) \; u_K(1) \; \cdots \; u_K(T)]$

Fig. 2. Data-driven model based on the Koopman operator and neural network.

3.2 Network design based on Deep-EDMD

In order to take advantage of the high expressiveness of a deep neural network, let $N_{en}(x|\omega) = \varPhi(x(t))$, then the eigenfunctions and Koopman operator $\bar{\mathcal{K}}$ are solved using a deep neural network. The initial state is selected according to the actual conditions, and the control vectors are randomly sampled in the range selected according to the actual situation to obtain K batches of training data, each of which is sampled by a time series of length T. To update the buffer data to achieve a more accurate approximation of the model, it is necessary to continuously sample the buffer data during training. Next, the optimized loss function will be established from three aspects: the reconstruction ability of the eigenfunction, the forward prediction ability, and the controllability of the system in the lifted space.

Concerning the reconstruction ability of the eigenfunction, according to Koopman's basic principle, the lifting function maps low-dimensional state vectors into high-dimensional space is invertible, and thus there exists a network $N_{de}(\varPhi|\omega_{de})$ such that the $\hat{x}(t)$ obtained by remapping high-dimensional state vectors into low-dimensional space is close to the actual values of the system [35], which shows in Fig. 3. Therefore, the first loss function can be defined as

$$L_1(\omega) = ||x(t) - N_{de}(N_{en}(x(t)|\omega))||. \tag{12}$$

Additionally, the established data-driven model must be able to accurately predict the lifted state vector of the system at the next moment and the state vector in the low-dimensional space. Therefore, the second loss function is defined as

$$L_2(\omega, A^*, B^*) = ||\chi(t+1) - \hat{\chi}(t+1)|| + ||x(t+1) - \hat{x}(t+1)||. \tag{13}$$

Subsequently, to ensure that the established lifting state space model is

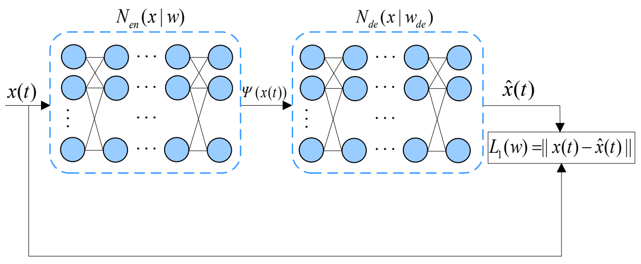


Fig. 3. Eigenfunction reconstruction capacity training network.

controllable, the loss function is defined as

$$L_3(A^*, B^*) = N - rank(R_c), \tag{14}$$

where

$$R_c = \begin{bmatrix} B^* & A^*B^* & (A^*)^2B^* & \cdots & (A^*)^{N-1}B^* \end{bmatrix}.$$

Based on the above three loss functions, the total loss function is defined as

$$L(w, A^*, B^*) = \sum_{i=1}^{3} w_i L_i, \tag{15}$$

where w_i denotes the weights of three loss functions. The Koopman networks are trained as described above until $L < L_{min}^{Koop}$ is a smaller positive number. Then the data-driven Deep-EDMD model (11) can be obtained by generating the training data based on the discretized model of the nonlinear system.

4 2D ILC-DRL control scheme

Introduce model mismatch matrices in the high-dimensional state-space model Eq. (11) obtained by the above Deep-EDMD method

$$\begin{cases} \chi_k(t+1) = (A^* + \Delta A_{t,k})\chi_k(t) + (B^* + \Delta B_{t,k})u_k(t) + \mathcal{H}(t,k) \\ y_k(t+1) = C^*\chi_k(t+1) \\ \chi_k(t) = \chi_0 \end{cases}, \tag{16}$$

where

$$\mathcal{H}(t,k) = \begin{bmatrix} d(t,k) & 0 & \cdots & 0 \end{bmatrix}^T, \mathcal{H}(t,k) \in \mathbb{R}^{(n+N)},$$

$\Delta A_{t,k} = E_1 \Delta^A_{t,k} F_1$, $\Delta B_{t,k} = E_2 \Delta^B_{t,k} F_2$ both denote model mismatch matrices related to time and batch simultaneously, E_1, E_2, F_1, F_2 denote constant coefficient matrices related to the upper bound of model mismatch, respectively. $\Delta^A_{t,k}$, $\Delta^B_{t,k}$ are unknown time and batch varying disturbance matrices with dimensions matching the system matrices. χ_0 denotes the initial state in the low-dimensional space x_0 after the upscaling of the eigenfunctions with the initial vector of each batch under the high-dimensional linear state space combined with itself. Since the state-space model obtained by the Deep-EDMD is more accurate and the consideration of the model mismatch matrix in the 2D ILC analysis tends to lead to unsolved linear matrix inequalities (LMI), one can utilize the 2D system theory to analyze the nominal system model and obtain the LMI for the convergence of the system.

The 2D deep reinforcement learning compensation ILC control block diagram is shown in Fig. 4. As can be seen from the figure, the control quantity

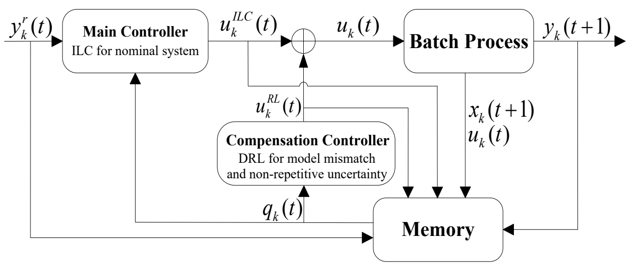


Fig. 4. 2D ILC-DRL control block diagram.

applied to the controlled object consists of two parts

$$u_k(t) = u_k^{ILC}(t) + u_k^{RL}(t),$$

where $u_k^{RL}(t)$ is the control vector generated by deep reinforcement learning, which compensates for the ILC control inputs and reduces the effects caused by non-repetitive uncertainty. $u_k^{ILC}(t)$ is the control vector generated by the ILC controller, which ensures the basic control performance of the batch run. The control vector between batch run can be designed as

$$u_k^{ILC}(t) = u_{k-1}(t) + r_k(t), \tag{17}$$

while can be utilized to overcome the deterioration of the control effect caused by the random exploration of DRL.

4.1 2D ILC based on Repetitive Process Model

Based on the linear state space model established by utilizing the above Deep-EDMD, a PD-type 2D system control law [36] of the 2D system below is adopted based on the state error,

$$\begin{aligned} r_k(t) =& K_1(\chi_k(t) - \chi_{k-1}(t)) + K_2 e_{k-1}(t) \\ &+ K_3(e_{k-1}(t+1) - e_{k-1}(t)), \end{aligned} \tag{18}$$

where $e_k(t) = y_r(t) - y_k(t)$ is the system output error, $y_r(t)$ is the desired system output trajectory, K_1, K_2, K_3 are the appropriate dimensional gain matrices to be solved.

Concerning the 2D system description in (16), the nominal system is extracted as

$$\begin{cases} \chi_k(t+1) = A^* \chi_k(t) + B^* u_k(t) \\ y_k(t+1) = C^* \chi_k(t+1) \end{cases} \tag{19}$$

Let $\bar{\chi}_k(t+1)$ be the state error vector of the neighboring batches of the system, i.e

$$\bar{\chi}_k(t+1) = \chi_k(t) - \chi_{k-1}(t) \tag{20}$$

Substituting the PD-type control law (18) into (17) yields

$$\begin{aligned} u_k(t) =& u_{k-1}(t) + K_1 \bar{\chi}_k(t+1) + K_2 e_{k-1}(t) \\ &+ K_3(e_{k-1}(t+1) - e_{k-1}(t)). \end{aligned} \tag{21}$$

Substituting (19) and (21) into (20) yields

$$\begin{aligned} \bar{\chi}_k(t+1) =& A^* \chi_k(t-1) + B^* u_k(t-1) - (A^* \chi_{k-1}(t-1) \\ &+ B^* u_{k-1}(t-1) \\ =& A^*(\chi_k(t-1) - \chi_{k-1}(t-1)) + B^*(u_k(t-1) - u_{k-1}(t-1)) \\ =& A^* \bar{\chi}_k(t) + B^*(u_k(t-1) - u_{k-1}(t-1)) \\ =& (A^* + B^* K_1)\bar{\chi}_k(t) + B^* K e_{k-1}(t-1) + B^* K_3 e_{k-1}(t), \end{aligned}$$

where $K = K_2 - K_3$. The difference between the output error at the same moment in time for the front and back batches of the system is

$$e_k(t) - e_{k-1}(t) = y_r(t) - y_k(t) - y_r(t) + y_{k-1}(t), \tag{22}$$

substituting (19), (20), (21) into the above equation yields

$$
\begin{aligned}
e_k(t) =& y_{k-1}(t) - y_k(t) + e_{k-1}(t) \\
=& - C^*(\chi_k(t) - \chi_{k-1}(t)) + e_{k-1}(t) \\
=& - C^*\bar{\chi}_k(t+1) + e_{k-1}(t) \\
=& - C^*(A^* + B^*K_1)\bar{\chi}_k(t) - C^*B^*Ke_{k-1}(t-1) \\
& + (I - C^*B^*K_3)e_{k-1}(t),
\end{aligned} \tag{23}
$$

Therefore, the repetitive process model of the nominal system (19) can be obtained as shown below

$$
\begin{cases}
\zeta_k(t+1) = \hat{A}\zeta_k(t) + \hat{B}e_{k-1}(t) \\
e_k(t+1) = \hat{C}\zeta_k(t+1) + \hat{D}e_{k-1}(t+1),
\end{cases} \tag{24}
$$

where

$$
\zeta_k(t) = \begin{bmatrix} \bar{\chi}_k(t) & e_k^T(t-1) \end{bmatrix}^T, \tag{25}
$$

$$
\hat{A} = \begin{bmatrix} A^* + B^*K_1 & B^*K \\ 0 & 0 \end{bmatrix}, \hat{B} = \begin{bmatrix} B^*K_3 \\ I \end{bmatrix}, \tag{26}
$$

$$
\hat{C} = \begin{bmatrix} -C^*(A^* + B^*K_1) & -C^*B^*K \end{bmatrix}, \hat{D} = \begin{bmatrix} I - C^*B^*K_3 \end{bmatrix}. \tag{27}
$$

4.2 LMI analysis of 2D linear repetitive process model

To derive the output error convergence condition for the system under the control law (21), the following theorem is presented based on Lemma 1,

Lemma 1 *([37]) Given matrices Υ_1, Υ_2, Υ_3 of appropriate dimensions, and $\Upsilon_2 \succ 0$, then $\Upsilon_1 + \Upsilon_3^T \Upsilon_2^{-1} \Upsilon_3 \prec 0$ is equivalent to the following two inequalities*

$$
(1) \begin{bmatrix} \Upsilon_1 & \Upsilon_3^T \\ \Upsilon_3 & -\Upsilon_2 \end{bmatrix} \prec 0,
$$

$$
(2) \begin{bmatrix} -\Upsilon_2 & \Upsilon_3 \\ \Upsilon_3^T & \Upsilon_1 \end{bmatrix} \prec 0.
$$

Theorem 1 *The repetitive system in (24) under the ILC law (22) is asymptotically stable as $k \to \infty$, if there exist positive definite matrices $W_1 \succ 0, W_2 \succ 0, W_3 \succ 0$ and matrices R_1, R_2, R_3 such that the following linear matrix inequality holds*

$$
\begin{bmatrix}
-W_1 & 0 & 0 & A^*W_1 + B^*R_1 & B^*R_2 & B^*R_3 \\
(*) & -W_2 & 0 & 0 & 0 & W_3 \\
(*) & (*) & -W_3 & -C^*(A^*W_1 + B^*R_1) & -C^*B^*R_2 & W_3 - C^*B^*R_3 \\
(*) & (*) & (*) & -W_1 & 0 & 0 \\
(*) & (*) & (*) & (*) & -W_2 & 0 \\
(*) & (*) & (*) & (*) & (*) & -W_3
\end{bmatrix} \prec 0,
$$

and the gain matrix of the control law (21) *is*

$$K_1 = R_1 W_1^{-1}, K_2 = R_2 W_2^{-1} + R_3 W_3^{-1}, K_3 = R_3 W_3^{-1}. \tag{28}$$

Proof. Based on the nature of the two-dimensional system of iterative learning control, a Lyapunov function of the following form is constructed,

$$V(k,t) = V_1(t,k) + V_2(k,t),$$

and

$$V_1(t,k) = \zeta_{k+1}^T Z \zeta_{k+1}, V_2(k,t) = e_k^T(t) Z_3 e_k(t),$$

where $Z = diag\{Z_1, Z_2\}$, $V_1(t,k)$ represents the energy change of the system in the time direction during an iteration, and $V_2(k,t)$ represents the energy change in the batch direction. Substituting (25) into the above equation, the energy increment of the system in both directions can be written as

$$\begin{aligned}
\Delta V_1(t,k) =& \zeta_{k+1}^T(t+1) Z \zeta_{k+1}(t+1) - \zeta_{k+1}^T(t) Z \zeta_{k+1}(t), \\
=& (\hat{A}\zeta_{k+1}(t) + \hat{B}e_k(t))^T Z (\hat{A}\zeta_{k+1}(t) + \hat{B}e_k(t)) \\
& - \zeta_{k+1}^T(t) Z \zeta_{k+1}(t) \\
\Delta V_2(k,t) =& e_{k+1}^T(t) Z_3 e_{k+1}(t) - e_k^T(t) Z_3 e_k(t) \\
=& (\hat{C}\zeta_{k+1}(t) + \hat{D}e_k(t))^T Z_3 (\hat{C}\zeta_{k+1}(t) + \hat{D}e_k(t)) \\
& - e_k^T(t) Z_3 e_k(t).
\end{aligned}$$

Then the total energy increment yields

$$\begin{aligned}
\Delta V(k,t) =& \Delta V_1(t,k) + \Delta V_2(k,t) \\
=& H_{k+1}^T(t)(\phi^T P \phi - P) H_{k+1}(t),
\end{aligned} \tag{29}$$

where

$$H_{k+1}^T(t) = \begin{bmatrix} \zeta_{k+1}(t) \\ e_k(t) \end{bmatrix}, \phi = \begin{bmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{bmatrix}, P = \begin{bmatrix} Z & 0 \\ (*) & Z_3 \end{bmatrix}, \tag{30}$$

$H_{k+1} \neq 0$, Hence a sufficient condition for the convergence of the system is

$$\Delta V(k,t) < 0.$$

i.e.

$$\phi^T P \phi \prec 0.$$

According to the second equation in Lemma 1, the above equation is equivalent to

$$\begin{bmatrix} -P^{-1} & \phi \\ \phi^T & -P \end{bmatrix} \prec 0,$$

substituting into equations (26), (27), (30) gives

$$\begin{bmatrix} -Z_1^{-1} & 0 & 0 & A^*+B^*K_1 & B^*K & B^*K_3 \\ (*) & -Z_2^{-1} & 0 & 0 & 0 & I \\ (*) & (*) & -Z_3^{-1} & -C^*(A^*+BK_1) & -C^*B^*K & I-C^*B^*K_3 \\ (*) & (*) & (*) & -Z_1 & 0 & 0 \\ (*) & (*) & (*) & (*) & -Z_2 & 0 \\ (*) & (*) & (*) & (*) & (*) & -Z_3 \end{bmatrix} \prec 0.$$

By post-multiplying both the left and right sides of the above matrix inequality by $diag\{I, I, I, Z_1^{-1}, Z_2^{-1}, Z_3^{-1}\}$ and

$$Z_1^{-1} = W_1, Z_2^{-1} = W_2, Z_3^{-1} = W_3,$$

$$K_1Z_1^{-1} = K_1W_1 = R_1, K_2Z_2^{-1} = KW_2 = R_2, K_3Z_3^{-1} = K_3W_3 = R_3,$$

the condition in Theorem 1 follows immediately. The proof is complete.

4.3 2D DRL compensator

Since the dimension of the lifted space cannot be constructed to be infinite, the linear state space model obtained through Koopman operator suffers from deviations and uncertain perturbations over time and batches, which may degrade the control performance. Additionally, to approximate the original nonlinear model as accurately as possible, the resulting linear state space model could be complex and highly dimensional, imposing difficulty on the control design and implementation. To circumvent this issue, a 2D deep reinforcement learning algorithm is introduced to compensate for the model mismatch, non-repetitive uncertainties, and high dimensionality.

4.3.1 Soft Actor-Critic algorithm

Reinforcement learning optimizes the feedback from the environment by training the agent to make sequential decisions in the environment, and there are two key elements in the training process: agent and environment. Fig. 5 shows the interaction process between the SAC agent and the 2D ILC environment, i.e., the Markov decision process. In the process of reinforcement learning, agent starts from a set initial state and acts on environment according to its own strategy. Environment will change the state of agent and provide feedback rewards, according to which agent evaluates the value of the strategy, then changes the strategy in the direction of greater rewards. In this way, agent continuously optimizes its strategy and will get the best control compensation signal finally. in which several key parameters are shown below:

· State space: $s_t \in S$, s_t represents the state of the agent in the environment, S represents the set of states.

· Action space: $a_t \in A$, a_t denotes the force of an agent on its environment, A represents the set of actions.

· Reward: $r(s_t, a_t) \in R$, $r(s_t, a_t)$ denotes the feedback gain that the agent receives from the environment, R represents the set of reward.

· State transfer probability: $p(s_{t+1}|(s_t, a_t))$ represents the probability that the agent will make an action a_t in state s_t to transfer to state s_{t+1}.

· Discount factor: γ indicates the relative importance of awards under each batch in terms of length of time.

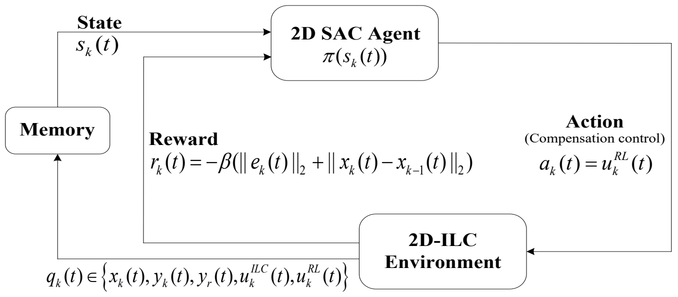

State
$s_k(t)$
2D SAC Agent
$\pi(s_k(t))$
Memory
Reward
$r_k(t) = -\beta(\| e_k(t) \|_2 + \| x_k(t) - x_{k-1}(t) \|_2)$
Action
(Compensation control)
$a_k(t) = u_k^{RL}(t)$
2D-ILC
Environment
$q_k(t) \in \{x_k(t), y_k(t), y_r(t), u_k^{ILC}(t), u_k^{RL}(t)\}$

Fig. 5. Illustration of agent-environment interaction process.

In order to characterize the cumulative benefit of an agent's strategy over a batch to help the agent estimate the long-term value of taking an action in a particular state, a state-action value function is defined for the strategy

$$Q(s_t, a_t) = E_{s_t \sim \mathcal{D}, a_t \sim \pi} \left[\sum_{t=0}^{T} \gamma^t r_t \right], \tag{31}$$

where $E_{(s_t \sim \mathcal{D}, a_t \sim \pi)}$ denotes the expected value of the cumulative reward under strategy π and distribution of the sampled states by $\mathcal{D}$. The purpose of reinforcement learning is to find the optimal policy by maximizing the state-action value function

$$\begin{aligned} \pi_*(s_t) &= \underset{\pi}{argmax}\, Q(s_t, a_t) \\ &= \underset{\pi}{argmax}\, E_{s_t \sim \mathcal{D}, a_t \sim \pi} \left[\sum_{t=0}^{T} \gamma^t r_t \right]. \end{aligned} \tag{32}$$

Lemma 2 *([38] Optimal Bellman Equation). The optimal policy π_* and the optimal state-action value function $Q_*(s_t, a_t)$ satisfy the following equation*

$$Q_*(s_t, a_t) = E_{s_{t+1}\sim\mathcal{D}, a_{t+1}\sim\pi_*}\left[r_t + \gamma Q_*(s_{t+1}, a_{t+1})\right]. \tag{33}$$

Meanwhile, in order to solve the local optimization problem, the Soft Actor-Critic algorithm (SAC), an RL algorithm with higher sampling efficiency, higher robustness, and better exploratory ability, was developed by combining the Bellman equation with entropy [39]. Although the model obtained by Koopman operator could be highly accurate, it brings the problem of high dimensionality of the system. Therefore, simple DRL algorithms are difficult to search for the best control compensation strategy and have poor convergence with high sampling complexity. The SAC algorithm combines the method of policy gradient and the value function by using the double Actor and the Critic network. At the same time, the principle of maximum entropy is introduced, which effectively balances the exploration and utilization of strategies and avoids the strategies falling into the local optimal point. In general, SAC can effectively solve the problems of high system matrix complexity and non-repetitive uncertainty.

The key of the SAC algorithm is to not only maximize the long-term return of the system but also keep the strategy entropy as large as possible. The Bellman equation shown in (32) is updated as

$$\pi_*(s_t) = \underset{\pi}{argmax}\, E_{s_t\sim\mathcal{D}, a_t\sim\pi}\left[\sum_{t=0}^{T}\gamma^t(r_t + \alpha\mathcal{H}(\pi(\cdot|s_t)))\right], \tag{34}$$

where $\mathcal{H}(\pi(\cdot|s_t))$ denotes the strategy entropy, and α is the relative coefficient of the strategy entropy, which indicates the importance of agent exploration relative to long-term reward. After the introduction of strategy entropy, the state-action value function of (33) is updated as

$$Q_*(s_t, a_t) = E_{s_t\sim\mathcal{D}, a_t\sim\pi}\left[r_t + \gamma(Q_*(s_{t+1}, a_{t+1}) - \alpha\mathcal{H}(\pi(\cdot|s_t)))\right] \tag{35}$$

SAC blends the ideas of DQN and Actor-Critic, adds entropy to encourage exploration, with four Q-value networks as well as two evaluation networks. Among them, the Q networks include two evaluation networks and the corresponding two target networks. The purpose of setting up two networks is to prevent the problem of overestimation of Q values. According to Lemma 2, the strategy evaluation network is trained as follow:

$$J_Q(\theta) = E_{(s_t, a_t, s_{t+1})\sim\mathcal{D}}\left[\frac{1}{2}(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t, s_{t+1}))\right], \tag{36}$$

where

$$\hat{Q}(s_t, a_t, s_{t+1}) = r(s_t, a_t) + \gamma E_{a_{t+1} \sim \pi} \left[Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - \alpha \mathcal{H}(\pi_\epsilon(\cdot|s_t)) \right], \quad (37)$$

where $Q_{\bar{\theta}}(s_{t+1}, a_{t+1})$ is the result obtained for the target network. θ_1 and θ_2 are the parameters of the two state action value assessment networks, $\bar{\theta_1}$ and $\bar{\theta_2}$ are the parameters of the state action value target network. In order to make the convergence process of policy π converging to the optimal policy π_* more stable [13], a target policy network $\pi_{\hat{\epsilon}}$ is added, then (37) is updated as [40]

$$\hat{Q}(s_t, a_t, s_{t+1}) = r(s_t, a_t) + \gamma E_{a_{t+1} \sim \pi} \left[Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - \alpha \mathcal{H}(\pi_{\bar{\epsilon}}(\cdot|s_t)) \right] \quad (38)$$

The parameters of the state-action value target network and policy target network will update the parameters by the following equation:

$$\hat{\theta_1} = \tau\theta_1 + (1 - \tau)\hat{\theta_1}$$

$$\hat{\theta_2} = \tau\theta_2 + (1 - \tau)\hat{\theta_2}$$

$$\hat{\epsilon_1} = \tau\epsilon_1 + (1 - \tau)\hat{\epsilon_1}$$

where τ is the weight of the evaluation network parameters.

After optimizing the state action value network, the strategy network is then optimized to maximize the cumulative reward and entropy, and the loss function of the strategy network is to minimize the following equation:

$$J_\pi(\epsilon) = E_{(s_t \sim \mathcal{D}, a_t \sim \pi_\epsilon} \left[\alpha \log \pi_\epsilon(a_t|s_t) - Q_\theta(s_t, a_t) \right] \quad (39)$$

where ϵ denotes the network parameters of the strategy network $\pi_\epsilon(a_t|s_t)$, the input to the network is state, and the outputs are normally distributed parameters μ and σ, i.e.

$$\pi(s_t) = \pi_\epsilon(s_t) = \mathcal{N}_\epsilon(\mu, \sigma^2|s_t), \quad (40)$$

and the actions will be sampled in a sequential distribution to obtain the dataset $\mathcal{D}$.

The purpose of (39) is to optimize the strategy towards a larger entropy and state-action value function, $-\log \pi_\epsilon(a_t|s_t)$ is the specific form of entropy $\mathcal{H}$. The entropy temperature factor α has two forms, one is a fixed value during training, and the other is changed during training to adjust the entropy constraints on the objective. In order to adjust the explored weights according to the actual situation during the training process and find the best control compensation signal more quickly and effectively, this paper adopts the second method. The entropy temperature factor is automatically adjusted by the following equation:

$$J_\alpha(\alpha) = E_{(s_t \sim \mathcal{D}, a_t) \sim \pi_\epsilon} \left[-\alpha \log \pi_\epsilon(a_t|s_t) - \alpha \bar{H} \right] \tag{41}$$

where $\bar{H}$ is the minimum entropy constant of the target, which makes it possible to put more effort into exploration at the beginning of the intelligent body and less exploration when the optimal action is gradually determined.

Lemma 3 *([41] Soft Policy Iteration). Repeated application of soft policy evaluation and soft policy improvement from any* $\pi \in \Pi$ *converges to a policy* π^* *such that* $Q_*(s_t, a_t) \geq Q(s_t, a_t)$ *for all* $(s_t, a_t) \in S \times A$*, assuming* $|A| < \infty$*,* Π *is the set of all probable policies.*

According to Lemma 3, the state-action value Q, which is constantly updated according to (38), will eventually guide the strategy network to find the optimal control compensation signal. The agent is constantly trained in the 2D ILC environment, constantly explores and adjusts the control compensation signal through the historical batch information. The agent will gradually adapt to the 2D ILC and obtain the optimal control compensation signal.

4.3.2 SAC compensator based on 2D ILC

In order to introduce the agent into the 2D ILC as a control compensator, we need to design the state $s_k(t)$, reward $r_k(t)$, action $a_k(t)$ of agent. Regarding the model mismatch and non-repetitive uncertainties associated with time and batch run, it is necessary to consider the information of the last batch of the system and the prior to the current moment of the current batch as the state of the agent [42], so the state can be defined as

$$s_k(t) = \left[\cdots, q_{k-1}(t-1), q_{k-1}(t), q_{k-1}(t+1), \cdots, q_k(t-1), q_k(t) \right], \tag{42}$$

with

$$q_k(t) \in \left\{ x_k(t), y_k(t), y_r(t), u_k^{ILC}(t), u_k^{RL}(t) \right\}.$$

The policy network of SAC generates actions that are sampled in accordance with the mean and variance of a normal distribution, and therefore, this action can be devised as a control compensation signal of a two-dimensional iterative learning controller

$$a_k(t) = \pi_\epsilon(s_t) = u_k^{RL}(t).$$

Considering that the control aim is to attain zero-error tracking of the desired trajectory and quickly stabilize the system state, the reward of the agent's action can be devised as follow

$$r_k(t) = -\beta(||e_k(t)||_2 + ||x_k(t) - x_{k-1}(t)||_2),$$

where $||\cdot||_2$ denotes L2 Norm. To evaluate the performance of the agent in a 2D ILC environment, define the average reward for the agent as

$$r_k^{av} = \frac{1}{T}\sum_{i=1}^{T} r_k(t).$$

At the same time, the root-mean-square error(RMSE) between the expected output and the actual output under each batch is an important indicator that evaluates the control system's performance

$$I_k = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(y_k(t) - y_r(t))^2} \tag{43}$$

For clarity, the 2D ILC method of DRL compensator based on Koopman operator theory is summarized as Algorithm 1. The algorithm is divided into three steps, the first step uses the Koopman operator network of Section Fig. 2 to lift the nonlinear batch process into a high-dimensional linear space, which consists of two phases. In the first phase, from the initial state, the control volume is sampled randomly, and the time series data of K batches are obtained based on the numerical model and stored in the Replay Buffer. In the second phase, K_1 batches of data are randomly sampled in the Replay Buffer to train the Koopman operator network until the number of training rounds $i > N_1$ or the loss value $L < L_{min}^{Koop}$. To update the training data in real time, the Replay Buffer is re-updated at regular intervals. In the second step, obtain the ILC control law for convergence of the high-dimensional linear model based on Theorem 1. In the third step, train SAC agent to compensate for non-repetitive uncertainty and model mismatch. During the actual training process of SAC, the agent's strong exploratory ability described by $\mathcal{H}$ in the initial training stage may generate inappropriate control compensation signals, leading to poor system performance. Additionally, online training is costly and time-consuming. To address these issues, the training is conducted offline and divided into two stages. In the first stage, training is performed on the linearized state space model with model mismatch until $r_k^{av} \leq \varsigma_1$ or the number of training iterations exceeds a specified value N_2. In the second stage, training is conducted in the actual nonlinear system with non-repetitive perturbation, allowing the agent to gradually compensate for uncertainties and time-varying disturbances until $I_k \leq \varsigma_2$ or the number of training iterations exceeds the specified value N_3.

5 Numerical simulations

In order to verify the performance of the above algorithms on nonlinear batch processes, the continuously stirred tank reactor (CSTR) with non-repetitive uncertainty is chosen as the simulation control object in [43]:

$$\begin{cases} \dot{x}_1 = -(\frac{F}{V} + k_0 e^{\frac{-E}{Rx_2}})x_1 + \frac{F}{V}C_{A_0} + d_{k,1}(t) \\ \dot{x}_2 = -\frac{\triangle H k_0}{\rho C_p} e^{\frac{-E}{Rx_2}} x_1 - \frac{F}{V}T_{A_0} + \frac{u}{\rho C_p V} + d_{k,2}(t) \\ y = x_2 \end{cases}, \tag{44}$$

where

$$d_{k,1}(t) = 0, d_{k,2}(t) = 0.5(n_1 sin(2.5t\pi) + n_2 sin(0.1k\pi))$$

is a perturbation related to time and batch, n_1, n_2 are random noises obeying a Gaussian distribution with mean 0 and variance 1. x_1 is the product concentration in the reactor ($kmol/m^3$) and x_2 is the reactor temperature (K), which is also the control target. The initial state of system is $x_0 = [0.5, 310]^T$.The running time per batch is fixed to be 2 min with a sampling time of 0.01 and $T = 200$. The control desired output trajectory is as follow:

$$y_r(t) = \begin{cases} 370, 0 \leqslant t \leqslant 100 \\ 350, 0 < t \leqslant 200 \end{cases}.$$

Substituting the parameters in [43] to (44) and discretizing the process description, the state space model shown below can be obtained through the Koopman operator shown in Section 3:

$$A^* = \begin{bmatrix} A^*_{1,1} & A^*_{1,2} \\ A^*_{2,1} & A^*_{2,2} \end{bmatrix}, B^* = \begin{bmatrix} B^*_{1,1} & B^*_{1,2} \end{bmatrix}^T,$$

$$C^* = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where

$$A^*_{1,1} = \begin{bmatrix} 7.69 \times 10^{-1} & 6.02 \times 10^{-5} & -8.36 \times 10^{-4} & -6.22 \times 10^{-3} \\ 2.44 \times 10^{-1} & 9.90 \times 10^{-1} & -3.78 \times 10^{-3} & -4.88 \times 10^{-2} \\ -4.55 \times 10^{-2} & 1.03 \times 10^{-5} & 1.00 & 9.46 \times 10^{-3} \\ -2.15 \times 10^{-2} & 1.31 \times 10^{-3} & -7.90 \times 10^{-5} & 3.39 \times 10^{-4} \end{bmatrix},$$

Algorithm 1 2D ILC-RL Control for Nonlinear Batch Process

```
 1: Transform the system to linear system in lifted space:
 2: Determine the initial state, lifted space dimension, loss function, networks according to Section 3.
 3: Phase 1. Obtain replay buffer:
 4: for $i \leq L_{buffer}$(length of replay buffer) do
 5:     for $j \leq T$ do
 6:         Sampling control vectors.
 7:         Starting from the initial state, obtain the next state according to the nonlinear system.
 8:     end for
 9: end for
10: Phase 2. Train the Koopman operator network:
11: for $i \leq N_1$ or $L > L_{min}^{Koop}$ do
12:     for $j \leq$ max sampling batches do
13:         Train the network according to Eq. (15).
14:         Update the data in the replay buffer.
15:     end for
16: end for
17: Obtain the ILC control law of the nominal system:
18: Design the 2D iterative learning controller based on the scheme in Section 4.2.
19: Initialize all relevant parameters for the 2D DRL compensator.
20: Train the agent to compensate the 2D ILC:
21: Phase 1. Train on the state space system (offline):
22: for maximal training batches $N_2$ do or the average reward per batch $r_{av} > \varsigma_1$
23:     Sample the action:
            $$u_k(t) = u_k^{ILC}(t) + \mathcal{N}_\epsilon(\mu, \sigma^2|s_t)$$
24:     Interaction with the virtual environment.
25:     Optimization of the 2D DRL compensator based on the SAC algorithm in Section 4.3.1.
26: end for
27: Phase 2. Train on the actual system (offline):
28: for maximal training batches $N_3$ do or $I_k < \varsigma_2$
29:     Sample the action:
            $$u_k(t) = u_k^{ILC}(t) + \mathcal{N}_\epsilon(\mu, \sigma^2|s_t)$$
30:     Interaction with the practical batch system.
31:     Optimization of the 2D DRL compensator based on the SAC algorithm in Section 4.3.1.
32: end for
33: Apply to the actual system (online):
34: Finally, the trained 2D ILC-RL controller serves as the ultimate controller.
```

$$A_{1,2}^* = \begin{bmatrix} -1.95\times10^{-3} & -2.70\times10^{-3} & -1.60\times10^{-2} & 2.39\times10^{-5} \\ 3.22\times10^{-3} & 2.34\times10^{-3} & 1.49\times10^{-2} & -1.71\times10^{-2} \\ 1.33\times10^{-2} & -1.99\times10^{-2} & 3.62\times10^{-2} & 2.19\times10^{-3} \\ -3.61\times10^{-6} & -1.86\times10^{-5} & 8.01\times10^{-05} & 1.69\times10^{-5} \end{bmatrix},$$

$$A_{2,1}^* = \begin{bmatrix} -1.10\times10^{-1} & 2.62\times10^{-5} & 5.98\times10^{-3} & -3.20\times10^{-3} \\ -1.36\times10^{-1} & 3.36\times10^{-5} & -2.84\times10^{-3} & 1.52\times10^{-2} \\ 6.43\times10^{-2} & -1.97\times10^{-5} & 2.04\times10^{-3} & -1.95\times10^{-1} \\ 4.56\times10^{-2} & -1.08\times10^{-5} & 8.10\times10^{-3} & 9.27\times10^{-3} \end{bmatrix},$$

$$A_{2,2}^* = \begin{bmatrix} 1.01 & -1.39\times10^{-02} & -8.64\times10^{-3} & -1.91\times10^{-3} \\ -1.24\times10^{-4} & 1.00 & 6.04\times10^{-2} & 4.04\times10^{-3} \\ -7.24\times10^{-3} & -1.03\times10^{-3} & 2.80\times10^{-1} & -2.30\times10^{-3} \\ 7.85\times10^{-4} & 1.26\times10^{-2} & 3.21\times10^{-2} & 1.01 \end{bmatrix},$$

$$B_{1,1}^* = \begin{bmatrix} 5.70\times10^{-6} & 4.19\times10^{-3} & -1.62\times10^{-7} & -1.16\times10^{-2} \end{bmatrix},$$

$$B_{1,2}^* = \begin{bmatrix} 2.11\times10^{-7} & 1.57\times10^{-7} & 2.25\times10^{-6} & 2.91\times10^{-6} \end{bmatrix},$$

Starting from the initial state, the control inputs are randomly sampled within $[-10^4, 10^4]$ to obtain four groups of the outputs of the state space model and the nonlinear system as Fig. 6, the outputs of four groups are close to each other, indicating that the Koopman operator network shown in Fig. 2 has a high accuracy with the nonlinear system. Meanwhile, due to the wide sampling range of the control vectors, it means that an globally accurate approximation of the original nonlinear system can be realized.

Designing the same P-type ILC controller as in [13], the RMSE under the Koopman operator is obtained and compared with the equilibrium point linearization, as in Fig. 7. Although both converge well to near zero, the 2D ILC controller obtained based on the Koopman operator modeling converges faster to the equilibrium point. This is attributable to the fact that the high-dimensional model derived from Fig. 2 exhibits a high level of accuracy even on unsteady state points.

Next, Designing the ILC controller according to Theorm 1 shown in 4.2 yields the ILC controller gains for (21):

$$K_1 = \begin{bmatrix} K_{1,1} & K_{1,2} \end{bmatrix}, \quad K_2 = 1.08, \quad K_3 = 1.08,$$
$$K_{1,1} = \begin{bmatrix} -1.83\times10^{-3} & -1.93\times10^{2} & -1.74\times10^{4} & 2.07 \end{bmatrix},$$
$$K_{1,2} = \begin{bmatrix} 5.47\times10^{3} & -6.31\times10^{3} & -2.58\times10^{3} & -2.64\times10^{4} \end{bmatrix}.$$

Concerning non-repetitive uncertainty, the model mismatch matrices are assumed to be E_1 and E_2 is 10% of the nominal system matrices, F_1 and F_2

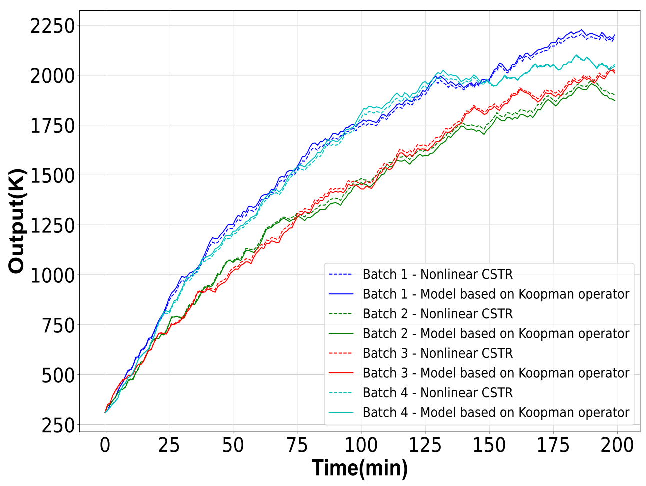


Fig. 6. Comparison between the outputs of the nonlinear CSTR and linearized state space model.

are unit matrix, and the perturbation terms are

$$\Delta_{t,k}^{A} = \Delta_{t,k}^{B} = 0.5sin(0.2k\pi) + 0.5sin(t).$$

Then, designing the state, action, and reward for the agent according to Section 4.3.2:

$$s_k(t) = \begin{bmatrix} u_k^{RL}(t-1) & u_k^{ILC}(t-1) & y_k(t-1) & y_k^r(t-1) & y_k^r(t) \\ u_{k-1}^{RL}(t) & u_{k-1}^{ILC}(t) & y_{k-1}(t) & y_{k-1}^r(t) & u_k^{ILC}(t) \\ u_{k-1}^{RL}(t-1) & u_{k-1}^{ILC}(t-1) & y_{k-1}(t-1) & y_{k-1}^r(t-1) & \end{bmatrix},$$

$$r_k(t) = -2(||e_k(t)||_2 + ||x_k(t) - x_{k-1}(t)||_2), a_k(t) = u_k^{RL}(t).$$

Finally, the agent in the 2D ILC environment is trained in stages according to Algorithm 1. The trained DRL agent is combined with 2D ILC controller and applied to the nonlinear system (44), the simulation results are obtained as shown in Fig. 8–Fig. 10 in comparison with the P-type and PD-type dynamic iterative linearization ILC schemes [14].

Fig. 8 shows that the control block diagram Fig. 4 can basically realize the accurate compensation of 2D ILC for non-repetitive uncertainty with Gaussian noise, so that the actual output of the system can realize almost zero steady-state tracking error, except for small error fluctuations at the initial moments of each batch and at the time of the sudden change of the desired output.Fig. 9 indicates that the control performance of the P-type, PD-type dynamic iterative linearization , classic ILC and the proposed control scheme. It can be seen that when non-repetitive uncertainty perturbations are added,

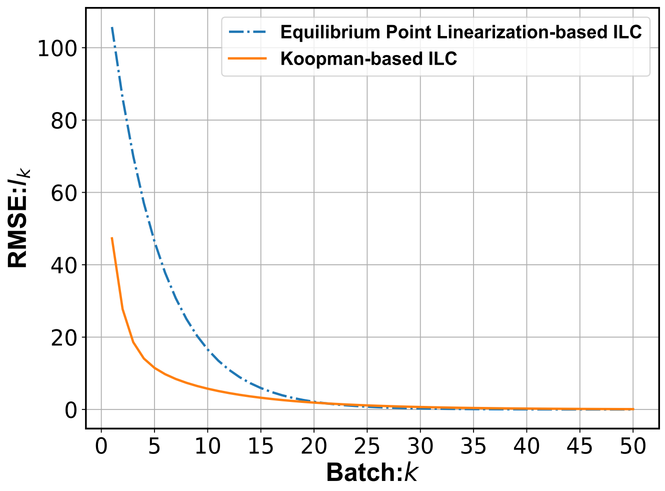


Fig. 7.The control performance without uncertainty based on Koopman operator and equilibrium point linearization.

the classic ILC and dynamic iterative linearized ILC provokes oscillation continuously with regard to the perturbations, whereas the compensated 2D-ILC scheme via Koopman operator contributes to no more fluctuation trend. Fig. 10 shows that as the number of training batches increases, the agent gradually adapts to the 2D ILC environment and generates positive control compensation signals, resulting in increasing average reward along the batch run. Nevertheless, in comparison with the methodology outlined in [42], the agent based on the linearisation of the deep Koopman operator in this paper demonstrates a superior convergence rate and is more straightforward to optimise for the desired control compensation signal.

6 Conclusions

For nonlinear batch processes subject to non-repetitive uncertainty, an RL based ILC control scheme has been proposed by using the Koopman operator for a linearized system description. The proposed control scheme uses a Koopman operator to realize a global linearization of the nonlinear system, which facilitates the controller design and can acquire a feasible solution for the sufficient condition in form of LMI. Meanwhile, DRL is introduced to compensate for the ILC control signal, and the affect of non-repetitive uncertainty can be eliminated by continuous training of the agent. An illustrative example of chemical CSTR batch operation well demonstrates that the proposed 2D ILC-RL scheme based on Koopman operator can realize

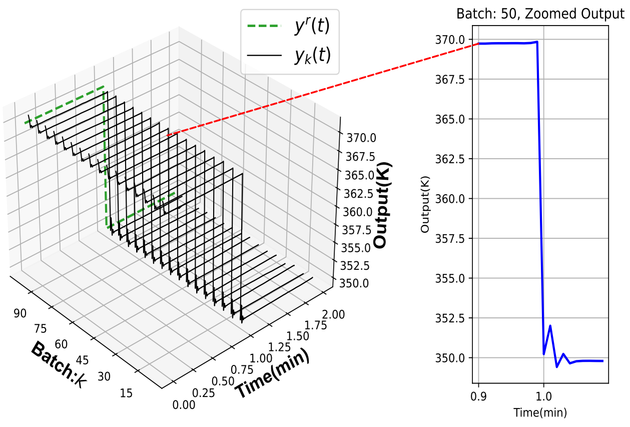


Fig. 8. The output response of the nonlinear CSTR.

significantly improved control performance for nonlinear batch process with non-repetitive uncertainty. Nevertheless, the proposed scheme still exhibits certain deficiencies. The considerable quantity of data necessitated for the offline training of Koopman operator networks and agent, in the form of thousands of batches, presents a significant challenge in terms of data collection in real industrial production settings. This article is based on a mathematical model that was constructed in advance for the purpose of obtaining data, which is, in essence, still model-based. Accordingly, the objective of future research in this field is to develop model-free intelligent control ILC using less data. Furthermore, in instances where the nonlinear model is intricate and the dimensionality of the high-dimensional linear model is considerable, alternative neural networks may be contemplated for substitution of the linear layer. These include Long Short-Term Memory (LSTM) networks, which are extensively utilized for model prediction and demonstrate efficacy in addressing the issue of gradient disappearance. This is also a prospective avenue of enquiry for future research.

CRediT authorship contribution statement

Hongfeng Tao: Conceptualization, Methodology, Supervision, Recourses, Writing - review & editing. Data curation, Software, Investigation, Visualization. **Yuan Huang:** Conceptualization, Data curation, Methodology, Writing - original - draft. **Tao Liu:** Supervision, Recourses, Writing - review & editing. **Wojciech Paszke:** Supervision, Recourses, Writing - review &

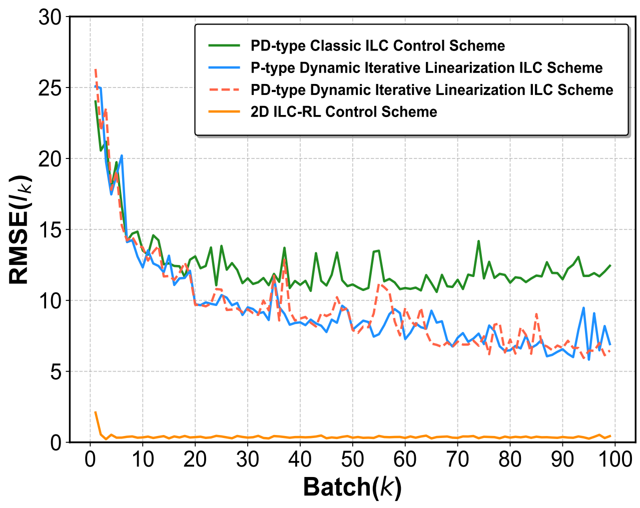


Fig. 9. The control performance comparison of P-type, PD-type dynamic iterative linearization, PD-type classic ILC and the proposed control scheme.

editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62361136585 and 62173058), the National Science Centre in Poland (2020/39/B/ST7/01487), and 111 Project (B23008).

References

[1] Jianyu Zhang and Kai Huang. Fault diagnosis of coal-mine-gas charging sensor networks using iterative learning-control algorithm. *Physical Communication*, 43:101175, 2020.

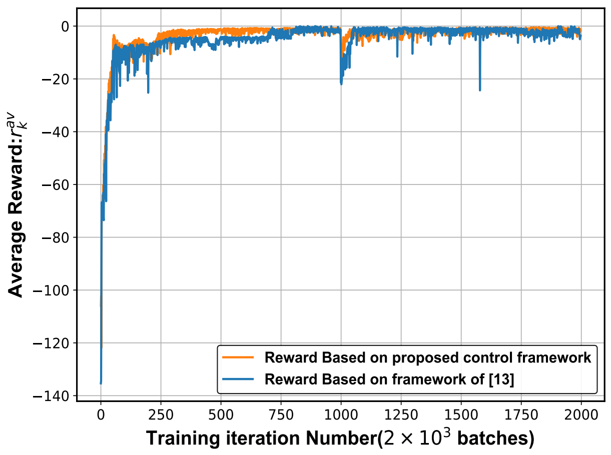


Fig. 10. The training reward with the 2D ILC CSTR environment.

[2] G Stadler, A Steinboeck, L Marko, A Deutschmann-Olek, and A Kugi. Iterative learning and feedback control for the curvature and contact force of a metal strip on a roll. *Control Engineering Practice*, 121:105071, 2022.

[3] Lin Yang, Yanan Li, Deqing Huang, Jingkang Xia, and Xiaodong Zhou. Spatial iterative learning control for robotic path learning. *IEEE Transactions on Cybernetics*, 52(7):5789–5798, 2022.

[4] Wenjie Wu, Lin Qiu, Xing Liu, Feng Guo, Jose Rodriguez, Jien Ma, and Youtong Fang. Data-driven iterative learning predictive control for power converters. *IEEE Transactions on Power Electronics*, 37(12):14028–14033, 2022.

[5] Chengyu Zhou, Li Jia, and Yang Zhou. Tube-based iterative-learning-model predictive control for batch processes using pre-clustered just-in-time learning methodology. *Chemical Engineering Science*, 259:117802, 2022.

[6] Luyuan Gao, Zhihe Zhuang, Hongfeng Tao, Yiyang Chen, and Vladimir Stojanovic. Non-lifted norm optimal iterative learning control for networked dynamical systems: A computationally efficient approach. *Journal of the Franklin Institute-Engineering and Applied Mathematics*, 361(15), OCT 2024.

[7] Hongfeng Tao, Junhao Zheng, Junyu Wei, Wojciech Paszke, Eric Rogers, and Vladimir Stojanovic. Repetitive process based indirect-type iterative learning control for batch processes with model uncertainty and input delay. *Journal of Process Control*, 132:103112, 2023.

[8] Hongfeng Tao, Longhui Zhou, Shoulin Hao, Wojciech Paszke, and Huizhong Yang. Output feedback based PD-type robust iterative learning control for uncertain spatially interconnected systems. *International Journal of Robust and Nonlinear Control*, 31(12):5962–5983, 2021.

[9] Tao Liu and Youqing Wang. A synthetic approach for robust constrained iterative learning control of piecewise affine batch processes. *Automatica*, 48(11):2762–2775, 2012.

[10] Li Jia, Chao Han, and Minsen Chiu. Dynamic R-parameter based integrated model predictive iterative learning control for batch processes. *Journal of Process Control*, 49:26–35, 2017.

[11] Xingzheng Wu, Ju H Park, and Mouquan Shen. Adaptive iterative learning control for continuous systems with non-repetitive uncertainties and unknown state delays. *International Journal of Robust and Nonlinear Control*, 33(4):2796–2810, 2023.

[12] Yiyang Chen, Wei Jiang, and Themistoklis Charalambous. Machine learning based iterative learning control for non-repetitive time-varying systems. *International Journal of Robust and Nonlinear Control*, 33(7):4098–4116, 2023.

[13] Jianan Liu, Zike Zhou, Wenjing Hong, and Jia Shi. Two-dimensional iterative learning control with deep reinforcement learning compensation for the non-repetitive uncertain batch processes. *Journal of Process Control*, 131:103106, 2023.

[14] Na Lin, Ronghu Chi, Biao Huang, and Zhongsheng Hou. Iterative dynamic linearization and identification of a nonlinear learning controller: A data-driven approach. *Journal of the Franklin Institute*, 356(13):7009–7027, 2019.

[15] Ronghu Chi, Yu Hui, Chiang-Ju Chien, Biao Huang, and Zhongsheng Hou. Convergence analysis of sampled-data ilc for locally lipschitz continuous nonlinear nonaffine systems with nonrepetitive uncertainties. *IEEE Transactions on Automatic Control*, 66(7):3347–3354, 2021.

[16] Ronghu Chi, Zhongsheng Hou, Shangtai Jin, and Biao Huang. Computationally efficient data-driven higher order optimal iterative learning control. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):5971–5980, 2018.

[17] Deyuan Meng and Jingyao Zhang. Robust optimization-based iterative learning control for nonlinear systems with nonrepetitive uncertainties. *IEEE/CAA Journal of Automatica Sinica*, 8(5):1001–1014, 2021.

[18] Jianmin Xing, Ronghu Chi, and Na Lin. Adaptive iterative learning control for 2D nonlinear systems with nonrepetitive uncertainties. *International Journal of Robust and Nonlinear Control*, 31(4):1168–1180, 2021.

[19] Zhengquan Chen, Yandong Hou, Ruirui Huang, and Qianshuai Cheng. Neural network compensator-based robust iterative learning control scheme for mobile robots nonlinear systems with disturbances and uncertain parameters. *Applied Mathematics and Computation*, 469:128549, 2024.

[20] Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern Koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

[21] Wenchong Tian, Zhiyu Zhang, Damien Bouffard, Hao Wu, Kunlun Xin, Xianyong Gu, and Zhenliang Liao. Enhancing interpretability and generalizability of deep learning-based emulator in three-dimensional lake hydrodynamics using Koopman operator and transfer learning: Demonstrated on the example of lake zurich. *Water Research*, 249:120996, 2024.

[22] Zuowei Ping, Zhun Yin, Xiuting Li, Yefeng Liu, and Tao Yang. Deep koopman model predictive control for enhancing transient stability in power grids. *International Journal of Robust and Nonlinear Control*, 31(6):1964–1978, 2021.

[23] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25:1307–1346, 2015.

[24] Bingzi Jin and Xiaojie Xu. Wholesale price forecasts of green grams using the neural network. *Asian Journal of Economics and Banking*, ahead-of-print(ahead-of-print), Jan 2024.

[25] Bingzi Jin and Xiaojie Xu. Price forecasting through neural networks for crude oil, heating oil, and natural gas. *Measurement: Energy*, 1:100001, 2024.

[26] Bingzi Jin and Xiaojie Xu. Carbon emission allowance price forecasting for china guangdong carbon emission exchange via the neural network. *Global Finance Review*, 6(1):3491, Jul. 2024.

[27] Bingzi Jin, Xiaojie Xu, and Yun Zhang. Thermal coal futures trading volume predictions through the neural network. *Journal of Modelling in Management*, ahead-of-print(ahead-of-print), Jan 2024.

[28] Aras Dargazany. Drl: Deep reinforcement learning for intelligent robot control–concept, literature, and future. *arXiv preprint arXiv:2105.13806*, 2021.

[29] Johanna Andrea Hurtado Sánchez, Katherine Casilimas, and Oscar Mauricio Caicedo Rendon. Deep reinforcement learning for resource management on network slicing: A survey. *Sensors*, 22(8):3031, 2022.

[30] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[31] Bingzi Jin and Xiaojie Xu. Machine learning coffee price predictions. *Journal of Uncertain Systems*, 17(04):2450023, 2024.

[32] Bingzi Jin and Xiaojie Xu. Forecasts of thermal coal prices through gaussian process regressions. *Ironmaking & Steelmaking*, 51(8):819–834, 2024.

[33] Bingzi Jin and Xiaojie Xu. Regional steel price index predictions for north china through machine learning. *International Journal of Mining and Mineral Engineering*, 15(3):314–350, 2024.

[34] Bingzi Jin and Xiaojie Xu. Forecasts of coking coal futures price indices through gaussian process regressions. *Mineral Economics*, Sep 2024.

[35] Jiao Bao and Mao Ye. Scale invariant constrained deep network for head pose estimation. *Adv. Model. Anal. B*, 59(1):113–130, 2016.

[36] Longhui Zhou, Hongfeng Tao, Wojciech Paszke, Vladimir Stojanovic, and Huizhong Yang. PD-type iterative learning control for uncertain spatially interconnected systems. *Mathematics*, 8(9):1528, 2020.

[37] Ruoxia Li and Jinde Cao. Stability analysis of reaction-diffusion uncertain memristive neural networks with time-varying delays and leakage term. *Applied Mathematics and Computation*, 278:54–69, 2016.

[38] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[39] Feng Ding, Guanfeng Ma, Zhikui Chen, Jing Gao, and Peng Li. Averaged soft actor-critic for deep reinforcement learning. *Complexity*, 2021(1):6658724, 2021.

[40] Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *International Conference on Machine Learning*, pages 834–843. PMLR, 2017.

[41] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[42] Jianan Liu, Zike Zhou, Wenjing Hong, and Jia Shi. 2D iterative learning control with deep reinforcement learning compensation for the non-repetitive batch processes. *Authorea Preprints*, 2023.

[43] Ronghu Chi, Biao Huang, Zhongsheng Hou, and Shangtai Jin. Data-driven high-order terminal iterative learning control with a faster convergence speed. *International Journal of Robust and Nonlinear Control*, 28(1):103–119, 2018.