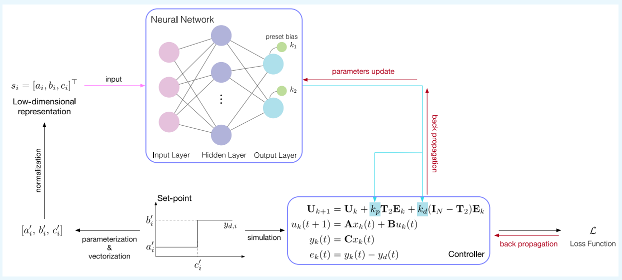
Learning of Iterative Learning Control for Flexible Manufacturing of Batch Processes

Libin Xu, Weimin Zhong,* Jingyi Lu, Furong Gao, Feng Qian, and Zhixing Cao*

Cite This: *ACS Omega* 2022, 7, 19939−19947 Read Online

ABSTRACT: Flexible manufacturing as an essential component of smart manufacturing implements the customized production mode, thereby requesting fast controller adaptation for producing different goods but still with high precision. This problem becomes even more acute for batch processes. Here we present a solution called learning of iterative learning control (ILC) based on neural networks. It is able to recommend control parameters for ILC controllers accordingly, so as to yield fast tracking error convergence and smaller steady-state error for disparate set-point profiles, which is deemed an abstraction of different production needs. The method substantially outperforms a benchmark ILC on a variety of systems and cases, thereby showing its potential for deployment in the industrial Internet of Things.


Neural Network
preset bias
k_1
k_2
Input Layer
Hidden Layer
Output Layer
$s_i = [a_i, b_i, c_i]^\top$
input
Low-dimensional representation
normalization
parameters update
back propagation
Set-point
b_i'
a_i'
c_i'
$y_{d,i}$
$[a_i', b_i', c_i']$
parameterization & vectorization
simulation
$\mathbf{U}_{k+1} = \mathbf{U}_k + k_p\mathbf{T}_2\mathbf{E}_k + k_d(\mathbf{I}_N - \mathbf{T}_2)\mathbf{E}_k$
$u_k(t+1) = \mathbf{A}x_k(t) + \mathbf{B}u_k(t)$
$y_k(t) = \mathbf{C}x_k(t)$
$e_k(t) = y_k(t) - y_d(t)$
Controller
back propagation
$\mathcal{L}$
Loss Function

■ INTRODUCTION

Batch processes are among the two predominant production approaches in modern industry, which fundamentally support the development of many high-end industries producing such items as semiconductors and pharmaceuticals.[1,2] Despite the inferior production efficiency as generally compared to continuous processes, batch processes are indispensable and in fact are gaining ever-increasing attention. Such an observation is underpinned by a twofold reason: (i) goods of remarkable complexity and also of high added value are produced in a batch processing fashion with a multitude of processing steps organized sequentially, which are yet difficult to reconfigure to satisfy continuous production constraints; (ii) the rapidly fluctuating customer demands and the increasing pursuit of personalization in the present society collectively give birth to flexible manufacturing, which is largely tantamount to producing goods in small batches with myriads of disparate configurations. This is indeed the outstanding merit of batch processes. As small as the production scale may be, there are difficulties in precise regulation. On top of the notorious presence of considerable nonlinearity, time variation, and uncertainties incurred by the underlying complex mechanisms,[3] such flexibility renders the precise regulation of batch processes an even more daunting task.

Just as every coin has two sides, a notable shortcut is enabled by the repeated operational pattern of batch processes. This is iterative learning control (ILC), which was initially devised for robot arm regulation[4] and is essentially a feedforward controller in stark contrast with most classic controllers such as PID or model predictive control (MPC).[5] The underlying idea is revolutionary, as it vividly mimics the learning process of human beings and well explains the word ”learning” it bears. An ILC controller distills information from the tracking error in the past to better tune the control input of the present trial (termed “batch” thereafter), etc., until achieving the perfect tracking of the given set-point profile. Notably, over the past decades, a multitude of achievements for better ILC have been witnessed both theoretically[6−11] and practically.[12−17] Encouraging endeavors of applying ILC in practice include injection molding,[18] bioreactors,[13] and batch chemical reactors,[19] whereas considerable theoretical efforts are devoted to answering the longstanding question—how to synthesize an ILC controller against various uncertainties. Such efforts include the introduction of feedback,[7] multipoint compensation,[8] adaptive tuning,[9,11] and optimal design[20] for linear systems and nonlinear systems. The review here is apparently not exhaustive due to limited space, and readers are encouraged to refer to excellent surveys in refs 3 and 21. Yet, readers should bear in mind that almost all the aforementioned are only suitable for one specific set-point profile except for adaptive ones. Despite the ability of the adaptively tuned controllers to track multiple set-point profiles, its complicated controller structure requires substantial expertise for intricate implementation in a real setting, which

Received: March 22, 2022
Accepted: May 13, 2022
Published: May 30, 2022

is not usually satisfied in industry. The ILC design for multiple set-point profiles is, to the best of our knowledge, rarely discussed. Notably, we will show that a universal ILC design leads to divergent control performance for different set-point profiles.

Arguably, this problem is pivotal to flexible manufacturing. The variation of set-point profiles abstractly stands for the switching of processing needs for producing different goods, and fast catering for such needs indicates the profit improvement and waste reduction, e.g., unqualified products, thereby calling for quick deployment of a precise controller. In this paper, we intend to present an intelligent system to recommend suitable ILC controllers for different set-point profiles so as to achieve faster convergence, which means waste reduction, and smaller steady-state tracking error, which means improved quality. Such an intelligent system is implemented via neural networks, more specifically, multilayer perceptrons. The most recent decade has witnessed the profound impact neural networks make in many domains, including playing the Go game,[22] industrial processes,[23,24] natural language processing,[25] and understanding gene expression.[26] The near-omnipotence of the neural network stems from the universal approximation theorem,[27,28] which states that a one-layer feedforward neural network is able to approximate any continuous function, as long as there are adequate neurons. Such a characteristic perfectly suits our need to develop quantitative mapping from set-point profiles to ILC controllers. The development of such mapping serves as the core of this paper.

Indeed, there are endeavors integrating ILC and neural network for better tracking performance reported in the literature. A neural network based ILC reported in ref 29 uses a neural network to approximate the nonlinear component in ILC output so as to achieve precise positioning compensation as well as expedite the iteration convergence. Similarly, ref 30 proposes a learning process with adaptable training parameters for both the intra- and interbatch domains and further shows that the synthesis of the controller is independent of any linearization and any complex optimization problem. Both attempts illustrate that neural networks can play an important role in the synthesis of the ILC controller of improved performance; yet neither is suitable for the case in flexible manufacturing with varying production needs, where the timely and expedient controller tuning to meet the production needs matters more. As such, we make use of neural networks to develop a recommender system suggesting controller configurations accordingly to achieve fast and precise ILC regulation, or equivalently better quality and higher production efficiency simultaneously.

The remainder of the paper is organized as follows: Section 2 presents the system formulation and a motivating example; the main method is described in Section 3; results are discussed in Section 4; and Section 5 concludes the work and provides an outlook.

■ PROBLEM STATEMENT

System Formulation. Without loss of generality, we assume that the system of interest is in the form

$$\begin{cases} x_k(t+1) &= f[x_k(t), u_k(t)] \\ y_k(t) &= g[x_k(t)] \end{cases} \tag{1}$$

where $u_k(t) \in \mathbb{R}^{n_u}$, $x_k(t) \in \mathbb{R}^{n_x}$, and $y_k(t) \in \mathbb{R}^{n_y}$ are the input signal, the internal state, and the output signal of the system, respectively, with n_u, n_x, and n_y being the dimensions. Besides, $k \in [1, \infty)$ and $t \in [1, T]$ are the cycle (or batch) and time indices, respectively. The cycle duration is denoted as T. The functions f and g are smooth functions. Such a formulation is so general to cover most cases reported in the literature.[3,31]

The ILC control can be presented in the following general form

$$u_{k+1}(t) = h[u_k(t), e_k(t), e_{k+1}(t)] \tag{2}$$

If the real-time information $e_{k+1}(t)$ is not incorporated, the ILC control law reduces to the feedforward type—its original flavor. As a pilot study, we will only focus on the classic PD-type ILC,[32] which is

$$u_{k+1}(t) = u_k(t) + k_p e_k(t) + k_d[e_k(t+1) - e_k(t)] \tag{3}$$

Here, the tracking error is defined as

$$e_k(t) = y_k(t) - y_d(t) \tag{4}$$

with $y_d(t)$ being the set-point profile. The parameters k_p and k_d in eq 3 are proportional and derivative gains that define the ILC control performance, thereby calling for careful tuning. Note that the derivative is approximated by one-step backward finite difference in eq 3, owing to the discrete-time nature of the system eq 1.

Indeed, the control law in eq 3 can be reorganized into a compact form so as to improve the efficiency of numeric implementation. By collecting $e_k(t)$ and $u_k(t)$ of the entire duration and formulating supervectors, one can have that

$$\mathbf{U}_{k+1} = \mathbf{U}_k + k_p T_2 \mathbf{E}_k + k_d(\mathbf{I}_N - T_2)\mathbf{E}_k \tag{5}$$

where $\mathbf{I}_N \in \mathbb{R}^{N\times N}$ is an identity matrix, the operator matrix T_2 is

$$T_2 = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \tag{6}$$

$$\mathbf{E}_k = [e_k(1), e_k(3), \dots, e_k(T)]^T \tag{7}$$

and

$$\mathbf{U}_k = [u_k(0), u_k(1), \dots, u_k(T-1)]^T \tag{8}$$

Specifically, if the system in eq 1 becomes a linear time-invariant (LTI) system, i.e.,

$$\begin{cases} x_k(t+1) &= \mathbf{A}x_k(t) + \mathbf{B}u_k(t) \\ y_k(t) &= \mathbf{C}x_k(t) \end{cases} \tag{9}$$

with matrices **A**, **B**, and **C** standing for system matrix, input matrix, and output matrix, respectively, it is also possible to rewrite the LTI system into a compact form

$$\mathbf{Y}_k = \mathbf{G}_x x_k(0) + \mathbf{G}_u \mathbf{U}_k \tag{10}$$

by applying the same trick as before. Here in eq 10, the matrices $\mathbf{G}_x$ and $\mathbf{G}_u$ are as follows

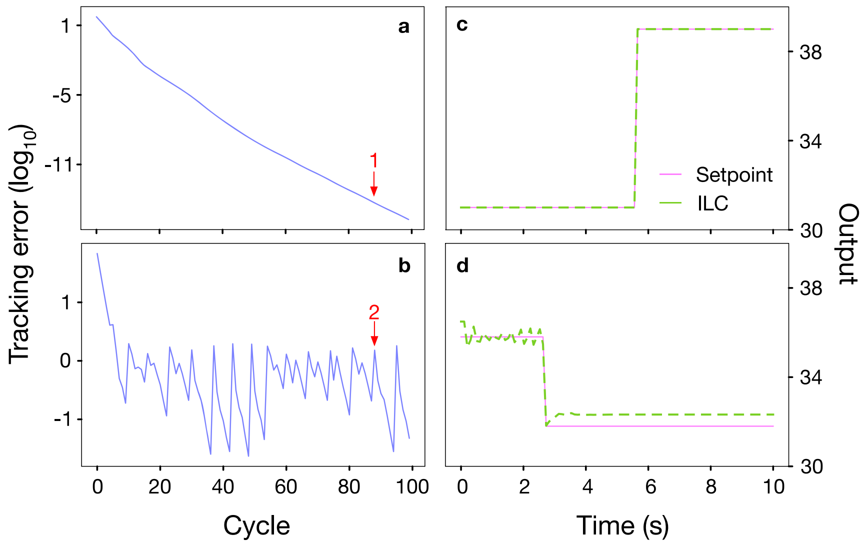


Figure 1. ILC controller performance is highly sensitive to set-point profiles. (a,b) For the same system, eq 14 regulated by the same PD-type ILC law, different set-point profiles lead to divergent responses of the cycle tracking error. It clearly shows that an ILC controller that achieves monotonic decrease on tracking error for one set-point profile still may have a cyclewise fluctuating tracking error for another. At point 1, perfect tracking is achieved, whereas the tracking performance is rather poor at point 2. (c,d) Corresponding process output of points 1 and 2 indicated in (a) and (b).

$$\mathbf{G}_x = \begin{bmatrix} \mathbf{CA} \\ \mathbf{CA}^2 \\ \vdots \\ \mathbf{CA}^T \end{bmatrix} \tag{11}$$

$$\mathbf{G}_u = \begin{bmatrix} \mathbf{CB} & 0 & \dots & 0 \\ \mathbf{CAB} & \mathbf{CB} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{CA}^{T-1}\mathbf{B} & \dots & \dots & \mathbf{CB} \end{bmatrix} \tag{12}$$

whereas the supervectors thereof are

$$\mathbf{Y}_k = [y_k(1), y_k(3), ..., y_k(T)]^T \tag{13}$$

Again, the form in eq 10 is helpful for numeric implementation. Furthermore, the matrices **A**, **B**, and **C** can be functions of time, i.e., **A**(t), **B**(t), and **C**(t). If so, the system of interest becomes a linear time-varying (LTV) system, which is, in some literature,[33] thought to be a linearization of a nonlinear system around a given set-point profile. Note that similar formulation in eq 10 is also valid for LTV systems but with slight modifications. The objective of the paper is to present a function mapping from the set-point profile $y_d(t)$ to PD-type ILC parameters k_p and k_d so as to minimize some function of tracking error $e_k(t)$, which is generally interpreted as the control performance.

Motivating Example. Next we will show why carefully tuning k_p and k_d for each set-point profile is of great importance by using a toy nonlinear system as an example. Let us consider the system

$$\begin{aligned} x_{1k}(t+1) &= \sin x_{1k}(t) + x_{2k}(t) \\ x_{2k}(t+1) &= \sin x_{1k}(t) + u_k(t) \end{aligned} \tag{14}$$

which is regulated by PD-type ILC with $k_p = k_d = -0.3$, and the second internal state x_{2k} serves as the process output as well. The system is operated within a duration $T = 10$ s, and its data is collected every 0.1 s. That means there are 100 data points in each cycle. It clearly shows in Figure 1 that a fine-tuned ILC controller that works well for one set-point profile may not work for another, even possibly leading to tracking error fluctuation (see Figure 1b). Either slow convergence or fluctuation of tracking error indicates the economic loss in practice. Hence, what people expects from ILC is the monotonic convergence of tracking error, which mathematically means

$$\|\mathbf{E}_{k+1}\| \le \|\mathbf{E}_k\| \tag{15}$$

for any positive integer k. This point is not new and has been bred in refs 34 and 35 and later strongly emphasized in ref 11. In short summary, the sensitivity of the ILC performance to set-point profile change, particularly the marked performance degradation, motivates us to develop a mapping from set-point profile to k_p and k_d.

■ METHODS

Prior to establishing such a mapping, one needs to figure out how to represent different set-point profiles. People may argue to use the supervector $\mathbf{Y}_d = [y_d(1), ..., y_d(T)]^T$ for the purpose; however, the high dimension of $\mathbf{Y}_d$ may impose a burden on the subsequent model training, for example, by increasing computational cost. Hence, in general, it is not trivial to introduce a low-dimensional representation. Fortunately, set-point profiles are not arbitrary in practice but in some standard form, for instance, step-change signal and slope-climbing signal. These signals can be represented by much shorter vectors. For instance, a step-change signal can be determined by three factors a, b, and c, where a, b, are the levels of the steps prior to and after the change, respectively, and c is the time when the change occurs. As such, any step-change signal can be conveniently represented as a point $s_i = [a_i, b_i, c_i]^T$ in space $\mathbb{R}^3$ as shown in Figure 2. By randomly sampling points in the space of s_i, one can get a set $\mathbf{S} = [s_1, s_2, ..., s_M]$ representing

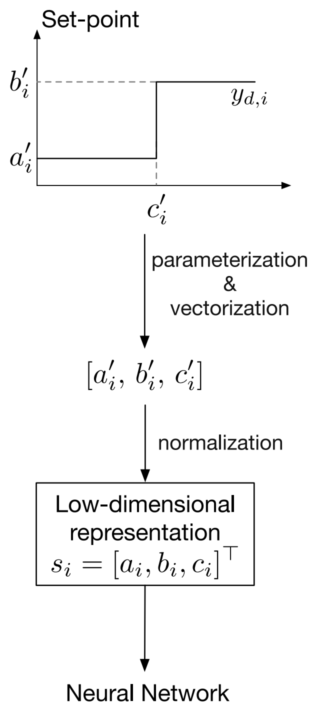

Set-point
b_i'
a_i'
$y_{d,i}$
c_i'
parameterization & vectorization
$[a_i', b_i', c_i']$
normalization
Low-dimensional representation $s_i = [a_i, b_i, c_i]^\top$
Neural Network

Figure 2. Parameterization and vectorization of set-point profiles in the form of step change.

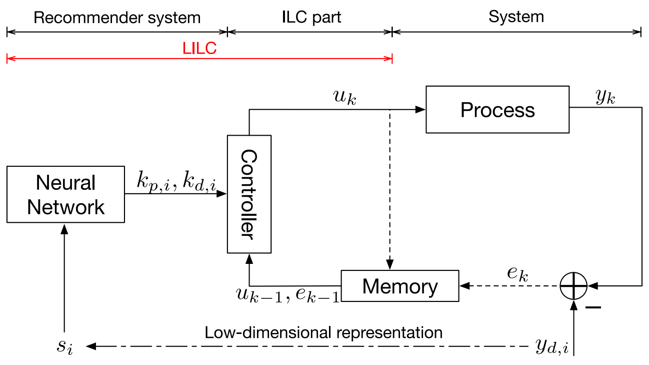


Figure 3. Block diagram of the proposed LILC. The neural network aided recommender system quickly suggests appropriate $k_{p,i}$ and $k_{d,i}$ for the ILC controller according to the low-dimensional representation of set-point profile y_d, while catering to the needs of fast-tracking error convergence.

a collection of set-point profiles. Indeed, such low-dimensional representation is general, as many complex set-point profiles can be approximated by a series of step-change signals. For the sake of neat presentation, we only focus on step-change set-point profiles.

Due to the powerful capability of functional approximation of the neural network, the mapping is decided to be neural-network-based. That is a mapping

$$\mathrm{NN}_\theta: s_i \to \{k_{p,i}, k_{d,i}\} \tag{16}$$

where θ encapsulates weights and biases of the neural network and will be determined through training. The neural network we use in this paper is the feedforward multilayer perceptron (MLP). If well trained, the neural network together with the PD-type ILC constitutes the learning of iterative learning control (LILC), the main result of the paper, which is shown in Figure 3.

Next we are going to fill in the last puzzle—the loss function for training. First, we define the following tracking error index

$$\mathcal{E}_i = \frac{1}{NT}\sum_{k=1}^{N} \|\mathbf{E}_k\|_2^2 = \frac{1}{NT}\sum_{k=1}^{N}\sum_{t=1}^{T} e_k^2(t) \tag{17}$$

for each recommended $\{k_{p,i}, k_{d,i}\}$ and given set-point profile $y_{d,i}(t)$. The index $\mathcal{E}_i$ sums the tracking error of the first N cycles, thereby implicitly emphasizing the cyclewise decrease of tracking error. Note that N is a hyperparameter that needs tuning and should not be too small; otherwise, the steady state may not reached. If one would like to yield a smaller steady-state tracking error, a larger weight can be imposed on the term $\|\mathbf{E}_N(t)\|$. Then, by summing the error index for each point in the set S, one can have the loss function

$$\mathcal{L}(\theta) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{E}_i \tag{18}$$

for a given θ. Following that, the neural network training becomes an optimization problem

$$\begin{aligned} &\min_\theta \quad \mathcal{L}(\theta) \\ &\text{s.t.} \quad \text{eqs (1), (3)} \end{aligned} \tag{19}$$

Such an optimization problem can be solved by many standard optimization tools; however, given its neural network structure, the back-propagation (BP) algorithm is probably more efficient than others. Note that BP is still a gradient base method, and the gradient $\nabla_\theta \mathcal{L}$ thereof can be calculated by automatic differentiation, which has been included in many machine learning packages such as *PyTorch*. Besides, the Adam

Algorithm 1 Train $\mathrm{NN}_\theta : s_i \to \{k_{p,i}, k_{d,i}\}$

```
1: **Input:** three parameters of step signal: $s_i = \{a_i, b_i, c_i\}$
2: **Output:** the optimal $k_{p,i}, k_{d,i}$
3: Initialize network parameters: $\theta$
4: Generate $M$ points in $\mathbb{R}^3$, $\mathbf{S} = [s_1, s_2, \cdots, s_M]$
5: for $j = epoch$ to $j_{max}$ do
6:     Generate $\{k_{p,i}, k_{d,i}\}$ for each $s_i$ by current neural network $\mathrm{NN}_{\theta_j}$
7:     Convert each $s_i$ into a step-change set-point profile, simulate the system with parameters $\{k_{p_i}, k_{d_i}\}$, and calculate the loss of the first $N$ batches: $\mathcal{E}_i = \frac{1}{NT}\sum_{k=1}^{N}\sum_{t=1}^{T} e_k^2(t)$
8:     Calculate Loss: $\mathcal{L}_j = \frac{1}{M}\sum_{i=1}^{M} \mathcal{E}_i$
9:     Use BP to calculate gradient: $\nabla_{\theta_j}\mathcal{L}$
10:    Update network parameters: $\theta_{j+1} \leftarrow \mathrm{Adam}(\theta_j, \nabla_{\theta_j}\mathcal{L})$
11:    $j \leftarrow j + 1$
12: end for
```

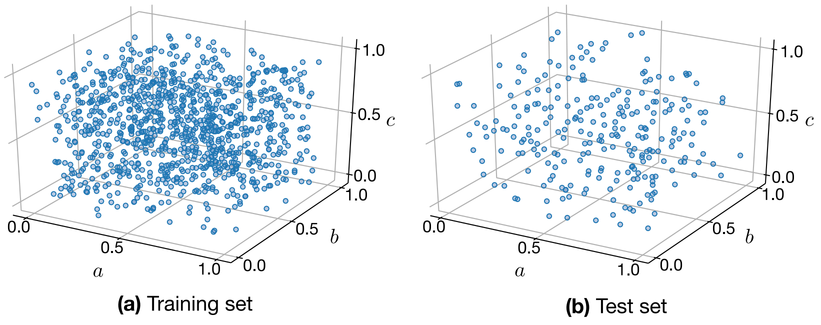


Figure 4. Data distribution of the training set (a) and the test set (b) in the normalized space $[0,1]^3$.

optimizer[36] can be used to update the neural network parameters θ. After each update, the neural network is able to recommend a new batch of $\{k_{p,i}, k_{d,i}\}$, which is fed to the system regulated by ILC for simulation and calculation of $\mathcal{E}_i$. It should be noted that given the structure of the loss function, the simulation step can be implemented in parallel to accelerate the training. Subsequently, the new gradient is computed again and used to update θ. These steps keep looping until a proper neural network aided recommender system is obtained. The entire training procedures of LILC are summarized in Algorithm 1.

Note that variable cycle duration usually occurs for batch processes, particularly in pharmaceutical industry, thereby becoming an important issue for iterative learning control. Indeed, there are many solutions reported,[37,38] among which the truncation method is the simplest.[37] Our proposed method here can be easily extended to cater for variable cycle duration by the truncation method, i.e., equating the cycle duration T to the minimal duration of all cycles.

NUMERICAL EXPERIMENTS

Data Set. First, we sampled 1250 data points uniformly from the normalized space $[0,1]^3$, of which 80% (1000 data points) form the training set and the rest become the test set. The data distributions of the training set and the test set in the normalized space $[0,1]^3$ are visualized in Figure 4. By doing so, we impose constraints on the range of the low dimension representation of set-point profiles, and it matches the reality that set-point profiles are not allowed to be arbitrarily chosen but are within a certain range. For linear systems, the range of a and b is [30, 40], whereas that of nonlinear systems is [3, 6]. The range of c is [200, 800] for linear systems, while it is [20, 80] for nonlinear systems. The data points should be scaled as per the ranges and converted into the set-point profiles y_d for simulation steps in the training.

Neural Network Training. Here we use a three-layer MLP wherein there are 3, 10, and 2 neurons in the input, hidden, and output layers, respectively. All the activation functions are *ReLU*. All the weights are initialized with Xavier uniform[39] with a gain of 0.05, while the biases are set to 0 except the ones of the output layer, which are set to the fixed values of k_p and k_d of some benchmark and will be detailed later. The neural network is trained by using Adam optimizer with the learning rate set to 0.001. The training is implemented in a minibatch fashion with a size of 250.

Note that for some initialization, the network may generate k_p and k_d that yield tracking error divergence and interrupt the training process. To circumvent the problem, the neural network parameters are initialized in a small-value region, and the biases of the output layer are set to a pair of k_p and k_d that yields a converged tracking error. This is akin to the idea of fine-tuning of neural networks.

RESULTS

LTI System. We first tested LILC on an LTI system, which is defined by matrices

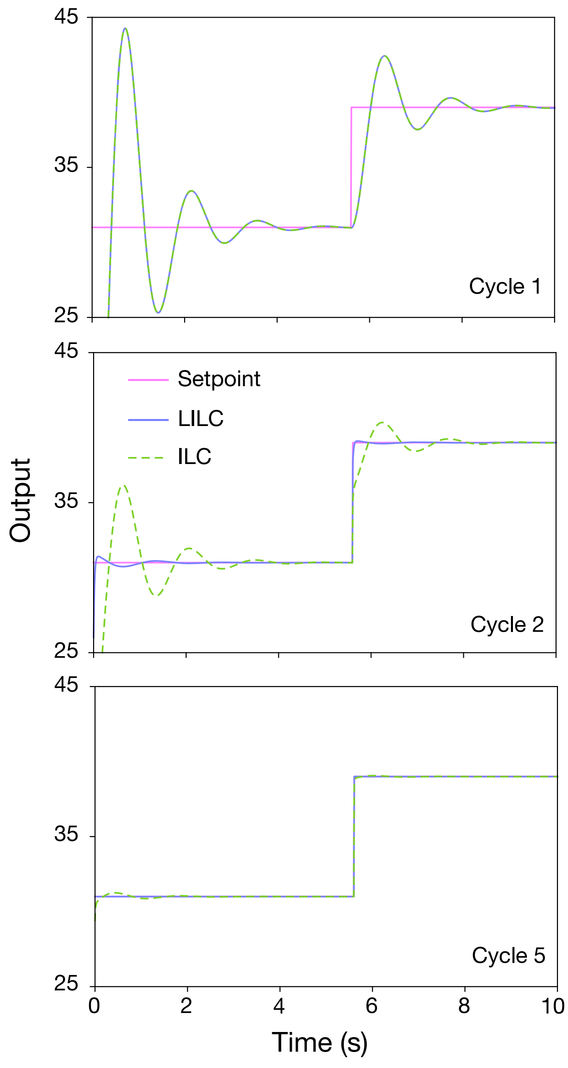


Figure 5. Control performance of benchmark ILC and LILC on an LTI system. The process outputs at cycles 1, 2, and 5 are plotted.

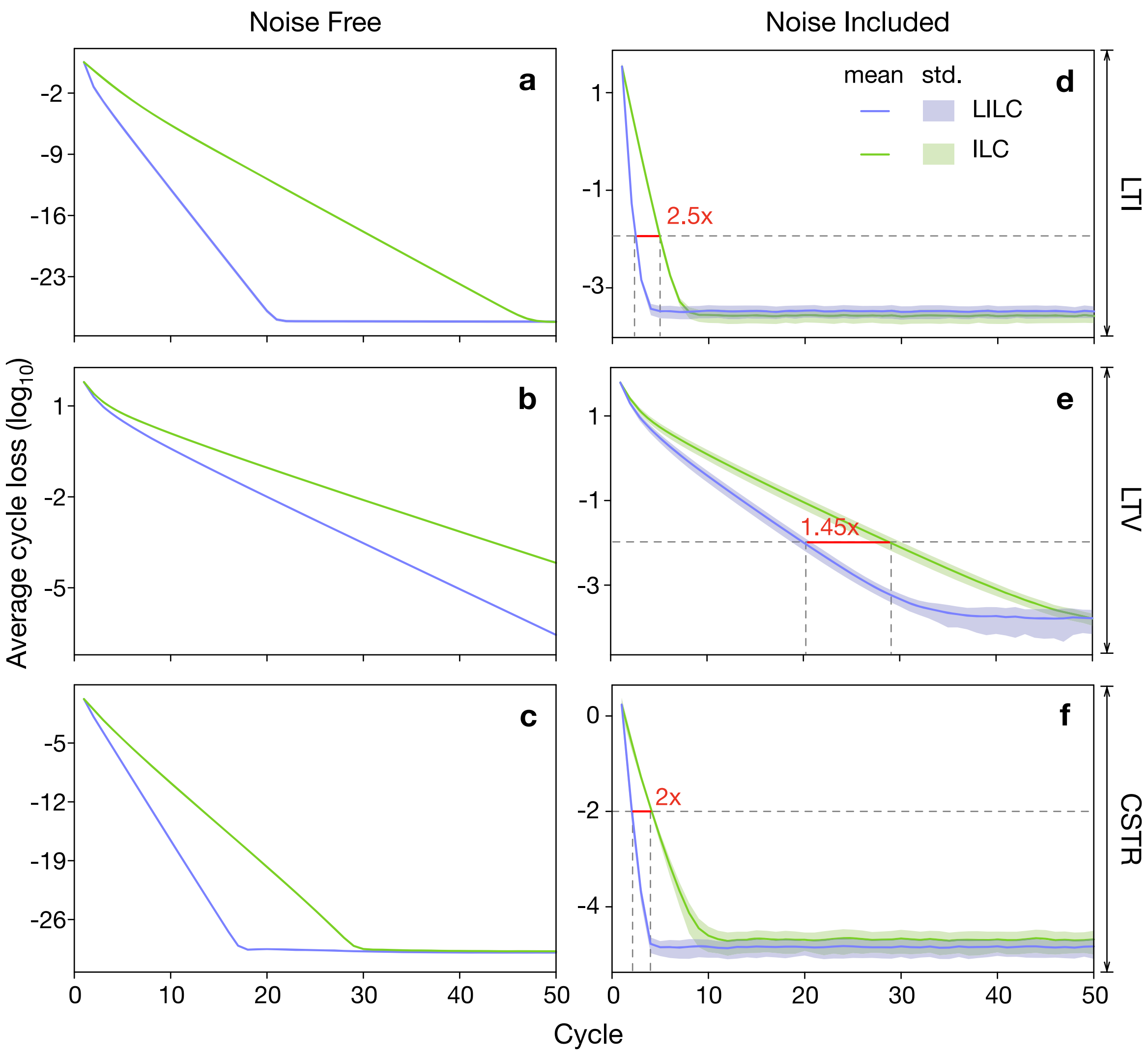


Figure 6. Control performance of LILC and the benchmark ILC indexed by ACL is compared on an LTI system (a,d), an LTV system (b,e), and a nonlinear system (c,f). (a), (b), and (c) correspond to the noise-free case, whereas (d), (e), and (f) are for the case wherein the process noise is subject to unit normal distribution. The blue stands for LILC, while green stands for the benchmark ILC. The mean (solid line) and standard deviation (std, shaded area) are calculated for all the data points in the test set.

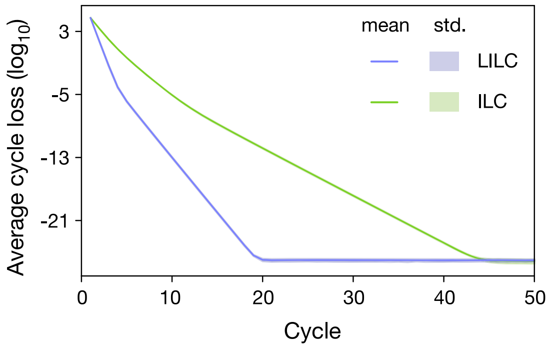


Figure 7. Averaged cycle loss of LILC and benchmark ILC in the repetitive disturbance case.

$$\mathbf{A} = \begin{bmatrix} 1.582 & -0.5916 \\ 1 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 1.69 \\ 1.49 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^T \tag{20}$$

The system indeed describes the typical dynamics of injection molding.[40,41] The sampling period of the system is 0.01 s, and the cycle duration is 10 s, which equivalently means there are $T = 1000$ points in a cycle. The internal states are initialized as $x_k(0) = [0, 0]^T$. The benchmark ILC that LILC will be compared against has the controller parameters $k_p = -0.01$ and $k_d = -0.3$. In order to achieve accelerated convergence of the averaged cycle loss (ACL), both ILC controllers are initialized with a PI controller in the first cycle, whose control law is

$$u_1(t) = u_1(t-1) + K[y_r(t-1) - y_1(t-1)] \tag{21}$$

In this case, K is set to 0.001. The hyperparameter N plays an important role in controlling performance, and it should generally be chosen to be not less than 20 so as to ensure that the steady state is reachable. An empirical value $N = 50$ is selected to appropriately trade off the transient and steady-state control performance, and the value will be used in the remaining examples of the paper. The control performance of benchmark ILC and LILC is compared in Figure 5, and it

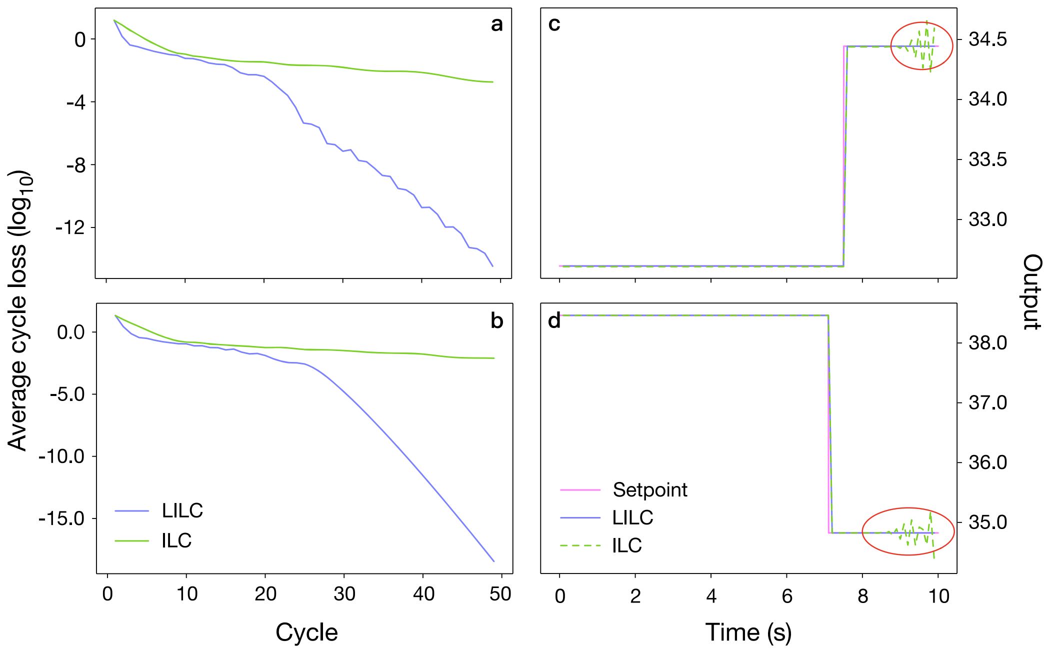


Figure 8. Averaged cycle loss and the tracking performance at cycle 50 when LILC and benchmark ILC tracking two different set-point profiles for the motivating example eq 14. (a) and (b) correspond to the averaged cycle losses of two different set points, respectively. (c) and (d) correspond to the tracking performances at cycle 50.

clearly shows that LILC is able to track the given set-point profile almost perfectly in cycle 2, whereas benchmark ILC still has a marked overshoot. The averaged cycle loss (ACL) is defined as the squared error averaged for each time point within a cycle. Such an index as a function of the cycle of both ILC and LILC is plotted in Figure 6a and d for noise-free and unit normal process noise, respectively. For the noisy case, if the accepting ACL is less than 0.01 (denoted as dashed gray line in Figure 6), LILC converged 2.5 times faster than the benchmark ILC, implying a remarkable reduction of waste.

In some batch processes, there exist repetitive disturbances which need to be rejected. Here we also show the capability of LILC to reject the repetitive disturbance in the LTI system. Within this example, the benchmark and neural network remain the same as mentioned before except the process noise, which replaced by a deterministic sine signal

$$w(t) = 100\sin\left(\frac{\pi}{5}t\right) \tag{22}$$

The result is shown in Figure 7, where LILC outperforms the benchmark ILC on repetitive disturbance rejection.

LTV System. LILC is further tested on an LTV system. All the configurations remain the same except the system matrix **A**, which has a slope change from the 200th to 700th data points, i.e.,

$$\mathbf{A} = \begin{bmatrix} 1.582 & -0.5916 \\ 1 & 0 \end{bmatrix} \xrightarrow{\text{Linear change}} \begin{bmatrix} 1.482 & -0.5916 \\ 1 & 0 \end{bmatrix} \tag{23}$$

The results for the LTV system are shown in Figure 6b and e. In both the noise-free and noisy cases, LILC robustly outperforms the benchmark ILC.

Nonlinear System. Another test is performed on a continuous stirred tank reactor (CSTR),[42] whose dynamics are described by

$$\begin{aligned} x_{1_k}(t+1) &= (1 - T_s\theta)x_{1_k}(t) + T_sD_a[1 - x_{1_k}(t)]e^{x_{2_k}(t)/\left(1+\frac{x_{2_k}(t)}{\gamma}\right)} \\ x_{2_k}(t+1) &= (1 - T_s\theta)x_{2_k}(t) + T_sD_a[1 - x_{1_k}(t)]e^{x_{2_k}(t)/\left(1+\frac{x_{2_k}(t)}{\gamma}\right)} \\ &\quad - T_s\beta x_{2_k}(t) + T_s\beta u_k(t) \end{aligned} \tag{24}$$

The sampling time T_s is equal to 0.1. The other parameters are $\theta = 1$, $\beta = 0.3$, $\gamma = 20$, and $D_a = 0.072$.[43] The second internal state x_{2_k} also serves the process output of the system and is required to follow the set-point profile y_d. The cycle duration is 10 s or equivalently $T = 100$ data points in a cycle. The system is initialized with $x_{1_k}(0) = 0.57$, $x_{2_k}(0) = 0.3$ for any k. The benchmark ILC is set with $k_p = -6.00$ and $k_d = -35$. Additionally, the PI controller for the first cycle is set with $K = 0.5$. The results for both cases are shown in Figure 6c and f, and substantial improvement of LILC against the benchmark ILC is clearly observed.

Motivating Example. Finally we present how to use the proposed LILC method to resolve the problem shown in Figure 1. Within this example, the benchmark ILC uses $k_p = k_d = -0.3$. The tracking error comparison in terms of ACL of both LILC and the benchmark ILC for two different set-point profiles is summarized in Figure 8a and b, where LILC outperforms the benchmark ILC on the speed of error convergence and steady-state tracking error. Indeed, the advantage of LILC in terms of steady-state tracking error is tangible. Such an observation is again confirmed in Figure 8. Figure 8c,d shows that LILC achieves almost perfect tracking, while the benchmark ILC starts to fluctuate after 8 s, which is generally unacceptable in industrial reality. In all, it again advocates the superior performance of LILC.

■ DISCUSSION

In this paper, we presented learning of the ILC method for batch processes that need to manufacture different products.

As a pilot study, different manufacturing needs for various products are abstracted as different set-point profiles for the same process. We used a toy nonlinear system as an example showing that different set-point profiles for the same process with the same ILC controller may lead to divergent regulation performance, thereby clearly showing the needs for adaptive ILC tuning for different set-point profiles. Set-point profiles were represented in low dimensions to facilitate the neural network training. The well-trained neural network was able to robustly outperform the benchmark ILC on an LTI system, an LTV system, and a nonlinear system no matter whether process noise is present or absent. Though used for ILC tuning, the method is quite general and is able to solve a range of tuning problems such as weight tuning of model predictive controller and controller tuning for multiagent systems. Hence, it is worth further exploring in the future.

It should be noted that the LILC serves for the controller tuning for one specific batch process. However, the LILC framework is rather flexible to achieve the interprocess generalization given that the class of the processes can be parametrized. These parameters can be lumped together with the parameters of set-points and are as a whole fed to neural networks. By collecting more data for various combination of processes and set-points, the intelligent recommendation for ILC controllers can be achieved.

In fact, there is an underlying assumption behind the method; that is, we require the model of the process to be readily available for training. Despite being seemingly strict at first glance, it is possible to satisfy in practice. Such a process model can be obtained by system identification based on data or derivation based on first-principles. The former is possible because of the abundance of data given the rapid development of 5G and cloud-based technology and increasing deployment of the industrial Internet of Things. For example, in the injection molding industry, such techniques can help to collect abundant data to develop a precise model for each type of injection molding machines of the same manufacturer. The mechanistic modeling is also possible, as some manufacturers provide such services by using their rich knowledge about equipment they sell. Alternatively, the transfer learning technique is also helpful to circumvent such an assumption. Indeed, this is also the major point to be distinguished from model-free optimization methods for batch processes.[44]

Additionally, it is also worthwhile to investigate the robustness of LILC, including the robustness against model mismatch, repeatable disturbance, and stochastic factors on different parts of a system, as well as its application to stochastic batch processes, biological processes in particular.[45−50]

AUTHOR INFORMATION

Corresponding Authors

Weimin Zhong − *MOE Key Laboratory of Smart Manufacturing in Energy Chemical Process, East China University of Science and Technology, Shanghai 200237, China;* orcid.org/0000-0002-4285-4739; Email: wmzhong@ecust.edu.cn

Zhixing Cao − *MOE Key Laboratory of Smart Manufacturing in Energy Chemical Process, East China University of Science and Technology, Shanghai 200237, China;* orcid.org/0000-0003-2600-5806; Email: zcao@ecust.edu.cn

Authors

Libin Xu − *MOE Key Laboratory of Smart Manufacturing in Energy Chemical Process, East China University of Science and Technology, Shanghai 200237, China*

Jingyi Lu − *MOE Key Laboratory of Smart Manufacturing in Energy Chemical Process, East China University of Science and Technology, Shanghai 200237, China; Department of Electrical Engineering and Information Technology, Paderborn University, 33098 Paderborn, Germany;* orcid.org/0000-0002-4889-6438

Furong Gao − *Department of Chemical and Biological Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong;* orcid.org/0000-0002-5900-1353

Feng Qian − *MOE Key Laboratory of Smart Manufacturing in Energy Chemical Process, East China University of Science and Technology, Shanghai 200237, China;* orcid.org/0000-0003-2781-332X

Complete contact information is available at: https://pubs.acs.org/10.1021/acsomega.2c01741

Author Contributions

Libin Xu, Weimin Zhong, Jingyi Lu, Feng Qian, and Zhixing Cao are with the MOE Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai, 200237, China. Jingyi Lu is also with the Department of Electrical Engineering and Information Technology, Paderborn University, 33098, Paderborn, Germany. Furong Gao is with the Department of Chemical and Biological Engineering, Hong Kong University of Science and Technology, Hong Kong.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

The work was supported by the National Natural Science Foundation of China under Grants 61925305, 62073137, 61988101, and Shanghai Action Plan for Technological Innovation Grant 22ZR1415300. The work is partially published as a conference paper in the Chinese Process Control Conference 2021.

REFERENCES

(1) Cao, Z.; Lu, J.; Gao, F. Batch process control—overview and outlook. *Acta Automatica Sinica* **2017**, *43*, 933−943.

(2) Bonvin, D.; Srinivasan, B.; Hunkeler, D. Control and optimization of batch processes. *IEEE Control Syst. Mag.* **2006**, *26*, 34−45.

(3) Lu, J.; Cao, Z.; Zhao, C.; Gao, F. 110th anniversary: An overview on learning-based model predictive control for batch processes. *Ind. Eng. Chem. Res.* **2019**, *58*, 17164−17173.

(4) Arimoto, S.; Kawamura, S.; Miyazaki, F. Bettering operation of robots by learning. *J. Robot. Syst.* **1984**, *1*, 123−140.

(5) Mayne, D. Q.; Rawlings, J. B.; Rao, C. V.; Scokaert, P. O. Constrained model predictive control: Stability and optimality. *Automatica* **2000**, *36*, 789−814.

(6) Owens, D. H. *Iterative learning control*; 2015.

(7) Shi, J.; Gao, F.; Wu, T.-J. Robust design of integrated feedback and iterative learning control of a batch process based on a 2D Roesser system. *J. Process Control* **2005**, *15*, 907−924.

(8) Lu, J.; Cao, Z.; Gao, F. Multipoint iterative learning mode predictive control. *IEEE Trans. Ind. Electron.* **2019**, *66*, 6230−6240.

(9) Chi, R.; Hou, Z.; Xu, J. Adaptive ILC for a class of discrete-time systems with iteration-varying trajectory and random initial condition. *Automatica* **2008**, *44*, 2207−2213.
(10) Chi, R.; Hou, Z.; Jin, S.; Huang, B. Computationally efficient data-driven higher order optimal iterative learning control. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 5971−5980.
(11) Lu, J.; Cao, Z.; Zhang, R.; Gao, F. Nonlinear monotonically convergent iterative learning control for batch processes. *IEEE Trans. Ind. Electron.* **2018**, *65*, 5826−5836.
(12) Gao, F.; Yang, Y.; Shao, C. Robust iterative learning control with applications to injection molding process. *Chem. Eng. Sci.* **2001**, *56*, 7025−7034.
(13) Estakhrouiyeh, M. R.; Vali, M.; Gharaveisi, A. Application of fractional order iterative learning controller for a type of batch bioreactor. *IET Control Theory Appl.* **2016**, *10*, 1374−1383.
(14) Cho, W.; Edgar, T. F.; Lee, J. Iterative learning dual-mode control of exothermic batch reactors. *Control Eng. Pract.* **2008**, *16*, 1244−1249.
(15) Choi, H.-K.; Kwon, J. S.-I. Multiscale modeling and control of Kappa number and porosity in a batch-type pulp digester. *AIChE J.* **2019**, *65*, No. e16589.
(16) Wang, L.; Yu, J.; Zhang, R.; Li, P.; Gao, F. Iterative learning control for multiphase batch processes with asynchronous switching. *IEEE Trans. Syst. Man Cybern.: Syst.* **2021**, *51*, 2536−2549.
(17) Son, S. H.; Choi, H.-K.; Kwon, J. S.-I. Application of offset-free Koopman-based model predictive control to a batch pulp digester. *AIChE J.* **2021**, *67*, No. e17301.
(18) Cao, Z.; Yang, Y.; Lu, J.; Gao, F. Two-time-dimensional model predictive control of weld line positioning in bi-injection molding. *Ind. Eng. Chem. Res.* **2015**, *54*, 4795−4804.
(19) Lu, P.-C.; Chen, J.; Xie, L. Iterative learning control (ILC)-based economic optimization for batch processes using helpful disturbance information. *Ind. Eng. Chem. Res.* **2018**, *57*, 3717−3731.
(20) Owens, D. H.; Hätönen, J. Iterative learning control—an optimization paradigm. *Annu. Rev. Control* **2005**, *29*, 57−70.
(21) Wang, Y.; Gao, F.; Doyle, F. J., III Survey on iterative learning control, repetitive control, and run-to-run control. *J. Process Control* **2009**, *19*, 1589−1600.
(22) Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484−489.
(23) Ou, C.; Zhu, H.; Shardt, Y. A.; Ye, L.; Yuan, X.; Wang, Y.; Yang, C. Quality-driven regularization for deep learning networks and its application to industrial soft sensors. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1−11.
(24) Yuan, X.; Jia, Z.; Li, L.; Wang, K.; Ye, L.; Wang, Y.; Yang, C.; Gui, W. A SIA-LSTM based virtual metrology for quality variables in irregular sampled time sequence of industrial processes. *Chem. Eng. Sci.* **2022**, *249*, 117299.
(25) Yu, D.; Hinton, G.; Morgan, N.; Chien, J.-T.; Sagayama, S. Introduction to the special section on deep learning for speech and language processing. *Trans. Audio Speech Lang. Process* **2012**, *20*, 4−6.
(26) Jiang, Q.; Fu, X.; Yan, S.; Li, R.; Du, W.; Cao, Z.; Qian, F.; Grima, R. Neural network aided approximation and parameter inference of non-Markovian models of gene expression. *Nat. Commun.* **2021**, *12*, 1−12.
(27) Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw* **1989**, *2*, 359−366.
(28) Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst* **1989**, *2*, 303−314.
(29) Zhang, D.; Wang, Z.; Masayoshi, T. Neural-network-based iterative learning control for multiple tasks. *IEEE Trans. Neural Netw. Learn Syst.* **2021**, *32*, 4178−4190.
(30) Patan, K.; Patan, M. Neural-network-based iterative learning control of nonlinear systems. *ISA Trans* **2020**, *98*, 445−453.
(31) Xu, J.-X. A survey on iterative learning control for nonlinear systems. *Int. J. Control* **2011**, *84*, 1275−1294.
(32) Chen, Y.; Moore, K. L. An optimal design of PD-type iterative learning control with monotonic convergence. *Proc. 2002 IEEE Int. Symp. Intell. Control* **2002**, 55−60.
(33) Cao, Z.; Yang, Y.; Yi, H.; Gao, F. Priori knowledge-based online batch-to-batch identification in a closed loop and an application to injection molding. *Ind. Eng. Chem. Res.* **2016**, *55*, 8818−8829.
(34) Lee, H.-S.; Bien, Z. A note on convergence property of iterative learning controller with respect to sup norm. *Automatica* **1997**, *33*, 1591−1593.
(35) Moore, K. L.; Chen, Y.; Bahl, V. Monotonically convergent iterative learning control for linear discrete-time systems. *Automatica* **2005**, *41*, 1529−1537.
(36) Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv*, 1412.6980, 2014.
(37) Cao, Z.; Yang, Y.; Lu, J.; Gao, F. Constrained two dimensional recursive least squares model identification for batch processes. *J. Process Control* **2014**, *24*, 871−879.
(38) Shen, D.; Saab, S. S. Noisy output based direct learning tracking control with markov nonuniform trial lengths using adaptive gains. *IEEE Trans. Automat. Contr.* **2021**, 1.
(39) Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. *Proc. AISTATS* **2010**, 249−256.
(40) Cao, Z.; Zhang, R.; Lu, J.; Gao, F. Online identification for batch processes in closed loop incorporating priori controller knowledge. *Comput. Chem. Eng.* **2016**, *90*, 222−233.
(41) Cao, Z.; Lu, J.; Zhang, R.; Gao, F. Online average-based system modelling method for batch process. *Comput. Chem. Eng.* **2018**, *108*, 128−138.
(42) Lu, J.; Cao, Z.; Hu, Q.; Xu, Z.; Du, W.; Gao, F. Optimal iterative learning control for batch processes in the presence of time-varying dynamics. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 680−692.
(43) Wu, F. LMI-based robust model predictive control and its application to an industrial CSTR problem. *J. Process Control* **2001**, *11*, 649−659.
(44) Kong, X.; Yang, Y.; Chen, X.; Shao, Z.; Gao, F. Quality control via model-free optimization for a type of batch process with a short cycle time and low operational cost. *Ind. Eng. Chem. Res.* **2011**, *50*, 2994−3003.
(45) Cao, Z.; Yu, J.; Wang, W.; Lu, H.; Xia, X.; Xu, H.; Yang, X.; Bao, L.; Zhang, Q.; Wang, H.; et al. Multi-scale data-driven engineering for biosynthetic titer improvement. *Curr. Opin. Biotechnol.* **2020**, *65*, 205−212.
(46) Cao, Z.; Filatova, T.; Oyarzún, D. A.; Grima, R. A stochastic model of gene expression with polymerase recruitment and pause release. *Biophys. J.* **2020**, *119*, 1002−1014.
(47) Holehouse, J.; Cao, Z.; Grima, R. Stochastic modeling of autoregulatory genetic feedback loops: A review and comparative study. *Biophys. J.* **2020**, *118*, 1517−1525.
(48) Cao, Z.; Grima, R. Analytical distributions for detailed models of stochastic gene expression in eukaryotic cells. *Proc. Natl. Acad. Sci. U.S.A.* **2020**, *117*, 4682−4692.
(49) Cao, Z.; Grima, R. Accuracy of parameter estimation for auto-regulatory transcriptional feedback loops from noisy data. *J. R. Soc. Interface* **2019**, *16*, 20180967.
(50) Cao, Z.; Grima, R. Linear mapping approximation of gene regulatory networks with stochastic dynamics. *Nat. Commun.* **2018**, *9*, 1−15.