Available online at www.sciencedirect.com

ScienceDirect

Journal of the Franklin Institute 355 (2018) 3641–3658

www.elsevier.com/locate/jfranklin

Terminal iterative learning control for discrete-time nonlinear systems based on neural networks☆

Jian Han[a,b], Dong Shen[a,*], Chiang-Ju Chien[c]

[a] *College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, PR China*

[b] *Informatics Institute, Faculty of Science, University of Amsterdam, Amsterdam 1098XH, The Netherlands*

[c] *Department of Electronic Engineering, Huafan University, New Taipei City 22301, Taiwan*

Received 16 December 2016; received in revised form 13 February 2018; accepted 14 March 2018

Available online 20 March 2018

Abstract

The terminal iterative learning control is designed for nonlinear systems based on neural networks. A terminal output tracking error model is obtained by using a system input and output algebraic function as well as the differential mean value theorem. The radial basis function neural network is utilized to construct the input for the system. The weights are updated by optimizing an objective function and an auxiliary error is introduced to compensate the approximation error from the neural network. Both time-invariant input case and time-varying input case are discussed in the note. Strict convergence analysis of proposed algorithm is proved by the Lyapunov like method. Simulations based on train station control problem and batch reactor are provided to demonstrate the effectiveness of the proposed algorithms.

1. Introduction

Iterative learning control (ILC) was first proposed by Arimoto in 1980s [1], which is applied to solve the tracking problem of a given task in finite time interval repeatedly. Since then, ILC has attracted a lot of attention due to its simple structure and effective performance.

☆ This work is supported by National Natural Science Foundation of China (61673045, 61304085), Beijing Natural Science Foundation (4152040).

* Corresponding author.

E-mail addresses: J.Han@uva.nl (J. Han), shendong@mail.buct.edu.cn (D. Shen), cjc@cc.hfu.edu.tw (C.-J. Chien).

https://doi.org/10.1016/j.jfranklin.2018.03.008

A detailed review was given in [2] from both theoretical and practical perspectives, including a categorization on ILC research from 1998 to 2004. The review article [3] presented a unified formulation of ILC, repetitive control and run-to-run control, also analyzed the similarities and differences.

The conventional ILC is designed to track the entire trajectory in a given time interval. However, in many practical scenarios, maybe only the terminal point of the output is claimed to be accurately tracked. As an example, taking the basketball shooting from a fixed position into consideration, the player only cares whether the basketball hits the basket, rather than whether the basketball follows some appointed trajectory. Another illustration is the train operation where the driver expects better accuracy on the arrival position than the train running process among stations. There are two primary characteristics in such kind of systems. The first is that only the last state or output is measured, thus only the tracking error at the last position can be used for updating the input signal. The other is that only the terminal state or output instead of the whole trajectory is selected as the control objective. ILC designed for these systems is called terminal ILC (TILC).

After TILC was applied in RTPCVD (rapid thermal processing chemical vapor deposition) thickness control [4], it has been extensively exploited. The train stop control is a typical terminal control where only the final train stop position is of major concern. The TILC was introduced to utilize the terminal stop position error in previous braking process to update the control profile [5]. Data-driven method has shown its effectiveness in TILC. A data-driven optimal TILC approach was provided in [6] for both linear and nonlinear discrete-time systems. Detailed proofs were formulated to show the convergence of the proposed algorithms. Then data-driven TILC was improved with time-varying inputs [7]. Furthermore, a dynamical linearization compensation was utilized to release the constraint of initial condition in every iteration [8]. The key component of a data-driven method is the partial derivative estimate law along the iteration axis, while in this paper the partial derivative estimation does not need to be updated iteratively. Additionally, the data-driven method was also demonstrated as effective in train station control [9]. A similar estimation algorithm is designed to update the system Markov parameters for linear systems [10]. In [11], a parameterized model of a linear time-varying system using shifted Legendre polynomials approximation was first provided and then the TILC problem was discussed by adjusting the parameters with the help of terminal tracking error information. The high-order case of TILC was dealt with in [12] where a genetic algorithm was utilized to find parameters of the updating law to obtain a good robustness.

TILC could be considered as a special case of point-to-point ILC (P2PILC). The control objective of P2PILC is to track part critical points of the output instead of the entire trajectory. The multiple P2PILC scheme was achieved by updating the reference iteratively between trials instead of input profile [13]. Hard and soft constraints have been added into the control profile to improve the performance while addressing the practical situation [14,15]. A new P2PILC structure based on successive projection is provided in [16] and experimentation on a robotic arm is given to shown the effectiveness.

In this paper, a neural network (NN) is introduced to resolve TILC problem for general nonlinear systems. The NN-based control approach has been shown effective in nonlinear approximation problem and the parameter estimating problem. As revealed in [17], NN-based ILC had remarkable performance for nonlinear systems. Radial basis function (RBF) NN was used to construct the controller of non-affine nonlinear systems. Besides, RBF could also be applied to approximate the effect of initial state on the terminal output [18]. Several adaptive TILC have been proposed to relax the constraints of identical conditions on the initial states

and target trajectories [19–21]. However, previous NN-based TILC papers kept the input constant in the operation interval and failed to consider the control performance with time-varying inputs. In order to obtain better performance in theoretical and practical situations, NN-based TILC with time-varying input is presented by this paper. A primary version of this paper was reported in [22]. However, the initial value was assumed to be precisely reset in the conference paper. Moreover, a unique precise expression of the desired input was assumed to exist; that is, the approximation error was absent for the desired input. In this paper, both limitations are removed so that we consider a general formulation of the problem.

To be specific, the NN-based TILC problem for nonlinear discrete-time systems is addressed in this paper. Instead of approximating the system itself, a RBFNN is introduced to approximate the nonlinear controller directly, in which the input and output are the terminal output target and control signal for the controlled system, respectively. A quadratic index is used to generate the recursive estimation of the weights of RBFNN. Both time-invariant input and time-varying inputs are considered in this paper. Compared with the traditional ILC method, the proposed NN-based method is more suitable for nonlinear systems. The traditional ILC usually uses a P-type learning update law, while the learning gain is hard to tune without knowing prior knowledge. The proposed NN-based method provides a robust compensation for the unknown nonlinearity and thus could ensure a bounded convergence of tracking error.

The rest of the paper is arranged as follows: Section 2 presents the NN-based TILC algorithm with time-invariant input and its convergence analysis. In Section 3, the NN-based TILC with time-varying inputs and convergence analysis are provided. Illustrative simulations for both cases are given in Section 4. Section 5 concludes this paper. The detailed proofs to the main theorems are put in the Appendix.

2. Neural networks based TILC with time-invariant input

2.1. Problem formulation

Consider the following non-affine nonlinear discrete-time system

$$y_k(t+1) = f(y_k(t), \ldots, y_k(t-n+1), u_k) \tag{1}$$

where $t = 0, 1, \ldots, N$ denotes the time instance of each iteration, N is the length of one iteration, and k denotes the iteration number. $y_k(t)$ and u_k are the system output and input, respectively. $f(y_k(t), \ldots, y_k(t-n+1), u_k)$ is an unknown real continuous nonlinear function of $y_k(t), y_k(t-1), \ldots, y_k(t-n+1)$ and u_k, where n is a positive integer denoting the output delay order. In this paper, we consider a finite response model for simplicity. Thus, we assume $y_k(t) = 0$ when $t < 0$. Besides, only the terminal output, i.e., $y_k(N)$ could be measured for updating the input. The input u_k would keep constant in the same iteration. The nonlinear system can be further expressed by rewriting as follows.

$$\begin{aligned}
y_k(1) &= f(y_k(0), \ldots, y_k(-n+1), u_k) = g_1(y_k(0), u_k) \\
y_k(2) &= f(y_k(1), y_k(0), \ldots, y_k(-n+2), u_k) \\
&= f(g_1(y_k(0), u_k), y_k(0), \ldots, y_k(-n+2), u_k) \\
&= g_2(y_k(0), u_k) \\
&\vdots
\end{aligned}$$

$$\begin{aligned} y_k(N) &= f(y_k(N-1), \ldots, y_k(N-n+1), u_k) \\ &= f(g_{N-1}(y_k(0), u_k), \ldots, g_{N-n+1}(y_k(0), u_k), u_k) \\ &= g_N(y_k(0), u_k). \end{aligned}$$

The control objective is to find a control input sequence $\{u_k\}$ such that the terminal output $y_k(N)$ can track the desired value $y_d(N)$ as the iteration number k goes to infinity. Without causing misunderstandings, the argument t (the time instant label) may be omitted in the rest of the note.

The following assumptions are needed for the algorithm.

A1. Denote $\varphi(y_k(0), u_k) = \partial g_N(y_k(0), u_k)/\partial u_k$, and $\varphi(y_k(0), u_k)$ is continuous and sign-unchanged. It is assumed that there are positive real numbers φ_l and φ_u such that $\varphi_l < \varphi(y_k(0), u_k) < \varphi_u$, $y_k(0) \in \mathbb{R}$, $u_k \in \mathbb{R}$.

A2. The initial system output in each iteration $y_k(0)$ is close to the desired initial output. That is, $|y_k(0) - y_d(0)| < \alpha$, where $\alpha > 0$ is a small constant and $y_d(0)$ is the desired initial output.

A3. The terminal desired value is realizable, i.e., there exists unknown desired input u_d such that $g_N(y_d(0), u_d) = y_d(N)$.

Remark 1. Assumption A1 implies that the control direction is known prior. Otherwise, the techniques handling the unknown control direction such as Nussbaum gain [23] may be applied. Assumption A2 is a relaxation of the conventional identical initialization condition. In A2, the initial output can vary in a small range around the desired initial position. Assumption A3 is an existence condition on the desired input. If this assumption is not valid (i.e., no suitable input exists in generating the tracking reference), the precise tracking problem can be formulated as an optimization problem.

2.2. Design of the neural networks based TILC with time-invariant input

Due to the existence of nonlinearity, it is impossible to get the inverse form of unknown desired input u_d precisely. In this paper, a RBFNN is introduced for designing the input u_k so as to approximate the unknown u_d. To be specific, a three layers NN is used. Denote the i-th RBF of the hidden layer as follows.

$$G_i(\chi) = \exp\left(-\sum_{j=1}^{N+1} \frac{\|\chi_j - a_i\|^2}{2c_i^2}\right), \quad i = 1, 2, \ldots, m \tag{2}$$

where m denotes number of nodes of hidden layer, a_i and c_i are the center and radius of the ith node, respectively. χ is the input of the network and it has the same dimension as the time interval of every iteration. χ_j denotes the jth element of χ. In this paper, only the terminal output is available for updating the weighted parameters. It is seen that such information is not sufficient for training the NN, thus we introduce a man-made output trajectory with the same desired terminal output to tune the parameters in the following. In other words, χ is a vector with the final point being the desired output while the left elements are virtual values. For instance, let the desired terminal output be $y_d(N)$ and we can construct a virtual trajectory $y_r(t)$, $t = 0, 1, \ldots, N$, with $y_r(N) = y_d(N)$. Then the input χ is a vector $\chi = [y_r(0), y_r(1), \ldots, y_r(N)]^T$. Thus the output of the hidden layer could be denoted as $G(\chi) = [G_1(\chi), G_2(\chi), \ldots, G_m(\chi)]^T$. The output of the RBFNN is a weighted summation of all hidden nodes, i.e.,

$O_j = \sum_{i=1}^{m} W_{ij} G_i(\chi) = W_j^T G(\chi)$, where W_{ij} denotes the weights of the ith hidden node to the output O_j and W_j is the associated weight vector. We note that the RBFNN is constructed to approximate the inverse mapping of $g_N(\cdot)$.

Remark 2. For the unknown desired input u_d, there exists an unknown desired weight $W_d \in \mathbb{R}^{m\times 1}$ such that $u_d = W_d^T G(\chi) + \varepsilon(\chi)$, in which $\varepsilon(\chi)$ denotes the approximation error with the property of $|\varepsilon(\chi)| \le \varepsilon_m$ for some bounded positive constant ε_m and for all χ belongs to a compact set. As is well known, the error $\varepsilon(\chi)$ could be sufficiently small by selecting enough nodes because of the universal approximation property of NNs. The selection of suitable centers a_i and spreads c_i also benefits the reduction of the approximation error. Thus, the selection of these parameters should involve the specific system formulation. A designed trajectory with the desired terminal output $y_d(N)$ would be used as the input of the proposed RBFNN.

However, the unknown desired input $u_d = W_d^T G(\chi) + \varepsilon(\chi)$ and the corresponding unknown desired weights W_d are not available for TILC design. As a result, the control for the original system (14) is formulated as follows.

$$u_k = W_k^T G(\chi) \tag{3}$$

where W_k is the weight vector for the kth iteration to be tuned during the ILC process and $W_k = [W_{1k}, W_{2k}, \ldots W_{mk}]^T$. Note that the input is a desired trajectory, which does not change along iteration axis, thus the output of the hidden layer $G(\chi)$ is static along the iteration axis.

The terminal tracking error is denoted by

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

Then according to A3 and by differential mean value theorem, we have

$$\begin{aligned} e_k(N) &= y_d(N) - y_k(N) \\ &= \varphi_k(u_d - u_k) + \varphi_{y,k}(y_d(0) - y_k(0)) \\ &= \varphi_k(W_d - W_k)^T G(\chi) + \varphi_k \varepsilon'(\chi) \end{aligned} \tag{5}$$

where $\varphi_k = \partial g_N(y_k(0), \bar{u}_k)/\partial \bar{u}_k$ with $\bar{u}_k$ locating between u_k and u_d, while $\varphi_{y,k} = \partial y_N/\partial \bar{y}_k$ with $\bar{y}_k$ lying between $y_d(0)$ and $y_k(0)$. It is noticed that φ_k is unknown prior, thus one would like to estimate it for control updating. In the following, it is replaced by a tuning parameter for learning algorithm. In addition, the approximation error and initialization error are both included in $\varepsilon'(\chi)$, that is, $\varepsilon'(\chi) = \varepsilon(\chi) + \varphi_k^{-1} \partial g_N/\partial \bar{y}_k (y_d(0) - y_k(0))$.

Define the following objective function

$$J = \|e_k(N)\|^2 + \lambda \|W_{k+1} - W_k\|^2 \tag{6}$$

where $0 < \lambda < 1$ is a weighting factor. Then one can derive the following update for W_k by minimizing the objective function J with respect to W_k,

$$W_{k+1} = W_k + \frac{\varphi_k G(\chi)}{\lambda} e_k(N). \tag{7}$$

We make a normalization step to Eq. (7) and replace the iteration-dependent derivative, then Eq. (7) is modified as

$$W_{k+1} = W_k + \eta \frac{\varphi G(\chi)}{\lambda + \varphi \|G(\chi)\|^2} e_k(N) \tag{8}$$

where φ is a prior setting parameter, which plays the role as a prior estimation of φ_k, and η is the learning gain.

Remark 3. The updating law (7) is derived by minimizing the objective function (6) where a real partial derivative is required to update the parameters. However, the derivative is not always easy to compute precisely. If the derivative is not available, an approximation of the derivative is used in the updating law to reduce the computation burden. This motivates us to use an estimated value of partial derivative instead of estimating it with an recursive algorithm, while the robustness and convergence properties are well retained. The denominator of Eq. (7) is also replaced by $\lambda + \varphi\|G(\chi)\|^2$ as a normalization step.

In order to compensate the approximation error generated by NN, we now define a dead-zone like auxiliary error function

$$e_k^{\phi}(N) = e_k(N) - \phi_k sat\left(\frac{e_k(N)}{\phi_k}\right) \tag{9}$$

where

$$sat\left(\frac{e_k(N)}{\phi_k}\right) = \begin{cases} -1, & e_k(N) < -\phi_k \\ \dfrac{e_k(N)}{\phi_k}, & |e_k(N)| \le \phi_k \\ 1, & e_k(N) > \phi_k \end{cases}. \tag{10}$$

It is noted that $e_k^{\phi}(N)$ can be rewritten as

$$e_k^{\phi}(N) = \begin{cases} e_k(N) + \phi_k, & e_k(N) < -\phi_k \\ 0, & |e_k(N)| \le \phi_k \\ e_k(N) - \phi_k, & e_k(N) > \phi_k \end{cases}. \tag{11}$$

Here, ϕ_k will be used to compensate the approximation error and the initialization error. It is seen that $\phi_k = \varphi_k \varepsilon'(\chi)$, which is product of two terms, namely, φ_k and $\varepsilon'(\chi)$. The former term has an upper bound φ_u according to A1, while the latter term is assumed to be bounded by ε'_m according to Remark 2 and A2. Therefore, we can conclude that ϕ_k has an unknown bound $\varphi_u \varepsilon'_m$.

Multiplying both sides of Eq. (9) with $e_k^{\phi}(N)$, we find that $(e_k^{\phi}(N))^2 = e_k(N)e_k^{\phi}(N) - \phi_k|e_k^{\phi}(N)|$, in which $e_k^{\phi}(N)sat(\frac{e_k(N)}{\phi_k}) = |e_k^{\phi}(N)|$. The auxiliary error function is then utilized to design adaptive laws for W_k and ϕ_k . The adaptive laws are given as follows

$$W_{k+1} = W_k + \eta_1 \frac{\varphi G(\chi)}{\lambda + \varphi\|G(\chi)\|^2} e_k^{\phi}(N) \tag{12}$$

$$\phi_{k+1} = \phi_k + \eta_2 \frac{\varphi\|G(\chi)\|^2}{\lambda + \varphi\|G(\chi)\|^2} \left| e_k^{\phi}(N) \right| \tag{13}$$

where η_1, η_2 are step sizes.

2.3. Convergence analysis

Theorem 1. *Consider the nonlinear discrete-time system (1) and assume A1–A3 hold. Select step sizes* η_1, η_2 *such that* $2\text{-}\varphi_u\eta_1\text{-}\eta_2 > 0$, *then the ILC algorithm (3) with parameters update*

laws (12) and (13) guarantees that the actual error $e_k(N)$ is bounded, $\lim_{k\to\infty} |e_k(N)| \le \phi_\infty$, *and the auxiliary error $e_k^{\phi}(N)$ converges to zero as iteration number goes to infinity.*

The detailed proof is in the appendix

Remark 4. The theorem provides stability analysis of the NN-based TILC algorithms for a nonlinear discrete-time system with constant input value. The final tracking error bound depends on the approximation error, which is related to selection and parameters setting of RBFNNs. It is noted that the approximation error is essential; in other words, it is difficult to further reduce this error by the designed algorithms.

3. Neural networks based TILC with time-varying inputs

3.1. Problem formulation

Consider the following non-affine nonlinear discrete-time system

$$y_k(t+1) = f(y_k(t), \ldots, y_k(t-n+1), u_k(t)) \tag{14}$$

where the constant input u_k in Eq. (1) is replaced by a time-varying input $u_k(t)$. Similar to the constant input case, we assume $y_k(t) = 0$ when $t < 0$. Then, the nonlinear system can be further expressed by rewriting as follows.

$$\begin{aligned}
y_k(1) &= f(y_k(0), \ldots, y_k(-n+1), u_k(0)) = g_1(y_k(0), u_k(0)) \\
y_k(2) &= f(y_k(1), y_k(0), \ldots, y_k(-n+2), u_k(1)) \\
&= f(g_1(y_k(0), u_k(0)), y_k(0), \ldots, y_k(-n+2), u_k(1)) \\
&= g_2(y_k(0), u_k(0), u_k(1)) \\
&\vdots \\
y_k(N) &= f(y_k(N-1), \ldots, y_k(N-n+1), u_k(N-1)) \\
&= f(g_{N-1}(y_k(0), \mathbf{u}_k(N-1)), \ldots, g_{N-n+1}(y_k(0), \mathbf{u}_k(\cdot)), u_k(N-1)) \\
&= g_N(y_k(0), \mathbf{u}_k(N))
\end{aligned}$$

where the vector input $\mathbf{u}_k(t)$ is defined as $\mathbf{u}_k(t) = [u_k(0), \ldots, u_k(t-1)]^T$. In particular, the input $\mathbf{u}_k(N)$ is formed as

$$\mathbf{u}_k(N) = [u_k(0), u_k(1), \ldots u_k(N-1)]^T. \tag{15}$$

In last section, the input is simply assumed to be a constant in each iteration. To further enlarge the application of NN in TILC problems, the time-varying input case is discussed in this section.

3.2. Design of the neural networks based TILC with time-varying inputs

The terminal tracking error can be derived as

$$\begin{aligned}
e_k(N) &= y_d(N) - y_k(N) \\
&= g_N(y_d(0), \mathbf{u}_d(N)) - g_N(y_k(0), \mathbf{u}_k(N)) \\
&= \Phi_k^T(\mathbf{u}_d(N) - \mathbf{u}_k(N)) + \partial g_N / \partial \bar{y}_k (y_d(0) - y_k(0))
\end{aligned} \tag{16}$$

where $\Phi_k = [\Phi_k(0), \dots \Phi_k(N)]^T$, $\Phi_k(i) = \partial g_N(y_k(0), \mathbf{u})/\partial u_k(i)|_{\mathbf{u}=\mathbf{u}_k(N)}$, $0 \le i \le N$. In addition, $\mathbf{u}_d(N)$ denotes the corresponding desired input vector for $y_d(N)$ defined similar to $\mathbf{u}_k(N)$.

Assumption A1 is replaced by the following one.

A4. Each component of the partial derivative vector Φ_k does not change its sign. Without loss of any generality, it is assumed that there exist unknown positive constants Φ_L and Φ_U such that $\Phi_L < \Phi_k(i) < \Phi_U$, where $\Phi_k(i)$ denotes the ith component of Φ_k.

Remark 5. Assumption A4 is a counterpart of A1 for the time-varying input case. In this case, the derivative of the desired terminal output with respect to the time-varying input is a vector rather than a scalar variable. Thus, we impose a general assumption in A4 for the control direction.

The input is given as

$$\mathbf{u}_k = W_k G(\chi) \tag{17}$$

where $W_k \in \mathbb{R}^{N\times m}$ is the weight matrix throughout this section and m is the number of hidden nodes. Similar to the update law for weights in Eq. (8), the following updating could be derived for the time-varying input case,

$$W_{k+1} = W_k + \gamma \frac{\Phi G(\chi)^T}{\mu + \|\Phi\|^2 \|G(\chi)\|^2} e_k(N) \tag{18}$$

where γ is a learning gain and μ is a positive tuning parameter. Since Φ_k is unknown, here Φ is adopted as a prior estimation of Φ_k. The final tracking error $e_k(N)$ can be expressed as

$$\begin{aligned} e_k(N) &= y_d(N) - y_k(N) \\ &= \Phi_k^T (W_d - W_k) G(\chi) + \sigma(\chi) \end{aligned} \tag{19}$$

where $\sigma(\chi)$ consists of two parts, the neural network approximation error and the initialization error $\partial g_N/\partial \bar{y}_k (y_d(0) - y_k(0))$. The approximation error comes from the definition of $\mathbf{u}_d$ similar to the constant input case. Specifically, we assume $\mathbf{u}_d = W_d G(\chi) + \xi(\chi)$ with W_d and $\xi(\chi)$ being a matrix and a vector. Therefore, $\sigma(\chi) = \Phi_k^T \xi(\chi) + \partial g_N/\partial \bar{y}_k (y_d(0) - y_k(0))$.

Similar to the time-invariant input case, an auxiliary error function is defined as follows.

$$e_k^\sigma(N) = e_k(N) - |\sigma_k| sat\left(\frac{e_k(N)}{|\sigma_k|}\right) \tag{20}$$

where

$$sat\left(\frac{e_k(N)}{|\sigma_k|}\right) = \begin{cases} -1, & e_k(N) < -\sigma_k \\ \dfrac{e_k(N)}{|\sigma_k|}, & |e_k(N)| \le \sigma_k \\ 1, & e_k(N) > \sigma_k \end{cases}. \tag{21}$$

It is noted that $e_k^\sigma(N)$ can be rewritten as

$$e_k^\sigma(N) = \begin{cases} e_k(N) + |\sigma_k|, & e_k(N) < -|\sigma_k| \\ 0, & |e_k(N)| \le |\sigma_k| \\ e_k(N) - |\sigma_k|, & e_k(N) > |\sigma_k| \end{cases}. \tag{22}$$

Multiplying both sides of Eq. (22) with e_k^σ leads to

$$[e_k^\sigma(N)]^2 = e_k(N)e_k^\sigma(N) - \mathrm{sgn}(\sigma_k)\sigma_k\left|e_k^\sigma(N)\right|. \tag{23}$$

Define a new variable $\nu_k = -\sigma_k + y_k - \Phi^T\mathbf{u}_k(N)$. The purpose of defining ν_k is to update σ_k substantially. The auxiliary error function is then utilized to design the adaptive laws for W_k and ν_k. The adaptive laws are given as follows,

$$W_{k+1} = W_k + \gamma_1\frac{\Phi G(\chi)^T}{\mu + \|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma(N) \tag{24}$$

$$\nu_{k+1} = \nu_k + \gamma_2\frac{\|G(\chi)\|^2}{\mu + \|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma(N). \tag{25}$$

Remark 6. It is easy to find out that ν_k is a combination of approximation error from the NN and tracking deviation caused by the estimated value of partial derivative. Therefore, we can update ν_k to compensate both errors. When ν_k is available, σ_k is then calculated by $\sigma_k = -\nu_k + y_k - \Phi^T\mathbf{u}_k(N)$.

3.3. Convergence analysis

Theorem 2. *Consider the nonlinear discrete-time system (14), and assume A2-A4 hold. Select step-sizes γ_1, γ_2 such that*

$$2 - \gamma_1\|\Phi\|^2\frac{\|G(\chi)\|^2}{\mu + \|\Phi\|^2\|G(\chi)\|^2} - \gamma_2\frac{\|G(\chi)\|^2}{\mu + \|\Phi\|^2\|G(\chi)\|^2} > 0,$$

then the ILC algorithm (17) with parameter update laws (24) and (25) guarantee that the $e_k(N)$ is bounded, $\lim_{k\to\infty}|e_k(N)| \le |\sigma_\infty|$, and $e_k^\sigma(N)$ converges to zero as the iteration number goes to infinity.

The proof can be found in the appendix. Note that $G(\chi)$ can be determined by selecting suitable neural network while Φ is a prior estimation of the unknown iteration-varying derivatives. Thus the condition given in this theorem is valid as long as the learning parameters γ_1 and γ_2 are sufficiently small. This condition can be regarded as a guideline for the selection of learning parameters. Noting that both $G(\chi)$ and Φ are available, we can make a rough calculation of γ_1 and γ_2. For example, we could let $\gamma_1 < (\|\Phi\|^2\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2})^{-1}$, $\gamma_2 < (\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2})^{-1}$.

Remark 7. The theorem gives an extension to time-varying input case, where the vector updating is proposed with a Lyapunov-method-based stability analysis. Compared with the time invariant input case, the time-varying input case provides more freedom for the control design. Meanwhile, in this case, the proposed algorithm can be further enlarged its applicability according to practical conditions and requirements.

Remark 8. Comparing the convergence conditions in Theorems 1 and 2, we have the following observations. First, both theorems give an inequality to the learning step sizes depicting a range of the parameters selection for practical applications. Second, we can observe from the proofs in the Appendix that the condition in Theorem 1 is stricter but more concise than that in Theorem 2 (see κ_1 in the Appendix). Final, the difference between the two conditions

lies in that the time-varying input case (Theorem 2) involves several vectors while the time-invariant input case (Theorem 1) mainly addresses constant variables. Thus, the former case requires a more complex condition formulation than the latter case.

4. Simulation

4.1. Simulation based on train station stop control

In this subsection, the effectiveness of the proposed NN-based TILC is demonstrated through simulation of train station stop control [5]. The input of the system could be braking force, braking position or their combination and the output is the final position of the train. The controlled train system is described by

$$\begin{cases} ds/dv = v/(-F(s^{v_0}) - w(v) - g(s)) \\ w(v) = \left(12.446{+}0.5468v + 0.0185v^2\right) \times 10^{-3} \end{cases} \tag{26}$$

where $F(s^{v_0})$ is the braking force on unit mass at the position s with velocity v_0 in the kth braking process and $w(v)$ the general resistance on unit mass at the speed v. The general resistance $w(v) = av^2/m + bv + c$ consists of three parts, aerodynamic drag av^2/m, mechanical drag bv, rolling frictional force c, in which m denotes the train's weight. $g(s)$ is the additional resistance on unit mass at the position s.

$$g(s) = \begin{cases} (s+600) \times 0.00012/100, & -600 \le s \le -500 \\ 0.00012, & -500 \le s \le -300 \\ (-s-200) \times 0.00012/100, & -300 \le s \le -200 \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

The initial braking position is set to be -785 m and the braking force is 1 N/kg at the first iteration. The initial speed v_0 is set to be 40 m/s. The ith output of activation function used in the RBFNN is $G_i(\chi) = \exp(-\sum \|\chi_j - a_i\|^2/2{c_i}^2)$, $i = 1, 2 \ldots, m$, where $m = 15$. The parameters for neurons are selected randomly obeying a uniform distribution; that is, a_i is uniformly distributed on $[0.9, 1]$ and c_i is uniformly distributed on $[0.84, 0.85]$, $1 \le i \le 15$, respectively. Due to the fact that the desired terminal output is only one signal and not enough to train the NN, so the input of NN is designed to be $y_d = 1 - \cos(\pi t/5)$, $t = 1, 2 \ldots, 10$. The partial derivative prior estimation is given as $\varphi = 0.6$. In practical applications, the initial conditions may have certain degree of randomness. To simulate this point, we further add Gaussian random disturbances (zero mean and standard deviation 0.1) to the initial speed and initial position at each iteration. We also note that the parameter selection is problem-dependent that in this simulation we provide the above parameters as they are already sufficient for well performance. If the system is more complex and nonlinear, it is suggested to disperse the parameter to gain a good approximation of the nonlinearities and introduce some swarm intelligence methods to tune the parameters.

4.1.1. NNTILC with initial braking position as time-invariant input

In this section, the braking position is used as input for the system and the braking force keeps constant in the iteration. In this case, the tuning parameters for Eqs. (12) and (13) are set as $\lambda = 0.0005$, $\eta_1 = 0.4$, $\eta_2 = 0.0001$ and the initial value of ϕ is 0.0001. The initial value of the weight of NN is uniformly distributed on $[-59500, -59499]$. The algorithm is run for

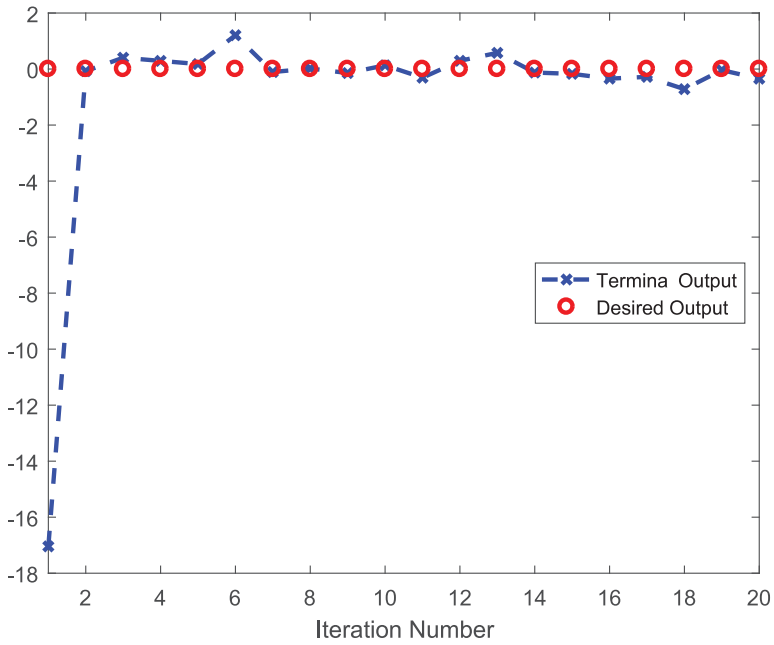

Fig. 1. Terminal output vs. desired output along iteration axis.

30 iterations. Fig. 1 shows the actual terminal output (crosses "$\times$") could track the desired terminal output (cycles) after several iterations. This fact shows that the time-invariant input can ensure certain asymptotical convergence along the iteration axis under suitable conditions. In Section 2, we have introduced an auxiliary error to compensate the approximation error, of which the control objective is to ensure the zero-error convergence of this auxiliary error. This point is verified in Fig. 2 where a faster convergence can be observed. The lower subfigure of Fig. 2 demonstrates that algorithms (12) and (13) could guarantee that the input value converges after a few iterations.

4.1.2. NNTILC with initial braking force as time-invariant input

In this case, the braking force is utilized as input for the system and the braking position keeps constant in the iteration. The initial settings of NN keep the same. While the tuning parameters for (12) and (13) are set as $\lambda = 0.0005$, $\eta_1 = -0.00018$, $\eta_2 = -0.0001$ and the initial value of ϕ is 0.0001. The initial value of the weight of NN is uniformly distributed on $[75, 76]$. Fig. 3 shows that the terminal output can also track the desired terminal position by iteratively learning the initial braking force. This convergence coincides with the theoretical analysis. On the other hand, results are given in Fig. 4 as similar to Fig. 2, in which the convergence of the auxiliary error and the input signal are plotted in two subfigures. In addition, the convergence speed in both cases can be tuned by selecting the learning parameters.

4.2. Simulation based on batch reactor

In this subsection, in order to demonstrate the effectiveness of the proposed algorithm with time-varying input. A nonlinear batch reactor with temperature as the control variable is considered [24–26]. The reaction process is $P_1 \xrightarrow{k_1} P_2 \xrightarrow{k_2} P_3$, in which P_1 is the raw material,

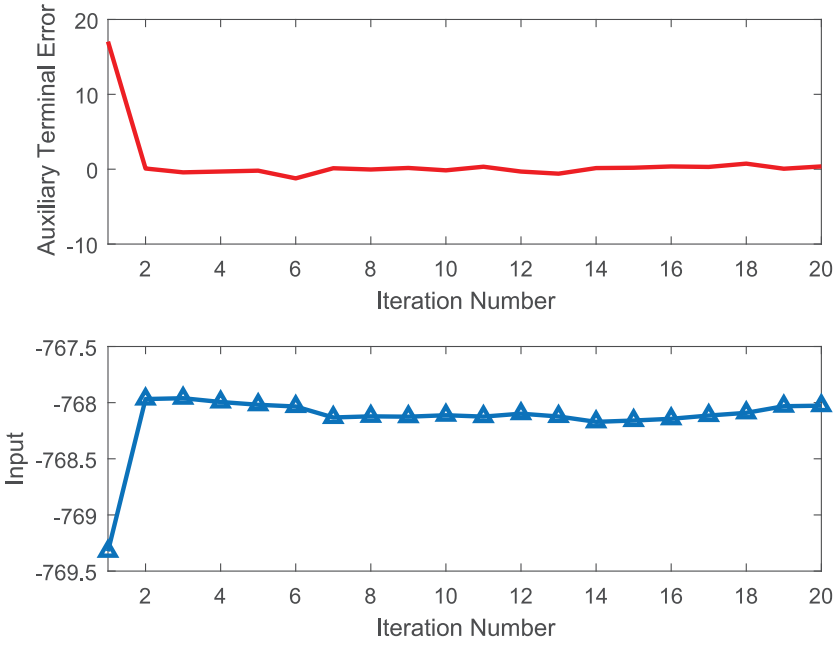

Fig. 2. Auxiliary terminal tracking error and input along iteration axis.

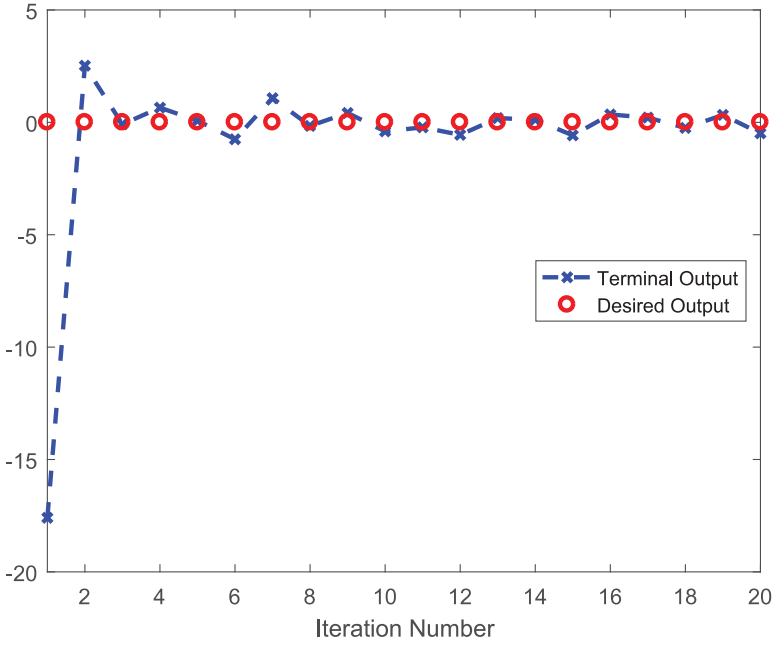

Fig. 3. Terminal output vs. desired output along iteration axis.

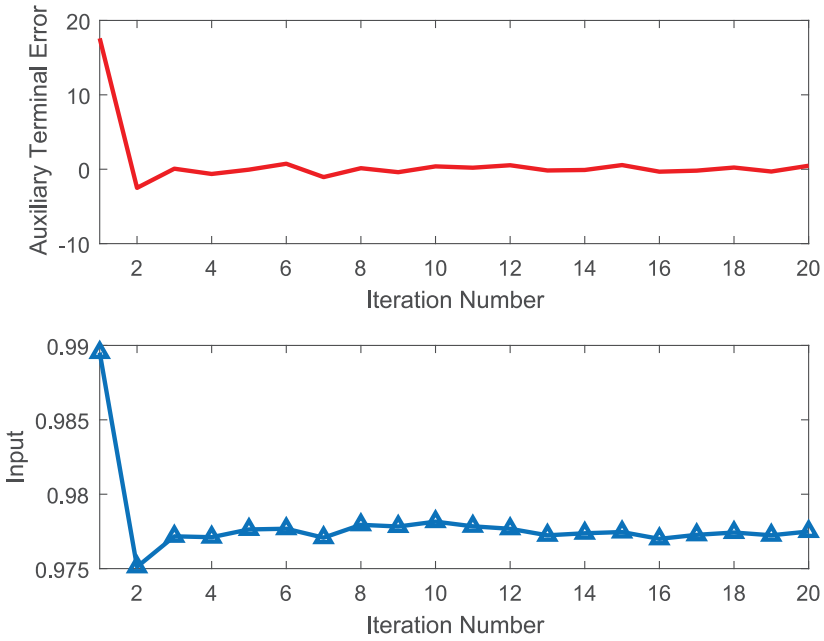

Fig. 4. Auxiliary terminal tracking error and input along iteration axis.

P_2 is the product, and P_3 is the by-product. The differential equations describing the reactor have been tested and validated by real-time data and generally applied for the simulation of the real plant.

$$\begin{cases} \dot{x}_1 = -k_1 \exp\left(\dfrac{-E_1}{uT_{ref}}\right) x_1^2 \\ \dot{x}_2 = k_1 \exp\left(\dfrac{-E_1}{uT_{ref}}\right) x_1^2 - k_2 \exp\left(\dfrac{-E_1}{uT_{ref}}\right) x_2 \\ y = x_2 \end{cases} \tag{28}$$

where $k_1 = 4.0 \times 10^3$ K, $k_2 = 6.2 \times 10^5$ K, $E_1 = 2.5 \times 10^3$ K, $E_2 = 5.0 \times 10^3$ K and $T_{ref} = 348$ K. x_1 and x_2 represent the dimensionless concentrations of the raw material and the product respectively; $u = T/T_{ref}$ is the dimensionless temperature of the reactor and T_{ref} is the reference temperature; the final time t_f is fixed to be 1.0 and the sampling time $h = t_f/N = 0.1$ with $N = 10$. The control objective is to maximize the amount of the product P_2 after a fixed reaction time. For this case, the tuning parameters for (24) and (25) are set as $\mu = 0.001$, $\gamma_1 = 0.2$, $\gamma_2 = -0.008$, $\Phi = [0.28, 0.27, 0.26, 0.25, 0.24, 0.23, 0.22, 0.21, 0.2, 0.19]$ and the initial value of σ and ν is 0.0001, -0.15, respectively. The parameters for neurons are randomly selected within some interval; that is, a_i is uniformly distributed on $[0.3, 0.4]$ and c_i is uniformly distributed on $[1.2, 1.3]$. The initial value of the weight of NN is uniformly distributed on $[0.4, 0.55]$. Due to the fact that the desired terminal output is only one signal and not enough to train the neural network, so the input of neural network is set to be $y_d = 0.61 \cos(\pi t/5)$, $t = 1, 2 \ldots, 10$. Fig. 5 demonstrates the convergence of the terminal output to the desired terminal output along the iteration axis. It is seen that the bounded convergence can be achieved after several iterations. We further plot the auxiliary tracking error in the upper part of Fig. 6 to show its zero-error convergence. Note that the time-varying

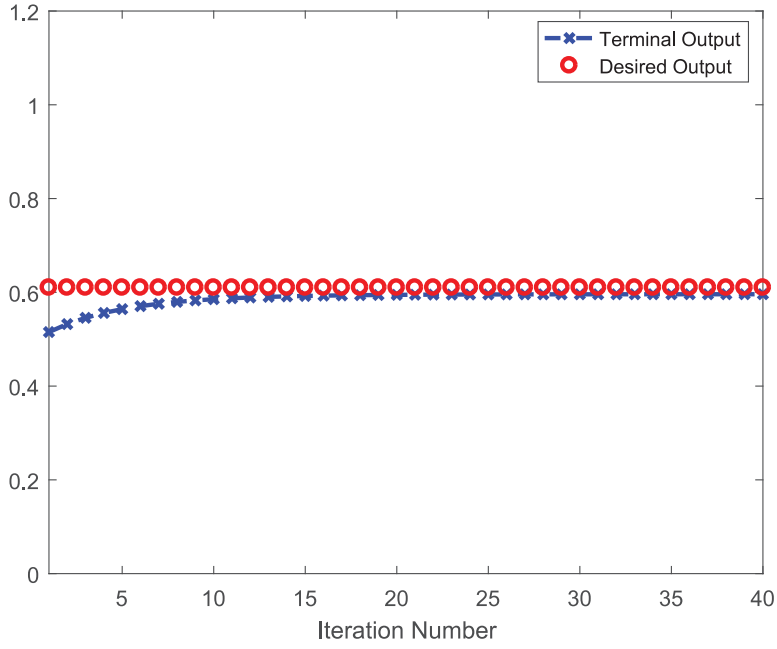

Fig. 5. Terminal output vs desired output along iteration axis.

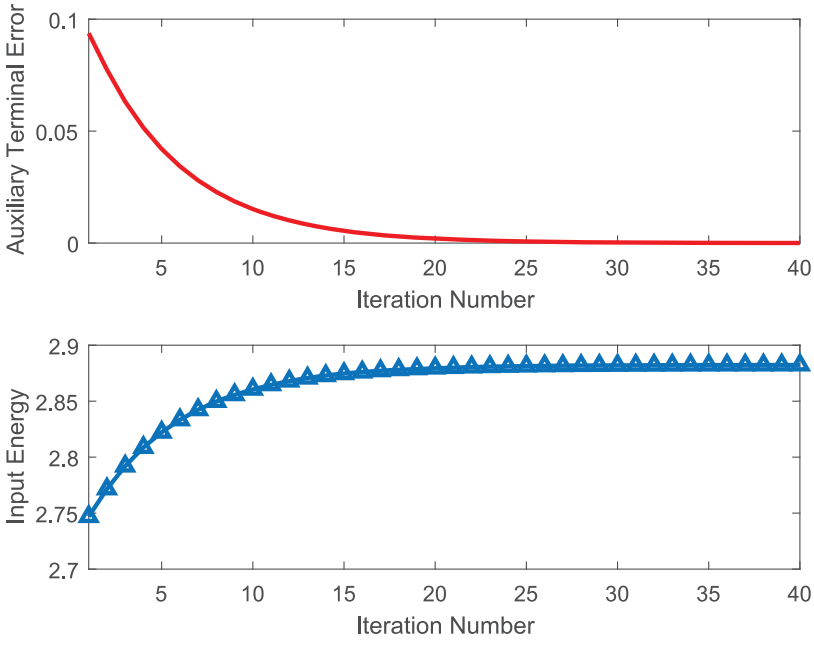

Fig. 6. Auxiliary terminal tracking error and input energy along iteration axis.

input case is considered in this simulation, thus we plot the input energy of each iteration, defined as $\|\mathbf{u}_k(N)\|$, rather than the input profiles to verify the convergence of input. This convergence is shown in the lower part of Fig. 6.

5. Conclusions

The NN-based TILC algorithm for discrete-time nonlinear systems is considered in this paper. In the proposed algorithm, an RBFNN is introduced as function approximation of the input signal corresponding to the desired tracking target. A dead-zone like auxiliary error is then constructed to overcome the approximation error from NN and initialization derivation. The weights are updated by optimizing a given objective function and then the input is generated. Both time-invariant and time-varying input cases are discussed to demonstrate the performance of NNTILC. To ensure the stability and learning performance, the sufficient condition for step-sizes of update laws are provided. A Lyapunov-like analysis is presented to show the convergence properties. For further research, the general point-to-point control based on NN is of interest.

Appendix

Convergence analysis for Theorem 1

Proof. From Eqs. (3) and (12), the updating law of input could be rewritten as

$$u_{k+1} = u_k + \eta_1 \frac{\varphi\|G(\chi)\|^2}{\lambda + \varphi\|G(\chi)\|^2} e_k^{\phi}. \tag{29}$$

We first define the parameter errors as $\tilde{W}_k = W_d - W_k$, $\tilde{\phi}_k = \phi_u - \phi_k$. Deriving from Eq. (5), one can have that $\tilde{W}_k^T G(\chi) = e_k(N)/\varphi_k - \varepsilon(\chi)$. Subtracted both sides of Eqs. (12) and (13) from the optimal control gains, it is easy to have that

$$\tilde{W}_{k+1} = \tilde{W}_k - \eta_1 \frac{\varphi G(\chi)}{\lambda + \varphi\|G(\chi)\|^2} e_k^{\phi} \tag{30}$$

$$\tilde{\phi}_{k+1} = \tilde{\phi}_k - \eta_2 \frac{\varphi\|G(\chi)\|^2}{\lambda + \varphi\|G(\chi)\|^2} \left| e_k^{\phi} \right|. \tag{31}$$

Define a positive function V_k as $V_k = \frac{\varphi_u}{\eta_1}(\tilde{W}_k^T G(\chi))^2 + \frac{1}{\eta_2}\tilde{\phi}_k^2$, then the difference between V_k and V_{k+1} can be derived as follows

$$\begin{aligned} V_{k+1} - V_k &= -2\varphi_u \tilde{W}_k^T G(\chi) \frac{\varphi\|G(\chi)\|^2}{\lambda + \varphi\|G(\chi)\|^2} e_k^{\phi}(N) + \varphi_u \eta_1 \left(\frac{\varphi\|G(\chi)\|^2}{\lambda + \varphi\|G(\chi)\|^2} e_k^{\phi}(N) \right)^2 \\ &\quad - 2\tilde{\phi}_k \frac{\varphi\|G(\chi)\|^2}{\lambda + \varphi\|G(\chi)\|^2} \left| e_k^{\phi} \right| + \eta_2 \left[\frac{\varphi\|G(\chi)\|^2}{\lambda + \varphi\|G(\chi)\|^2} \left| e_k^{\phi} \right| \right]^2 \\ &= -2\frac{\varphi_u}{\varphi_k} \frac{\varphi\|G(\chi)\|^2}{\lambda + \varphi\|G(\chi)\|^2} e_k(N) e_k^{\phi}(N) + \varphi_u \eta_1 \left(\frac{\varphi\|G(\chi)\|^2}{\lambda + \varphi\|G(\chi)\|^2} e_k^{\phi}(N) \right)^2 \\ &\quad + 2\varphi_u \varepsilon(\chi) \frac{\varphi\|G(\chi)\|^2}{\lambda + \varphi\|G(\chi)\|^2} e_k^{\phi}(N) - 2\tilde{\phi}_k \frac{\varphi\|G(\chi)\|^2}{\lambda + \varphi\|G(\chi)\|^2} \left| e_k^{\phi}(N) \right| \end{aligned}$$

$$
\begin{aligned}
&+\eta_2\left[\frac{\varphi\|G(\chi)\|^2}{\lambda+\varphi\|G(\chi)\|^2}\left|e_k^{\phi}(N)\right|\right]^2\\
&\le -2\frac{\varphi\|G(\chi)\|^2}{\lambda+\varphi\|G(\chi)\|^2}\left[e_k^{\phi}(N)^2+\phi_k\left|e_k^{\phi}(N)\right|\right]+\varphi_u\eta_1\left(\frac{\varphi\|G(\chi)\|^2}{\lambda+\varphi\|G(\chi)\|^2}e_k^{\phi}(N)\right)^2\\
&+2\phi_u\frac{\varphi\|G(\chi)\|^2}{\lambda+\varphi\|G(\chi)\|^2}\left|e_k^{\phi}(N)\right|-2\tilde{\phi}_k\frac{\varphi\|G(\chi)\|^2}{\lambda+\varphi\|G(\chi)\|^2}\left|e_k^{\phi}(N)\right|\\
&+\eta_2\left[\frac{\varphi\|G(\chi)\|^2}{\lambda+\varphi\|G(\chi)\|^2}\left|e_k^{\phi}(N)\right|\right]^2\\
&=\left(-2+\varphi_u\eta_1\frac{\varphi\|G(\chi)\|^2}{\lambda+\varphi\|G(\chi)\|^2}+\eta_2\frac{\varphi\|G(\chi)\|^2}{\lambda+\varphi\|G(\chi)\|^2}\right)\frac{\varphi\|G(\chi)\|^2}{\lambda+\varphi\|G(\chi)\|^2}\left[e_k^{\phi}(N)\right]^2\\
&=-\kappa_1\left[e_k^{\phi}(N)\right]^2
\end{aligned}
$$

where $\kappa_1=2-\varphi_u\eta_1\frac{\varphi\|G(\chi)\|^2}{\lambda+\varphi\|G(\chi)\|^2}-\eta_2\frac{\varphi\|G(\chi)\|^2}{\lambda+\varphi\|G(\chi)\|^2}$.

In general, if there are η_1 and η_2 such that $2-\varphi_u\eta_1-\eta_2>0$, it is obvious that $\kappa_1>0$. This further leads that

$$\kappa_1\left[e_k^{\phi}(N)\right]^2\le V_k-V_{k+1} \tag{32}$$

therefore, $\forall n$

$$\sum_{k=0}^{n}\kappa_1\left[e_k^{\phi}(N)\right]^2\le\sum_{k=0}^{n}V_k-V_{k+1}=V_0-V_{n+1}<V_0. \tag{33}$$

Thus it follows that $e_k^{\phi}(N)\to 0$ as k goes to infinity. The boundedness of V_k at each iteration also ensures the boundedness of $\tilde{W}_k$, $\tilde{\phi}_k$. The boundedness of $e_k(N)$ at each iteration could be concluded by (11) because ϕ_k is always bounded. Furthermore, it will satisfy $\lim_{k\to\infty}|e_k(N)|\le|\phi_\infty|$. This completes the proof. □

Convergence analysis for Theorem 2

Proof. Define the parameter errors as $\tilde{u}_k=u_d-u_k$, $\tilde{\nu}_k=\nu_d-\nu_k$, where $\nu_d=y_d-\Phi^T u_d$. From Eqs. (24) and (25), it is easy to get

$$\tilde{u}_{k+1}=\tilde{u}_k-\gamma_1\frac{\Phi\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^{\sigma}(N) \tag{34}$$

$$\tilde{\nu}_{k+1}=\tilde{\nu}_k-\gamma_2\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^{\sigma}(N). \tag{35}$$

Define the Lyapunov function $V_k=\frac{1}{\gamma_1}\tilde{u}_k^T\tilde{u}_k+\frac{1}{\gamma_2}\tilde{\nu}_k^2$. It leads to

$$
\begin{aligned}
&V_{k+1}-V_k\\
&=-2\Phi^T\tilde{u}_k\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^{\sigma}+\gamma_1\|\Phi\|^2\left(\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^{\sigma}\right)^2
\end{aligned}
$$

$$
\begin{aligned}
&-2\left(\sigma_k+(\Phi_k-\Phi)^T\tilde{u}_k\right)\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma+\gamma_2\left(\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma\right)^2\\
&=-2e_k(N)\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma(N)-2\sigma_k\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma(N)\\
&\quad+\gamma_1\|\Phi\|^2\left(\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma(N)\right)^2+\gamma_2\left(\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma(N)\right)^2\\
&=-2\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}\left(e_k^\sigma(N)^2+\mathrm{sgn}(\sigma_k)\sigma_k\left|e_k^\sigma(N)\right|\right)-2\sigma_k\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma(N)\\
&\quad+\gamma_1\|\Phi\|^2\left(\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma(N)\right)^2+\gamma_2\left(\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma(N)\right)^2\\
&\le-2\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma(N)^2-2\sigma_k\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}\left(e_k^\sigma(N)+\mathrm{sgn}(\sigma_k)\left|e_k^\sigma(N)^2\right|\right)\\
&\quad+\gamma_1\|\Phi\|^2\left(\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma(N)\right)^2+\gamma_2\left(\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}e_k^\sigma(N)\right)^2\\
&\le\left(-2+\gamma_1\|\Phi\|^2\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}+\gamma_2\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}\right)\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}\left(e_k^\sigma(N)\right)^2\\
&=-\kappa_2\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}\left(e_k^\sigma(N)\right)^2.
\end{aligned}
$$

The sign of σ_k does not influence the validation of last inequality in the above derivation. If $\sigma_k>0$, $e_k^\sigma(N)+\left|e_k^\sigma(N)\right|\ge 0$. Thus it is easy to get

$$
\sigma_k\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}\left(e_k^\sigma(N)+\mathrm{sgn}(\sigma_k)\left|e_k^\sigma(N)\right|\right)>0. \tag{36}
$$

While if $\sigma_k<0$, $\left|e_k^\sigma(N)\right|-e_k^\sigma(N)\le 0$, thus Eq. (36) also hold. Define κ_2 as follows

$$
\kappa_2=2-\gamma_1\|\Phi\|^2\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}-\gamma_2\frac{\|G(\chi)\|^2}{\mu+\|\Phi\|^2\|G(\chi)\|^2}. \tag{37}
$$

By selecting suitable value for γ_1, γ_2, Φ, we can ensure that $\kappa_2>0$ is satisfied. This further leads to $\kappa_2\left[e_k^\sigma(N)\right]^2\le V_k-V_{k+1}$, and therefore $\forall n$,

$$
\sum_{k=0}^{n}\kappa_2\left[e_k^\sigma(N)\right]^2\le\sum_{k=0}^{n}V_k-V_{k+1}=V_0-V_{n+1}<V_0. \tag{38}
$$

Then it follows that $e_k^\sigma(N)\to 0$ as k goes to infinity. The boundedness of V_k at each iteration also ensures the boundedness of $\tilde{u}_k$, $\tilde{\sigma}_k$. The boundedness of $e_k(N)$ at each iteration could be concluded by (22) because σ_k is always bounded. Furthermore, it will satisfy $\lim_{k\to\infty}|e_k(N)|\le|\sigma_\infty|$. This completes the proof. □

References

[1] S. Arimoto, S. Kawamura, F. Miyazaki, Bettering operation of robots by learning, J. Robot. Syst. 1 (2) (1984) 123–140.

[2] H.-S. Ahn, Y. Chen, K.L. Moore, Iterative learning control: brief survey and categorization, IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 37 (6) (2007) 1099.
[3] Y. Wang, F. Gao, F.J. Doyle, Survey on iterative learning control, repetitive control, and run-to-run control, J. Process Control 19 (10) (2009) 1589–1600.
[4] Y. Chen, J.-X. Xu, C. Wen, A high-order terminal iterative learning control scheme, in: Proceedings of the Thirty-Sixth IEEE Conference on Decision and Control, 4, IEEE, 1997, pp. 3771–3772.
[5] Z. Hou, Y. Wang, C. Yin, T. Tang, Terminal iterative learning control based station stop control of a train, Int. J. Control 84 (7) (2011) 1263–1274.
[6] R. Chi, D. Wang, Z. Hou, S. Jin, Data-driven optimal terminal iterative learning control, J. Process Control 22 (10) (2012) 2026–2037.
[7] R. Chi, Z. Hou, S. Jin, D. Wang, Improved data-driven optimal TILC using time-varying input signals, J. Process Control 24 (12) (2014) 78–85.
[8] R. Chi, B. Huang, D. Wang, R. Zhang, Y. Feng, Data-driven optimal terminal iterative learning control with initial value dynamic compensation, IET Control Theory Appl. 10 (12) (2016) 1357–1364.
[9] S. Jin, Z. Hou, R. Chi, Optimal terminal iterative learning control for the automatic train stop system, Asian J. Control 17 (5) (2015) 1992–1999.
[10] R. Chi, N. Lin, R. Zhang, B. Huang, Y. Feng, Stochastic high-order internal model-based adaptive TILC with random uncertainties in initial states and desired reference points, Int. J. Adapt. Control Signal Process. (2017), doi:10.1002/acs.2707.
[11] L.P. Zhang, F.W. Yang, Study on the application of iterative learning control to terminal control of linear time-varying systems, Acta Autom. Sin. 31 (2) (2005) 309–313.
[12] S. Boudria, G. Gauthier, High order robust terminal iterative learning control design using genetic algorithm, in: Proceedings of the IECON Thirty-Eight Annual Conference on IEEE Industrial Electronics Society, IEEE, 2012, pp. 2313–2318.
[13] C.T. Freeman, Z. Cai, E. Rogers, P.L. Lewin, Iterative learning control for multiple point-to-point tracking application, IEEE Trans. Control Syst. Technol. 19 (3) (2011) 590–600.
[14] C.T. Freeman, Constrained point-to-point iterative learning control with experimental verification, Control Eng. Pract. 20 (5) (2012) 489–498.
[15] C.T. Freeman, Y. Tan, Iterative learning control with mixed constraints for point-to-point tracking, IEEE Trans. Control Syst. Technol. 21 (3) (2013) 604–616.
[16] B. Chu, C.T. Freeman, D.H. Owens, A novel design framework for point-to-point ILC using successive projection, IEEE Trans. Control Syst. Technol. 23 (3) (2015) 1156–1163.
[17] C.-J. Chien, L.-C. Fu, An iterative learning control of nonlinear systems using neural network design, Asian J. Control 4 (1) (2002) 21–29.
[18] Y. Liu, R. Chi, Z. Hou, Neural network state learning based adaptive terminal ILC for tracking iteration-varying target points, Int. J. Autom. Comput. 12 (3) (2015) 266–272.
[19] R. Chi, D. Wang, F.L. Lewis, Z. Hou, S. Jin, Adaptive terminal ILC for iteration-varying target points, Asian J. Control 17 (3) (2015) 952–962.
[20] Y.-C. Wang, C.-J. Chien, R. Chi, Z. Hou, A fuzzy-neural adaptive terminal iterative learning control for fed-batch fermentation processes, Int. J. Fuzzy Syst. 17 (3) (2015) 423–433.
[21] C.-J. Chien, Y.-C. Wang, R. Chi, D. Shen, An adaptive terminal iterative learning control for nonaffine nonlinear discrete-time systems, in: Proceedings of the twenty-Seventh Chinese Control and Decision Conference (CCDC), IEEE, 2015, pp. 1090–1094.
[22] J. Han, D. Shen, C.-J. Chien, Terminal iterative learning control for discrete-time nonlinear system based on neural networks, in: Proceedings of the Thirty-Fourth Chinese Control Conference, IEEE, 2015, pp. 3190–3195.
[23] R.D. Nussbaum, Some remarks on a conjecture in parameter adaptive control, Syst. Control Lett. 3 (5) (1983) 243–246.
[24] W.H. Ray, Advanced Process Control, McGraw-Hill Companies, 1981.
[25] J.S. Logsdon, L.T. Biegler, Accurate solution of differential-algebraic optimization problems, Ind. Eng. Chem. Res. 28 (11) (1989) 1628–1639.
[26] J.S. Logsdon, L.T. Biegler, Decomposition strategies for large-scale dynamic optimization problems, Chem. Eng. Sci. 47 (4) (1992) 851–864.