A SELF-TUNING ITERATIVE LEARNING CONTROLLER FOR TIME VARIANT SYSTEMS

Ali Madady

ABSTRACT

We consider the iterative learning control problem from an adaptive control viewpoint. The self-tuning iterative learning control systems (STILCS) problem is formulated in a general case, where the underlying linear system is time-variant and its parameters are all unknown and where its initial conditions are not constant and not determinable in various iterations. A procedure for solving this problem will be presented. The Lyapunov technique is employed to ensure the convergence of the presented STILCS. Computer simulation results are included to illustrate the effectiveness of the proposed STILCS.

Key Words: Iterative learning, repetitive systems, time variant systems, self tuning control.

I. INTRODUCTION

Iterative learning control (ILC) was originally proposed in the robotic communities as an intelligent learning mechanism for robot manipulators [1]. ILC is an effective method for controlling the systems, which perform a task (not necessarily a fixed task) repeatedly over a finite time-interval. The principle of ILC can be described as follows. At each execution of the control algorithm, some data are recorded as errors. This data is used by the learning algorithm in the next execution for modifying the control inputs and progressively reducing the output errors. After some iterations, the system should obtain an appropriate control input.

In the last two decades this field has attracted considerable research interest and has achieved significant progress in both theory and application. Interested readers can refer to [2], a comprehensive book on ILC survey.

In this paper, we address the problem of designing a controller for repetitive systems where the parameters of the systems are unknown. As in traditional (un-repetitive) cases, in which the adaptive techniques are used to design the controller for unknown systems, we can use adaptive control schemes to control repetitive unknown systems. For this purpose, some efforts are done to use adaptive control approaches in iterative learning. An adaptive robust ILC method based on a dead-zone scheme was presented in [3] for control of a class of continuous-time nonlinear uncertain systems. In [4] a non-linear iterative learning technique, based on an adaptive Lyapunov technique, was offered to control the discrete-time linear systems. A two-dimensional (2D) system approach based on nonlinear adaptive control techniques was offered in [5]. The linear matrix inequality (LMI) approach has been studied for analyzing and controller designing of discrete linear repetitive systems in [6]. The problem of adaptive ILC for robot manipulators was solved in [7] and its experimental results were reported in [8]. In [9], an output tracking error model based on the signals filtered from the plant input and output data was derived, then a new output-based adaptive iterative learning controller for repeatable SISO (single input-single output) linear time-invariant systems was presented. A model reference adaptive control strategy was used in [10] to design an iterative learning controller for a special class of repeatable nonlinear systems with uncertain parameters. In [11], the application of a model-based iterative learning control technique was presented to
position tracking of a piezoelectric system. The bounded input–bounded output (BIBO) stability of an adaptive time-frequency ILC was analyzed in [12]. In [13], a new adaptive switching learning control approach, which is a combination of the feedback proportional and derivative (PD) control law with a gain switching technique, was proposed for trajectory tracking of robot manipulators in an iterative operation mode. A new parameter-optimal high-order ILC algorithm was proposed based on the parameter optimization in [14]. A D-type iterative learning controller was developed for systems with unknown relative degree [15]. In [16] an adaptive model reference type iterative learning controller for unknown linear systems was presented in repetition domain. A new model reference adaptive control strategy was proposed in [17] for continuous-time SISO linear time-invariant systems with unknown parameters, which perform repetitive tasks. In [18] a phase-lead type learning algorithm was established for non-minimum phase plants to remove the instability and improve the convergence and final error. Application of iterative learning control as an intervention aid to stroke rehabilitation was studied experimentally in [19].

However, any of above papers, doesn’t address linear time-varying unknown systems with variable initial conditions. In this paper, we deal with the learning control problem for a discrete linear time-varying unknown system, where the system parameters are all unknown and its initial conditions are variable in various iterations. The aim of this paper is to extend the self-tuning approach to repetitive cases when the underlying unknown systems are time-variant.

The paper is organized as follows. Section II formulates the STILCS (self-tuning iterative learning control systems) problem. In Section III, we solve this problem. The convergence of the proposed self-tuning iterative learning procedure is analyzed in Section IV. In Section V, a numerical simulation example is given. Conclusions are presented in Section VI.

II. PROBLEM FORMULATION

Suppose the underlying discrete-time repetitive system described by:

\[
\begin{align*}
x(i + 1, j) &= A(i)x(i, j) + b(i)u(i, j), \quad i = 0, 1, \ldots, M \\
y(i, j) &= c^T(i - 1)x(i, j) \\
&= c^T(i - 1)x(i, j) \\
&= c^T(i - 1)x(i, j) \\
&= c^T(i - 1)x(i, j)
\end{align*}
\]

where \( x \in \mathbb{R}^n \), \( u \in \mathbb{R} \), and \( y \in \mathbb{R} \) denote the state, the input, and the output, respectively. Integer independent variables \( i \) and \( j \) denote the time variable and the operation or iteration number, respectively. Integer \( M \) is the time duration of the iterations. \( A(i) \), \( b(i) \), and \( c(i - 1) \) are real-valued coefficients with appropriate dimensions that are also time-variant, but assumed to be the same from one iteration to the next. \( T \) denotes the transpose.

We define the STILCS problem as follows:

Consider (1) and make the following reasonable assumptions:

A1. All the system parameters, namely the matrix \( A(i) \) and the vectors \( b(i) \) and \( c(i) \), are unknown.

A2. All components of \( A(i) \), \( b(i) \) and \( c(i) \) are bounded functions of \( i \) (for \( i = 0, 1, \ldots, M \)).

A3. The system initial conditions in each iteration \( x(0, j) \) are not determinable and may be any random value.

A4. A collection of bounded desired output trajectories \( y_d(i, j) \), which may be different for various iterations, is given. That is, the control task is not necessarily fixed.

A5. The state \( x(i, j) \) is accessible.

A6. The scalars \( c^T(i) b(i) \) are all nonzero (for \( i = 0, 1, \ldots, M \)).

Determine the control input sequence \( u(i, j) \), such that the following tracking can be established:

\[
\lim_{j \to \infty} (y(i, j) - y_d(i, j)) = 0
\]

for \( i = 1, 2, \ldots, M + 1 \) (2)

Remark 1. Assumption A6 is a standard assumption in ILC design which guarantees the existence of the learning gains. This is not really a restriction because it can be satisfied by choosing a proper sampling period in discretizing the continuous-time systems.

III. SOLUTION PROCEDURE OF STILCS PROBLEM

In the STILCS problem, we deal with controlling repetitive systems with unknown parameters. Both adaptive control approaches, MRAC (model reference adaptive control) and STR (self-tuning regulators), are reported to have nice features for dealing with uncertainty. However, they are not well-suited for handling repetitive trajectory tracking control tasks, since adaptive control does not utilize the knowledge that the operation is repetitive.
Here, we extend the self-tuning approach to repetitive trajectory tracking. For this purpose, the following closed-loop control law is proposed for determining \( u(i, j) \) in (1):

\[
\begin{align*}
  u(i, j) &= F(i, j)x(i, j) + G(i, j)y_d(i + 1, j) \\
  \text{for} \quad i &= 0, 1, \ldots, M, \quad j = 0, 1, \ldots \tag{3}
\end{align*}
\]

where \( F(i, j) \in \mathbb{R}^{1 \times n} \) and \( G(i, j) \in \mathbb{R} \) are adjustable gains, and we call them the learning gains. Substituting for \( u(i, j) \) from (3) into (1) yields:

\[
\begin{align*}
  x(i + 1, j) &= (A(i) + b(i)F(i, j))x(i, j) \\
  &\quad + b(i)G(i, j)y_d(i + 1, j) \\
  \text{for} \quad i &= 1, 2, \ldots, M + 1, \quad j = 0, 1, \ldots \tag{4}
\end{align*}
\]

Multiplying (4) from left side by \( c^T(i) \) implies:

\[
\begin{align*}
  y(i + 1, j) &= c^T(i)(A(i) + b(i)F(i, j))x(i, j) \\
  &\quad + c^T(i)b(i)G(i, j)y_d(i + 1, j) \\
  \text{for} \quad i &= 0, 1, \ldots, M \quad \text{and} \quad j = 0, 1, \ldots \tag{5}
\end{align*}
\]

If we choose \( F(i, j) \) and \( G(i, j) \) as follows:

\[
\begin{align*}
  F(i, j) &= -\frac{e^T(i)A(i)}{c^T(i)b(i)}, \quad G(i, j) = \frac{1}{c^T(i)b(i)} \tag{6}
\end{align*}
\]

Then we will have:

\[
\begin{align*}
  y(i, j) &= y_d(i, j) \\
  \text{for} \quad i &= 1, 2, \ldots, M + 1, \quad j = 0, 1, \ldots \tag{7}
\end{align*}
\]

However, \( A(i), b(i), \) and \( c(i) \) are unknown. Thus, these should first be estimated then \( F(i, j) \) and \( G(i, j) \) can be determined according to the following equations:

\[
\begin{align*}
  F(i, j) &= \frac{-\hat{e}^T(i, j)\hat{A}(i, j)}{\hat{e}^T(i, j)\hat{b}(i, j)}, \\
  G(i, j) &= \frac{1}{\hat{e}^T(i, j)\hat{b}(i, j)} \tag{8}
\end{align*}
\]

where \( \hat{A}(i, j), \hat{b}(i, j), \) and \( \hat{c}(i, j) \) are, respectively, the estimations of \( A(i), b(i), \) and \( c(i) \) in the \( j \)th iteration, and they are determined by a suitable method so that the following condition holds for all \( i \in \{0, 1, \ldots, M\} \) and \( j \in \{0, 1, \ldots\} \):

\[
\hat{e}^T(i, j)\hat{b}(i, j) \neq 0 \tag{9}
\]

until the learning gains \( F(i, j) \) and \( G(i, j) \) always exist and are bounded.

The next step is to establish an online adaptive algorithm for estimating \( A(i), b(i), \) and \( c(i) \) so that (9) and (2) hold. For this purpose (1) is written as following compact form:

\[
\begin{align*}
  Y(i, j) &= \theta(i)z(i, j) \\
  \text{for} \quad i &= 0, 1, \ldots, M, \quad j = 0, 1, \ldots \tag{10}
\end{align*}
\]

where:

\[
\begin{align*}
  Y(i, j) &= \begin{bmatrix} x(i + 1, j) \\ y(i + 1, j) \end{bmatrix}, \\
  z(i, j) &= \begin{bmatrix} x(i, j) \\ u(i, j) \end{bmatrix}, \\
  \theta(i) &= \begin{bmatrix} A(i) & b(i) \end{bmatrix} [0 \ 0 \ c^T(i)]
\end{align*}
\]

The relation (10) has two-dimensional regressor form, which is introduced in [20]. According to the procedure of adaptive parameter estimation for two-dimensional systems, which is discussed in [20], a suitable algorithm is proposed as follows, for estimating the components of \( \theta(i) \) in (10):

\[
\begin{align*}
  \hat{A}(i, j + 1) &= \hat{A}(i, j) \\
  -\mu(i, j)P_1 \begin{bmatrix} e_x(i, j) \\ e_y(i, j) \end{bmatrix} x^T(i, j) \\
  \hat{b}(i, j + 1) &= \hat{b}(i, j) \\
  -\mu(i, j)P_1 \begin{bmatrix} e_x(i, j) \\ e_y(i, j) \end{bmatrix} u(i, j) \tag{11a}
\end{align*}
\]

\[
\begin{align*}
  \hat{c}(i, j + 1) &= \hat{c}(i, j) - \mu(i, j)P_2 \begin{bmatrix} e_x(i, j) \\ e_y(i, j) \end{bmatrix} x^T(i + 1, j) \\
  \text{for} \quad i &= 0, 1, \ldots, M, \quad j = 0, 1, \ldots \tag{11b}
\end{align*}
\]

where:

\[
\begin{align*}
  e_x(i, j) &= \hat{A}(i, j)x(i, j) + \hat{b}(i, j)u(i, j) - x(i + 1, j) \\
  e_y(i, j) &= \hat{c}(i, j)x(i + 1, j) - y(i + 1, j) \\
  \text{for} \quad i &= 0, 1, \ldots, M, \quad j = 0, 1, \ldots \tag{12}
\end{align*}
\]

Also, \( \mu(i, j) \) is a positive scalar called algorithm step size, \( P_1 \) and \( P_2 \) are respectively the \( n \) first rows, and
the last row of an arbitrary symmetric positive definite matrix \( P \), that is:

\[
P = \begin{bmatrix}
P_1 \\
P_2
\end{bmatrix}, \quad P_1 \in \mathbb{R}^{n \times (n+1)},
\]

\[
P_2 \in \mathbb{R}^{1 \times (n+1)}, \quad P = P^T > 0
\]  

(13)

For establishing the important condition (9), which guarantees the existence and boundedness of the learning gains \( F(i, j) \) and \( G(i, j) \), we do the following steps:

S1. In the choosing of the initial conditions for algorithm (11), we select \( \hat{b}(i, 0) \) and \( \hat{c}^T(i, 0) \) so that their multiplication is nonzero, that is:

\[
\hat{c}^T(i, 0) \hat{b}(i, 0) \neq 0 \quad \text{for } i = 0, 1, \ldots, M
\]  

(14)

S2. We provide some conditions so that from the following assumption

\[
\hat{c}^T(i, j) \hat{b}(i, j) \neq 0 \quad \text{for } i = 0, 1, \ldots, M
\]  

(15)

the following result could be obtained:

\[
\hat{c}^T(i, j + 1) \hat{b}(i, j + 1) \neq 0
\]

for \( i = 0, 1, \ldots, M \)

(16)

Therefore, using the above two steps and mathematical induction, condition (9) will be guaranteed for all \( i \in \{0, 1, \ldots, M\} \) and \( j \in \{0, 1, \ldots, M\} \).

In order to provide the necessary conditions for step S2, (11b) is multiplied by (11c) from the left side. Therefore, we achieve the following relation:

\[
\hat{c}^T(i, j + 1) \hat{b}(i, j + 1) = \hat{c}^T(i, j) \hat{b}(i, j) + \alpha(i, j) u^2(i, j) + \beta(i, j) \mu(i, j)
\]  

(17)

where scalars \( \alpha(i, j) \) and \( \beta(i, j) \) are

\[
\alpha(i, j) = P_2 \epsilon(i, j) x^T(i + 1, j) P_1 e(i, j) u(i, j)
\]

\[
\beta(i, j) = -\hat{c}^T(i, j) P_1 e(i, j) u(i, j) + P_2 \epsilon(i, j) x^T(i + 1, j) \hat{b}(i, j)
\]

For any given \( \hat{c}^T(i, j) \hat{b}(i, j) \neq 0 \), the following equation has a maximum of two real roots for \( \mu(i, j) \):

\[
\alpha(i, j) u^2(i, j) + \beta(i, j) \mu(i, j)
\]

\[
+ \hat{c}^T(i, j) \hat{b}(i, j) = 0
\]  

(18)

Choosing \( \mu(i, j) \), we select it so that it is not equal to the roots of (18). Therefore, (16) will hold.

One can choose \( \mu(i, j) \) so that for some \( \epsilon > 0 \) the following assumption

\[
|\hat{c}^T(i, j) \hat{b}(i, j)| \geq \epsilon \quad \text{for } i = 0, 1, \ldots, M
\]

which implies:

\[
|\hat{c}^T(i, j + 1) \hat{b}(i, j + 1)| \geq \epsilon \quad \text{for } i = 0, 1, \ldots, M
\]

For this purpose it is sufficient that \( \mu(i, j) \) is chosen sufficiently away from the roots of (18).

The control law (3), algebraic equations (8), and the adjusting algorithm (11) are the main parts of the presented STILCS. Thus, STILCS can be represented as Fig. 1.

**IV. CONVERGENCE ANALYSIS**

The following definition is presented:

**Definition.** The proposed STILCS is said to be convergent, if, for any initial conditions \( x(0, j) \), it generates an input sequence \( u(i, j) \) for system (1), so that (2) holds.

**Theorem.** The presented STILCS is convergent if the step size \( \mu(i, j) \) in the adjusting algorithm (11) is chosen in the following interval:

\[
0 < \mu(i, j) < \frac{2}{\max(r_1(i, j), r_2(i, j)) \lambda_{\text{max}}(P)}
\]  

(19)

where:

\[
r_1(i, j) = x^T(i, j) x(i, j) + u^2(i, j),
\]

\[
r_2(i, j) = x^T(i + 1, j) x(i + 1, j)
\]  

(20)

and \( \lambda_{\text{max}}(P) \) denotes the largest eigenvalue of \( P \).

**Remark 2.** According to assumption A2, all system parameters (components of \( A(i) \), \( b(i) \), and \( c(i) \)) are bounded functions of \( i \) (for \( i = 0, 1, \ldots, M \) and, as per A4, the desired output trajectory \( y_d(i, j) \) is a bounded function of \( i \) and \( j \). Also, according to (9), the controller parameters \( F(i, j) \) and \( G(i, j) \) are bounded. Thus, by attention to this fact that our system is linear and operates over a finite-time interval \( 0 \leq i \leq M \) (for example \( 0 \leq i \leq 10 \)) in all iterations, the boundedness of system parameters, desired output trajectory and controller parameters guarantee the boundedness of the state \( x(i, j) \) and input \( u(i, j) \) (for a similar case see [17]). Thus, \( r_1(i, j) \) and \( r_2(i, j) \) are bounded in all iterations for finite time interval \( 0 \leq i \leq M \). Hence, according to (19)
Proof of Theorem. We consider $V(i, j)$ as follows for candidate of Lyapunov function:

$$V(i, j) = \text{Trace} [\tilde{A}^T(i, j)\tilde{A}(i, j)] + \tilde{b}^T(i, j)\tilde{b}(i, j) + \tilde{c}^T(i, j)\tilde{c}(i, j)$$

$$i = 0, 1, \ldots, M, \ j = 0, 1, \ldots$$  \hspace{1cm} (21)

where \text{Trace} is the sum of the diagonal elements of the matrix, and:

$$\tilde{A}(i, j) = A(i, j) - A(i)$$
$$\tilde{b}(i, j) = b(i, j) - b(i)$$
$$\tilde{c}(i, j) = c(i, j) - c(i)$$

Variation of $V(i, j)$ is defined as follows:

$$\Delta V(i, j) = V(i, j + 1) - V(i, j)$$
$$i = 0, 1, \ldots, M, \ j = 0, 1, \ldots$$  \hspace{1cm} (22)

We can write:

$$\Delta V(i, j) = \Delta V_A(i, j) + \Delta V_b(i, j) + \Delta V_c(i, j)$$  \hspace{1cm} (23)

where:

$$\Delta V_A(i, j) = \text{Trace} [\tilde{A}^T(i, j + 1)\tilde{A}(i, j + 1) - \tilde{A}^T(i, j)\tilde{A}(i, j)]$$

$$\Delta V_b(i, j) = \tilde{b}^T(i, j + 1)\tilde{b}(i, j + 1) - \tilde{b}^T(i, j)\tilde{b}(i, j)$$

$$\Delta V_c(i, j) = \tilde{c}^T(i, j + 1)\tilde{c}(i, j + 1) - \tilde{c}^T(i, j)\tilde{c}(i, j)$$

From (11a) we have:

$$\begin{cases}
\tilde{A}(i, j + 1) = \tilde{A}(i, j) - \mu(i, j)P_1e(i, j)x^T(i, j) \\
\tilde{A}^T(i, j + 1) = \tilde{A}^T(i, j) - \mu(i, j)x(i, j)e^T(i, j)P_1^T
\end{cases}$$  \hspace{1cm} (24)

where:

$$e(i, j) = \begin{bmatrix} e_x(i, j) \\ e_y(i, j) \end{bmatrix}$$  \hspace{1cm} (25)

From (24) we get:

$$\tilde{A}^T(i, j + 1)\tilde{A}(i, j + 1) - \tilde{A}^T(i, j)\tilde{A}(i, j)$$

$$= \mu^2(i, j)x(i, j)e^T(i, j)P_1^TP_1e(i, j)x^T(i, j)$$

$$- \mu(i, j)\tilde{A}^T(i, j)\tilde{A}(i, j) - \mu(i, j)x(i, j)e^T(i, j)P_1^T\tilde{A}(i, j)$$

and:

$$\Delta V_A(i, j) = \text{Trace} [\tilde{A}^T(i, j + 1)\tilde{A}(i, j + 1) - \tilde{A}^T(i, j)\tilde{A}(i, j)]$$

$$\Delta V_b(i, j) = \tilde{b}^T(i, j + 1)\tilde{b}(i, j + 1) - \tilde{b}^T(i, j)\tilde{b}(i, j)$$

$$\Delta V_c(i, j) = \tilde{c}^T(i, j + 1)\tilde{c}(i, j + 1) - \tilde{c}^T(i, j)\tilde{c}(i, j)$$
Hence:
\[ \Delta V_A(i, j) \]
\[ = -\mu(i, j)\text{Trace}[\hat{A}^T(i, j)P_1e(i, j)x^T(i, j)] \]
\[ -\mu(i, j)\text{Trace}[x(i, j)e^T(i, j)P_1^T\hat{A}(i, j)] \]
\[ +\mu^2(i, j)\text{Trace}[x(i, j)e^T(i, j)P_1^TP_1e(i, j)] \]
\[ \times e(i, j)x^T(i, j) \]
\[ (26) \]

We know that for any two vectors \( \Theta, \Phi \in \mathbb{R}^{n \times 1} \) the following relation holds:
\[ \text{Trace}[\Theta\Phi^T] = \text{Trace}[\Phi^T\Theta] = \Phi^T\Theta \]

Thus (26) can be written as follows:
\[ \Delta V_A(i, j) = -2\mu(i, j)x^T(i, j)\hat{A}^T(i, j)P_1e(i, j) \]
\[ +\mu^2(i, j)x^T(i, j)x(i, j) \]
\[ \times e^T(i, j)P_1^TP_1e(i, j) \]
\[ (27) \]

From (11b) we have:
\[ \begin{align*}
\hat{b}(i, j + 1) &= \hat{b}(i, j) - \mu(i, j)P_1e(i, j)u(i, j) \\
\hat{b}^T(i, j + 1) &= \hat{b}^T(i, j) \]
\[ -\mu(i, j)u(i, j)e^T(i, j)P_1^TP_1e(i, j) \]
\[ (28) \]

From the above relations we get:
\[ \Delta V_b(i, j) = -2\mu(i, j)u(i, j)\hat{b}^T(i, j)P_1e(i, j) \]
\[ +\mu^2(i, j)u^2(i, j)e^T(i, j)P_1^TP_1e(i, j) \]
\[ (29) \]

In a similar manner, from (11c) one gets:
\[ \Delta V_c(i, j) = -2\mu(i, j)x^T(i, j + 1)\hat{c}(i, j)P_2e(i, j) \]
\[ +\mu^2(i, j)x^T(i, j + 1)x(i, j + 1) \]
\[ \times e^T(i, j)P_2^TP_2e(i, j) \]
\[ (30) \]

Substituting \( \Delta V_A(i, j), \Delta V_b(i, j) \) and \( \Delta V_c(i, j) \) respectively from (27)–(29) into (23) yields:
\[ \Delta V(i, j) = \Delta V_1(i, j) + \Delta V_2(i, j) \]
\[ (31) \]

where:
\[ \Delta V_1(i, j) = -2\mu(i, j)x^T(i, j)\hat{A}^T(i, j) \]
\[ +u(i, j)\hat{b}^T(i, j); \]
\[ x^T(i, j + 1)\hat{c}(i, j)P_2e(i, j) \]
\[ (32) \]

and \( r_1(i, j), r_2(i, j) \) are given by (20).

From (12) and (1) we have:
\[ e_x(i, j) = \hat{A}(i, j)x(i, j) + \hat{b}(i, j)u(i, j) \]
\[ (33) \]
\[ e_y(i, j) = e^T(i, j)x(i + 1, j) \]
\[ (34) \]

From (25), (31), (33) and (34) the following result is obtained:
\[ \Delta V_1(i, j) = -2\mu(i, j)e^T(i, j)P_1e(i, j) \]
\[ (35) \]

Also, we can write (32) as follows:
\[ \Delta V_2(i, j) = \mu^2(i, j)e^T(i, j)[P_1^TP_1^T] \]
\[ \times \begin{bmatrix}
    r_1(i, j)I_n & 0 \\
    0 & r_2(i, j)
\end{bmatrix}
\begin{bmatrix}
P_1 \\
P_2
\end{bmatrix}
\begin{bmatrix}
e(i, j)
\end{bmatrix} \]
\[ (36) \]

where \( I_n \in \mathbb{R}^{n \times n} \) is the identity matrix.

or:
\[ \Delta V_2(i, j) = \mu^2(i, j)e^T(i, j)PQ(i, j)Pe(i, j) \]
\[ (37) \]

where matrix \( Q(i, j) \in \mathbb{R}^{(n+1) \times (n+1)} \) is as follows:
\[ Q(i, j) = \begin{bmatrix}
r_1(i, j)I_n & 0 \\
0 & r_2(i, j)
\end{bmatrix} \]
\[ (38) \]

Substituting \( \Delta V_1(i, j) \) and \( \Delta V_2(i, j) \), respectively, from (35) and (36) into (30) yields:
\[ \Delta V(i, j) = -e^T(i, j)H(i, j)e(i, j) \]
\[ i = 0, 1, \ldots, M, \quad j = 0, 1, \ldots \]
\[ (39) \]

where \( H(i, j) \) is the following symmetric matrix:
\[ H(i, j) = 2\mu(i, j)P - \mu^2(i, j)PQ(i, j)P \]
\[ (40) \]

It is easy to show that if \( \mu(i, j) \) is in the interval (19), then the symmetric matrix \( H(i, j) \) will be positive definite and we will have:
\[ \Delta V(i, j) \leq 0, \quad i = 0, 1, \ldots, M, \quad j = 0, 1, \ldots \]

That is, \( V(i, j) \) is a non-increasing function along \( j \) direction and hence \( \hat{A}(i, j), \hat{b}(i, j) \) and \( \hat{c}(i, j) \) will be bounded. Also since \( V(i, j) \) is a nonnegative sequence,
we conclude from (40):

\[
\lim_{j \to \infty} \Delta V(i, j) = 0 \quad \text{for } i = 0, 1, \ldots, M \quad (41)
\]

Since \( \mathbf{H}(i, j) \) is a symmetric and positive definite matrix, equation \( \Delta V(i, j) = 0 \) implies \( e(i, j) = 0 \), i.e. from (41) we get:

\[
\lim_{j \to \infty} e(i, j) = 0 \quad \text{for } i = 0, 1, \ldots, M \quad (42)
\]

From (25) and (42) the following results are obtained:

\[
\begin{align*}
\lim_{j \to \infty} e_x(i, j) &= 0 \quad \text{and} \\
\lim_{j \to \infty} e_y(i, j) &= 0, \quad i = 0, 1, \ldots, M
\end{align*} \quad (43)
\]

From (43) one can conclude that, for sufficiently large \( j \), we have:

\[
\begin{align*}
e_x(i, j) &= 0, \quad e_y(i, j) = 0 \quad \text{for } i = 0, 1, \ldots, M \quad (44)
\end{align*}
\]

Therefore, from (11), the constant values relative to \( j \) are obtained for \( \mathbf{A}(i, j) \), \( \mathbf{b}(i, j) \), and \( \mathbf{c}(i, j) \), like \( \hat{\mathbf{A}}_i \), \( \hat{\mathbf{b}}_i \), and \( \hat{\mathbf{c}}_i \), respectively, that is:

\[
\hat{\mathbf{A}}(i, j) = \hat{\mathbf{A}}_i, \quad \hat{\mathbf{b}}(i, j) = \hat{\mathbf{b}}_i, \quad \hat{\mathbf{c}}(i, j) = \hat{\mathbf{c}}_i
\]

for \( i = 0, 1, \ldots, M \) and sufficiently large \( j \) \quad (45)

Since for \( i \in \{0, 1, \ldots, M\} \), \( \hat{\mathbf{c}}_i^T \hat{\mathbf{b}}_i \) are the final values of \( \hat{\mathbf{c}}_i^T(i, j) \hat{\mathbf{b}}(i, j) \) and according to (9) the amounts of \( \hat{\mathbf{c}}_i^T(i, j) \hat{\mathbf{b}}(i, j) \) are nonzero for all \( i \in \{0, 1, \ldots, M\} \) and \( j \in \{0, 1, \ldots\} \), we conclude:

\[
\hat{\mathbf{c}}_i^T \hat{\mathbf{b}}_i \neq 0 \quad \text{for } i = 0, 1, \ldots, M
\]

From (12), (44) and (45), the following relations can be concluded:

\[
\begin{align*}
x(i + 1, j) &= \hat{\mathbf{A}}_i x(i, j) + \hat{\mathbf{b}}_i u(i, j) \\
&\quad \text{for } i = 0, 1, \ldots, M \text{ and sufficiently large } j
\end{align*} \quad (46)
\]

\[
y(i, j) = \hat{\mathbf{c}}_{i-1}^T x(i, j)
\]

for \( i = 1, 2, \ldots, M + 1 \) and sufficiently large \( j \) \quad (47)

Although the above relations are similar to (1), it will be illustrated by simulation example in Section 5 that their coefficients are not necessarily equal to the coefficients of (1). That is we may have:

\[
\hat{\mathbf{A}}_i \neq \mathbf{A}(i), \quad \hat{\mathbf{b}}_i \neq \mathbf{b}(i), \quad \hat{\mathbf{c}}_i \neq \mathbf{c}(i)
\]

for some \( i \in \{0, 1, \ldots, M\} \)

However (3), (8) and (45) imply:

\[
u(i, j) = -\frac{\hat{\mathbf{c}}_i^T \hat{\mathbf{A}}_i}{\hat{\mathbf{c}}_i^T \hat{\mathbf{b}}_i} x(i, j) + \frac{1}{\hat{\mathbf{c}}_i^T \hat{\mathbf{b}}_i} y_d(i + 1, j)
\]

for \( i = 0, 1, \ldots, M \) and sufficiently large \( j \) \quad (48)

Substituting \( u(i, j) \) from (48) into (46) yields:

\[
x(i, j) = \left( \hat{\mathbf{A}}_{i-1} - \frac{\hat{\mathbf{b}}_{i-1} \hat{\mathbf{c}}_{i-1}^T \hat{\mathbf{A}}_{i-1}}{\hat{\mathbf{c}}_{i-1}^T \hat{\mathbf{b}}_{i-1}} \right) x(i - 1, j) \\
&\quad + \frac{\hat{\mathbf{b}}_{i-1}}{\hat{\mathbf{c}}_{i-1}^T \hat{\mathbf{b}}_{i-1}} y_d(i, j)
\]

for \( i = 1, 2, \ldots, M + 1 \)

and sufficiently large \( j \)

Multiplying above relation from left side by \( \hat{\mathbf{c}}_{i-1}^T \) implies:

\[
\hat{\mathbf{c}}_{i-1}^T x(i, j) = y_d(i, j)
\]

for \( i = 1, 2, \ldots, M + 1 \) and sufficiently large \( j \) \quad (49)

From (47) and (49) we have:

\[
y(i, j) = y_d(i, j)
\]

for \( i = 1, 2, \ldots, M + 1 \) and sufficiently large \( j \) \quad (50)

That means:

\[
\lim_{j \to \infty} y(i, j) = y_d(i, j), \quad i = 1, 2, \ldots, M + 1
\]

Here the proof of the theorem is completed.

**Remark 3.** For choosing \( \mu(i, j) \), as well as condition (19), we must consider equation (18). That is, if any roots of (18) lie in interval (19) we must select \( \mu(i, j) \) to be not equal to that root.
V. SIMULATION RESULTS

In order to illustrate the performance of the proposed STILCS procedure, a numerical example is presented. Consider the following time-variant second order repetitive system:

\[
\begin{bmatrix}
x_1(i + 1, j) \\
x_2(i + 1, j)
\end{bmatrix} = \mathbf{A}(i) \begin{bmatrix}
x_1(i, j) \\
x_2(i, j)
\end{bmatrix} + \mathbf{b}(i)u(i, j)
\]

\[
y(i, j) = \mathbf{c}^T(i - 1) \begin{bmatrix}
x_1(i, j) \\
x_2(i, j)
\end{bmatrix}
\]

\(i = 0, 1, \ldots, 20, \quad j = 0, 1, \ldots\)

where:

\[
\mathbf{A}(i) = \begin{bmatrix}
a_{11}(i) & a_{12}(i) \\
a_{21}(i) & a_{22}(i)
\end{bmatrix} = \begin{bmatrix}
1 + \frac{i}{19} & 2 \times (-1)^i \\
2^{-i} & 1.2 \sin \left( \frac{\pi i}{20} \right)
\end{bmatrix},
\]

\[
\mathbf{b}(i) = \begin{bmatrix}
b_1(i) \\
b_2(i)
\end{bmatrix} = \begin{bmatrix}
2 + \cos \left( \frac{\pi i}{10} \right) \\
0
\end{bmatrix}
\]

\[
\mathbf{c}^T(i) = [c_1(i) \quad c_2(i)] = \begin{bmatrix}
i + 1 & \frac{i}{i + 10}
\end{bmatrix}
\]

Matrix \(\mathbf{A}(i)\) and vectors \(\mathbf{b}(i), \mathbf{c}(i)\) are assumed to be unknown, and also the system initial conditions are supposed to be varied during the various iterations as follows:

\[
x_1(0, j) = 1 + \sin \left( \frac{\pi j}{50} \right), \quad x_2(0, j) = (-0.5)^j
\]

Desired output trajectory, which is shown in Fig. 2, is as follows for all iterations:

\[
y_d(i, j) = (i - 10.5)^2, \quad 1 \leq i \leq 20
\]

Matrix \(\mathbf{P}\) is chosen as identity matrix in the adjusting algorithm (11), and considering (19), \(\mu(i, j)\) is selected as follows:

\[
\mu(i, j) = \frac{1}{\max(r_1(i, j), r_2(i, j))\lambda_{\text{max}}(\mathbf{P})}
\]

Initial conditions of algorithm (11), which is the prior estimations of the system parameters, are taken as follows:

\[
\hat{\mathbf{A}}(i, 0) = \begin{bmatrix}
1.5 & 0 \\
0.5 & 1
\end{bmatrix}, \quad \hat{\mathbf{b}}(i, 0) = \begin{bmatrix}
2 \\
1
\end{bmatrix}, \quad \hat{\mathbf{c}}^T(i, 0) = [5 \quad 1]
\]

The obtained trajectories for system output are shown in Figs. 3–5 in some iterations. These results demonstrate that with increasing the number of iterations, the system output progressively became close to the desired output trajectory.

In order to have a careful evaluation of the simulation result, we define the total learning error in
iteration \( j \) as follows:

\[
E(j) = \sqrt{\sum_{i=1}^{M+1} (y(i, j) - y_d(i, j))^2}, \quad j = 0, 1, \ldots
\]

The result for \( E(j) \) from simulation is shown in Fig. 6 versus iteration number \( j \). It is seen that with increasing the iteration number, \( E(j) \) monotonically vanishes. This fact demonstrates the very good performance of the proposed STILCS procedure.

The obtained results for some components of \( \hat{A}(i, j), \hat{b}(i, j) \), and \( \hat{c}^T(i, j) \) are shown in Figs. 7–10 versus iteration number \( j \), where:

\[
\hat{A}(i, j) = \begin{bmatrix} \hat{a}_{11}(i, j) & \hat{a}_{12}(i, j) \\ \hat{a}_{21}(i, j) & \hat{a}_{22}(i, j) \end{bmatrix},
\]

\[
\hat{b}(i, j) = \begin{bmatrix} \hat{b}_1(i, j) \\ \hat{b}_2(i, j) \end{bmatrix},
\]

\[
\hat{c}^T(i, j) = [\hat{c}_1(i, j) \hat{c}_2(i, j)].
\]

The above figure demonstrates that the final value of \( \hat{a}_{11}(0, j) \) is approximately 1.555, while the value of
Fig. 8. The result for $\hat{a}_{12}(19, j)$.

Fig. 9. The results for $\hat{a}_{12}(19, j)$, $\hat{b}_{2}(10, j)$ and $\hat{b}_{2}(20, j)$.

Fig. 10. The result for $\hat{c}_{2}(10, j)$.

$\hat{a}_{11}(0)$ is 1. That is:

$$\lim_{j \to \infty} \hat{a}_{11}(0, j) = a_{11}(0)$$

Fig. 8 shows $\hat{a}_{12}(19, j)$ converges to −0.3, while we have $a_{12}(19) = -2$. Thus:

$$\lim_{j \to \infty} \hat{a}_{12}(19, j) \neq a_{12}(19)$$

Fig. 9 demonstrates final values of $\hat{b}_{2}(0, j)$, $\hat{b}_{2}(10, j)$ and $\hat{c}_{2}(20, j)$ are approximately 0, 0.4 and 1.07 respectively, while we have $b_{2}(i) = 0$ (for $i = 0, 1, \ldots, 20$). Hence:

$$\lim_{j \to \infty} \hat{b}_{2}(0, j) = b_{2}(0),$$

$$\lim_{j \to \infty} \hat{b}_{2}(10, j) \neq b_{2}(10),$$

$$\lim_{j \to \infty} \hat{b}_{2}(20, j) \neq b_{2}(20)$$

Fig. 10 shows that the $\hat{c}_{2}(10, j)$ is convergent to 0.5, also $c_{2}(10) = 0.5$. Thus, we have:

$$\lim_{j \to \infty} \hat{c}_{2}(10, j) = c_{2}(10)$$

In summary, Figs. 7–10 demonstrate that the final values of $\hat{A}(i, j)$, $\hat{b}(i, j)$, and $\hat{c}(i, j)$ may be different from $A(i)$, $b(i)$, and $c(i)$, respectively.

VI. CONCLUSION

This paper has extended the self-tuning control approach to repetitive systems. The STILCS problem is formulated in a general case, when the underlying repetitive linear system is possibly time-variant and its parameters are all unknown and its initial conditions is randomly. A solution procedure is presented for this problem. In this solution procedure, the system input is assigned according to a closed-loop control law...
incorporating the system state and the desired output trajectory. There are some feedback and feed-forward adjustable gains in this control law, which are called the learning gains. An algorithm is proposed for adjusting the learning gains, so that the learning gains are modified in each repetition, using the input-output data of the controlled system in previous operation. There is an easily selectable factor in this adjustment algorithm, named algorithm step size, where proper choice of it guarantees the boundedness of learning gains. The convergence of the presented STILCS procedure was analyzed via the Lyapunov approach, and convergence condition is obtained in terms of algorithm step size range.

Although the single-input and single-output case was studied, the obtained results can easily be extended to the multi-input and multi-output case. However, there are a number of problems which merit further research, and these are subjects of our future work. One such research problem is as follows:

Solving the STILCS problem when the state in the underlying system is not accessible but only the output of the system is measurable.

ACKNOWLEDGEMENTS

The useful comments from the Reviewers are gratefully acknowledged.

REFERENCES

18. Freeman, C. T., P. L. Lewin, and E. Rogers, “Further results on the experimental evaluation...


**Ali Madady** was born in Aqdash, Saveh, Iran in 1966. He received his B.S., M.S. and Ph.D. degrees in electrical and electronic engineering from Amir-Kabir University of Technology (Tehran Polytechnic), Iran, in 1989, 1993 and 2001, respectively. He is now an Assistant Professor at electrical engineering department of Tafresh University, Iran. His research interests include learning control, adaptive control, robust control and digital signal processing.