Starting Point Independent Iterative Learning Algorithm on LTI System with Non-zero Initial Errors

Byungyong You¹ and Jinwook Kim²

¹School of Mechanical and Automotive Engineering, Kyungil University
50 Gamasil-gil, Hayang-eup, Gyeonsansi, Gyeongbuk, 38428, Republic of Korea
²Korea Electrotechnology Research Institute12, Bulmosan-ro 10 beon-gil, Seongsan-gu, Changwon-si, Geongsangnam-do, 51543, Republic of Korea

E-mail: zealot@kiu.kr

Abstract. In this paper, we propose a starting point independent iterative learning algorithm. This algorithm does not need to train iterative learning control (ILC) algorithm each time when initial condition changes. If the plant is sufficiently stable, the output trajectory converges to the desired trajectory exponentially fast. Regardless of learning rule type, this algorithm can be used and has a same convergence rate. So in case of the multiple starting and goal points application problems, it can be used with significant reduced computational cost. In fact, if there are M starting points and N goal points, the conventional ILC algorithm needs MN training for generating whole trajectories, however proposed algorithm needs only N training. In order to expand our algorithm to the multi-dimensional point-to-point (PTP) problem, we propose a learning rule for PTP problem, then shows its convergence.

1. Introduction
The ILC is used widely in real world applications that perform the same tasks repetitively. The basic idea of ILC is that the information obtained from previous trial is utilized to improve the control input for next trial. ILC has many advantages because it requires no or less plant information and it is robust for repetitive disturbances and uncertainties in a system. However, ILC generally requires the initial condition at each iteration and it should be the same and equal to that of the desired trajectory[1]-[6]. This requirement significantly limits ILC applications in the real field. Therefore there have been substantial research efforts in ILC with non-zero initial errors.

Heinzinger et al. [7] has shown that the output errors are bounded in the case of bounded initial errors. But there is no information on parameters on how to adjust the size of the bounds. Lee et al. [8] proposed a PD-type ILC algorithm on LTI system with parameter which reduces the non-zero initial errors and exponentially converges to the desired trajectory. However, it needs to train ILC every time whenever the initial condition changes. Therefore, in this paper, we propose a starting point independent iterative learning algorithm (SPIILA) on LTI system with non-zero initial errors. This algorithm does not need to train ILC algorithm each time when initial condition changes. If the plant is sufficiently stable, the output trajectory converges to the desired trajectory exponentially fast. So in the case of the multiple starting and goal points application problems, it can be used with significantly reduced computational cost. In fact, if there are M starting points and N goal points, the conventional ILC algorithm needs MN training for generating whole desired input trajectories but proposed
algorithm needs only \( N \) training. In order to expand our algorithm to the multi-dimensional Point-to-Point (PTP) problem, we propose a learning rule for PTP, then shows its convergence.

This paper is organized as follows. Section 2 describes the ILC with non-zero initial errors. Section 3 details the usage and properties of the main theorem and propose the learning rule for PTP control problem and show the proof of its convergence. Section 4 shows computer simulation results. Finally, the conclusion and future works are given in Section 5.

In the following, ||\( x \)|| denotes the Euclidean norm of a vector \( x \). For a matrix \( A, A^T \) denotes its transpose and ||\( A \)|| denotes its induced matrix norm. When \( A \) is a square matrix, \( \lambda(A) \) is an eigenvalue of \( A \). \( I_n \) denotes the \( n \times n \) identity matrix. As in the notation \( u_k(t) \), the suffix \( k \) is employed to denote the iteration number.

2. ILC with non-zero initial errors

Consider the LTI system described by

\[
\begin{align*}
\dot{x}(t) &= A x(t) + B u(t) \\
y(t) &= C x(t)
\end{align*}
\]

where \( x \in \mathbb{R}^n, u \in \mathbb{R}^r \) and \( y \in \mathbb{R}^p \) denote the state, input and output respectively. \( A, B \) and \( C \) are matrices with appropriate dimensions and it is assumed that \( CB \) is non-singular. Then ILC algorithm for the system (1) can be described as follows [9]

\[
u_{k+1}(t) = u_k(t) + \gamma \delta y_k(t)
\]

where

\[
\begin{align*}
\dot{x}_k(t) &= A x_k(t) + B u_k(t) \\
y_k(t) &= C x_k(t) \\
y_d(t) &= C x_d(t)
\end{align*}
\]

and

\[
\delta y_k(t) = y_d(t) - y_k(t).
\]

If \( ||I_r - \gamma CB|| \leq \rho < 1 \)

and

\[
y_k(0) = y_d(0) \equiv 0, k = 0,1,2,\ldots.
\]

It is already known that by using the learning rule (2), the error between \( y_k(t) \) and \( y_d(t) \) approaches zero as \( k \to \infty \). In the following, we want to discuss the case when (6) does not necessarily hold. The following Theorem 1, 2 describe the output trajectory when we use D-type and PD-type learning rule for the system (1) with same non-zero initial errors at each iteration. The proof of Theorem 1, 2 were provided by Lee et al. [8].

**Theorem 1:** Suppose that \( \gamma \) is chosen such that (5) holds, and that the D-type learning rule (2) is repetitively applied to (1). If the initial errors remain the same at each iteration, i.e. \( x_k(0) = x_0, k = 0,1,2,\ldots \), then

\[
\lim_{k \to \infty} y_k(t) = y_d(t) + C x_0
\]

Theorem 1 shows that if the initial errors at each iteration remain the same, the output trajectory follows the desired trajectory with the offset of the initial errors. Now consider the following learning rule

\[
u_{k+1}(t) = u_k(t) + \gamma (\delta \dot{y}_k(t) - R \delta y_k(t)).
\]

**Theorem 2:** Suppose that (5) holds and the PD-type learning rule (8) is applied to (1). If the initial errors remain the same at each iteration, i.e. \( x_k(0) = x_0, k = 0,1,2,\ldots \), then

\[
\lim_{k \to \infty} y_k(t) = y_d(t) + e^{\rho t} C x_0
\]

Theorem 2 implies that if \( R \) is chosen such that \( \lambda(R) < 0 \), the learned control input enables the system to possess an asymptotic tracking capability even in the face of non-zero initial errors.
3. PTP problem and convergence when changing starting point

The conventional ILC focuses on tracking the desired trajectory for the fixed starting and goal points. However, it needs to train ILC when the initial error exists or user changes the initial condition arbitrarily. In this section we are going to investigate the usage of the desired input trajectory with the changing initial points. If this algorithm works, the pre-calculated desired input trajectory can be used to the changing points situation widely.

**Theorem 3:** Suppose that \( y \) is chosen such that (2) holds, and suppose that the desired input trajectory when \( x_d(0) = x_{d0} \) is applied to (1). If \( x_k(0) = x_o, k = 0, 1, 2, \cdots \), and system matrix \( A \) is sufficiently stable then the output trajectory error decreases exponentially fast.

**Proof:**

We can describe the \( y_k(t) \) from (1)

\[
y_k(t) = Ce^{At}x_0 + C \int_0^t e^{A(t-\tau)}Bu_k(\tau)d\tau.
\]

By assumption

\[
y_d(t) = Ce^{At}x_{d0} + C \int_0^t e^{A(t-\tau)}Bu_d(\tau)d\tau.
\]

If we substitute \( u_d(\tau) \) for \( u_k(\tau) \) into (10), then

\[
y_k(t) = Ce^{At}x_0 + C \int_0^t e^{A(t-\tau)}Bu_d(\tau)d\tau
\]

\[
= Ce^{At}x_0 + y_d(t) + Ce^{At}x_{d0}.
\]

As a result, we can obtain the following error dynamics.

\[
e_k(t) = y_k(t) - y_d(t) = Ce^{At}(x_0 - x_{d0}).
\]

According to the assumption, if matrix \( A \) is sufficiently stable, the output trajectory error decreases exponentially fast.

Theorem 3 shows that we can change the starting point easily with non-zero initial errors. Also, this algorithm works for any type of learning rule that is applied to the system (1). According to (13), the convergence rate depends on the eigenvalues of the system matrix \( A \). If \( \lambda(A) \ll 0 \), the output error becomes small. If the final time \( T \) becomes large, the output error also becomes small. Although the initial errors vary, this algorithm shows a same convergence rate to the various initial errors. However, in this algorithm, the system matrix \( A \) has to be a Hurwitz for convergence and there is no parameters to adjust for convergence rate. If we want to adjust the convergence rate, we have to move the pole location of the system using by feedback information. But, in order to use the feedback information, we have to know the precise system information. Therefore it is recommended to use this algorithm for a stable plant.

Let’s find out the total computation cost. If there are \( M \) starting points and \( N \) goal points, conventional ILC needs \( MN \) desired trajectories to cover whole trajectories. However, if we use the proposed algorithm SPILLA, we can cover the whole trajectories with only \( N \) desired trajectories. That is, the previously calculated values can be reused. Therefore the total computation cost can be reduced efficiently. In case of \( M \gg N \), this algorithm will be very effective.

Following example describes the SPILLA

**Example 1.** Consider the system (1) with

\[
A = \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C = [0, 1].
\]

Suppose the desired trajectory is given as

\[
y_d(t) = 12t(1 - t), 0 \leq t \leq 1.
\]
The P-type learning rule \( u_{k+1}(t) = u_k(t) + \gamma \delta y_k(t) \) is used, value of \( \gamma \) is 0.7 and the iteration number is 50. Figure 1 shows the output trajectory with non-zero initial error converges to the desired trajectory when we use the SPILLA. The dotted line is a trajectory of conventional ILC when the initial error is zero and it means that 50 iterative training have been done. On the other hand, the dashed line (Proposed) is trajectory of SPILLA when the initial error is 1, i.e. \( y_k(0) = 1, k = 0,1,2,\ldots \), and it means it is possible to reuse the desired input trajectory of ILC when initial condition changes 0 to 1. Therefore, if we want to change the starting point, we can adjust it by using initial errors.

![Figure 1. Example of SPILLA.](image)

Now, we consider the PTP control problems. In the PTP problem which is no need to consider the whole trajectory, we can suggest the following learning rule.

\[
\begin{align*}
  u_{k+1}(0) &= u_k(0) + \gamma \delta y_k(T), \\
  \text{where } T &\text{ is a final time of the PTP system. The convergence of the learning rule (15) is given in the following Theorem.}
\end{align*}
\]

**Theorem 4:** Suppose that the learning rule (15) is applied to (1) and the initial errors are zero at each iteration, i.e. \( x_k(0) = x_{d0}, k = 0,1,2,\ldots \). If

\[
\| I_r - \gamma e^{AT}B \|_\infty \leq \rho < 1
\]

then,

\[
\lim_{k \to \infty} y_k(T) = y_d(T).
\]

**Proof:** Let \( u_d(t) \) be a control input such that

\[
\begin{align*}
  y_d(T) &= Ce^{AT}x_{d0} + C \int_0^T e^{A(T-t)}Bu_d(t)\, dt \\
  &= Ce^{AT}x_{d0} + Ce^{AT}Bu_d(0)
\end{align*}
\]

The integral term can be represented as (16) because \( u_d(t) \) exists only when \( t \) is zero. The proof is completed if it is possible to show that \( \lim_{k \to \infty} \delta y_k(0) = \delta y_k(T) \) then the main idea is to prove that \( \lim_{k \to \infty} \| \delta u_k(0) \|_\infty = 0 \). Then it follows from (15) and (18) that

\[
\begin{align*}
  \delta u_{k+1}(0) &= u_d(0) - u_k(0) + \gamma \delta y_k(T) \\
  &= \delta u_k(0) - \gamma [Ce^{AT}x_{d0} + Ce^{AT}Bu_d(0) - Ce^{AT}x_{d0} + Ce^{AT}Bu_k(0)] \\
  &= (I_r - \gamma Ce^{AT}B)\delta u_k(0)
\end{align*}
\]

Taking the norm \( \| \cdot \|_\infty \) on both sides of (19), we have

\[
\| \delta u_{k+1}(0) \|_\infty \leq \| I_r - \gamma Ce^{AT}B \|_\infty \cdot \| \delta u_k(0) \|_\infty
\]
Since it is assumed that $\|I_r - \gamma C e^{AT} B\|_\infty \leq \rho < 1$. Thus
$$\lim_{k \to \infty} \|\delta u_k(0)\|_\infty = 0$$
\begin{equation}
\therefore \lim_{k \to \infty} y_k(T) = y_d(T) \tag{22}
\end{equation}
This completes the proof.

4. Simulation results
We verify the main result for a single starting and goal point problem with initial errors and the simulation was performed in case of multiple starting and goal points problem. Finally, in order to verify the proposed algorithm is valid not only one-dimensional problem but also multi-dimensional problem, 3-dimensional PTP example is performed.

Figure 2 and 3 show that main result working for P and D-type learning rules, respectively. Consider the system (1) with
$$A = \begin{bmatrix} 0 & 1 \\ -1 & -5 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C = \begin{bmatrix} 0 & 1 \end{bmatrix}. \tag{23}$$
Suppose the desired trajectory is given as
$$y_d(t) = 12 \sin^3(4t), 0 \leq t \leq 1$$
and let there are non-zero initial errors from -3 to 3. The $\gamma = 0.7, R = -5$, iteration numbers are 100, 15 for the P-type, D-type learning rule respectively. The solid line is desired trajectory and the dotted line is trajectory with ILC when the initial error is zero. Dashed lines are trajectories with the proposed algorithm when the initial errors are exists. We reuse the desired input information getting from the dotted line for SPILLA. According to SPILLA, trajectories with non-zero initial errors converge to the desired trajectory with same convergence rate.

Figure 2. P-type learning rule for SPILLA

Figure 3. D-type learning rule for SPILLA

Figure 4. and 5. show the multiple starting and goal points problem. The system is same with above simulation. The desired trajectory is given as $y_d(t) = 12(t(1-t)), 0 \leq t \leq 1$. There are three starting points and goal points. As for whole 9 trajectories, 6 trajectories (dashed lines) are determined by SPILLA. It means only 3 trajectories are trained with the conventional ILC.
Now, we can expand this proposed algorithm to the multi-dimensional problem. Let’s consider the 3-dimensional PTP problem.

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\dot{x}_3 \\
\dot{v}_1 \\
\dot{v}_2 \\
\dot{v}_3
\end{bmatrix} = 
\begin{bmatrix}
-1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 1 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
v_1 \\
v_2 \\
v_3
\end{bmatrix} + 
\begin{bmatrix}
0 \\
0 \\
0 \\
1/m \\
1/m \\
1/m
\end{bmatrix}
\begin{bmatrix}
y_1 \\
y_2 \\
y_3 \\
y_4 \\
y_5 \\
y_6
\end{bmatrix} + 
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
-mg
\end{bmatrix}
\begin{bmatrix}
x_5 \\
x_6 \\
x_7 \\
x_8 \\
x_9 \\
x_{10}
\end{bmatrix}
\tag{24}
\]

where \( m = 0.5kg, T = 5sec, g = \frac{9.8m}{s^2}, \gamma = 0.5 \) for all dimensions and P-type learning rule is used. The starting point is (0,0,0) and goal point is (50, 50, 20). The initial force is (0,0,0) and using the learning rule given as (13), the initial force \((F_x, F_y, F_z)\) is updated. The initial force effect to the system with very short time. Note that the given system (24) has 3-dimension and each dimension has a negative eigenvalue respectively. Therefore we can apply the SPILLA to this system.

Figure 6 shows the output trajectory when the PTP learning rule in (15) is applied in (24). The iteration number is 100. Then we can expect that the starting point can be changed with non-zero initial errors. Figure 7 presents that the each trajectory starting from several other points can converge to the same goal point by using SPILLA.
5. Conclusion
In this paper, we proposed a starting point independent iterative learning algorithm on LTI system with non-zero initial errors. This algorithm reduced the computation cost significantly. Also, the convergence rate for all starting points are the same. Then we suggested a PTP learning rule and showed its convergence. Validity of these algorithms are shown by computer simulation with multi-dimensional problem.

References
[1] Lee HS and Bien Z 2002 Intelligent Auto. Soft Computing. 8 95-106
[2] Kuc TY and Lee JS 1992 Automatica 28 1215-1221
[3] Chen Y, Wen C, Gong Z and Sun M 1999 IEEE Trans. Auto. Cont. 44 371-376
[4] Park KH and Bien Z 2000 Int. J. Cont. 73 871-881
[5] Sun M and Wang D 2003 IEEE Trans. Auto. Cont. 48 144-148
[6] You B, Kim M, Lee D, Lee J and Lee J 2011 Cont. Eng. Prac. 19 234-242
[7] Heinzinger G, Fenwick D, Paden B and Miyazaki F 1992 IEEE Trans. Auto. Cont. 37 110-114
[8] Lee HS and Bien Z 1996 Int. J. Cont. 64 345-359
[9] Arimoto S, Kawamura S and Miyazaki F 1984 J. Robotic Sys. 1 123-140