INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, e-ISSN 1841-9844, 14(3), 359-374, June 2019.

Effect of Soft Errors in Iterative Learning Control and Compensation using Cross-layer Approach

G.-M. Jeong, K. Lee, S.-I. Choi, S.-H. Ji, N. Dutt

Gu-Min Jeong
School of Electrical Engineering
Kookmin University, Seoul, Korea
gm1004@kookmin.ac.kr

Kyoungwoo Lee*
Department of Computer Science
Yonsei University, Seoul, Korea
*Corresponding author: kyoungwoo.lee@gmail.com

Sang-Il Choi
Department Applied Computer Engineering
Dankook University, Korea
choisi@dankook.ac.kr

Sang-Hoon Ji
Advanced Robotics Group
KITECH, Ansan, Korea
robot91@kitehc.re.kr

Nikil Dutt
Department of Computer Science
UC Irvine, CA, USA
dutt@ics.uci.edu

Abstract: In this paper, we study the effects of radiation-induced soft errors in iterative learning control(ILC) and present the compensation techniques to make the ILC systems robust against soft errors. Soft errors are transient faults, which occur temporarily in memories where the energetic particles strike the sensitive region in the transistors mainly under abnormal conditions such as high radiation, high temperature, and high pressure. These soft errors can cause bit value changes without any notification to the controller, affect the stability of the system, and result in catastrophic consequences. First, we investigate and analyze the effects of soft errors in the ILC systems. Our analytical study shows that when a single soft error occurs in the output data from the ILC, the performance of the learning control is significantly degraded. Second, we propose novel learning methods by incorporating the existing techniques across the system abstraction levels in the ILC to compensate for soft-error-induced incorrect output. The occurrence of soft errors is estimated by using a monotonic convergence of the erroneous outputs in a cross-layer manner, and our proposed methods can significantly reduce these negative impacts on the system performance. Under the assumption of soft error occurrence, our analytic study has proved the convergence of the proposed methods in the ILC systems and our simulation results show the effectiveness of the proposed methods to efficiently reduce the impacts of soft-error-induced outputs in the ILC systems.
Keywords: Soft error, iterative learning control, compensation.

1 Introduction

With the advances in the highly integrated semiconductor and microelectronic manufacturing techniques, the variability of electronic devices has drastically increased in real time. On the

other hand, soft errors or transient faults induced by various sources (e.g., heat and cosmic radiation) have become one of the critical threats to the system stability [4]. Soft errors, different from hard errors, which are permanent faults in the system and cannot be corrected without replacement, are difficult to detect and can be as critical as hard errors to the system reliability in certain application areas [4, 10, 14, 15, 17, 19, 22]. The probability of soft error occurrence can be extremely low because these errors generally occur under abnormal situations. However, a single soft error can result in a bit value change, that is, from 0 to 1, or vice versa, in the memory without notifications to the application if there is no error detection or correction mechanism. Thus, depending on the bit location of the soft error, it can be critical to the stability of the controller.

Until now, there have been several researches for fault tolerant techniques to reduce the negative impacts of the soft errors on embedded systems. In particular, cross-layer-based approaches have been considered as promising techniques to analyze the trade-off among the performance, power, and reliability and have been presented to maximize the reliability with minimal overheads of the power and performance in various embedded systems [3, 8, 12, 13, 21].

Iterative learning control(ILC) systems have been widely deployed and used for various conventional and emerging control applications [1, 2, 5, 7, 9, 16, 20]. A single error or an incorrect output can cause a catastrophic consequence because these control systems can be used for mission- and safety-critical systems. However, no research has been conducted to protect the ILC systems from soft errors, to the best of our knowledge, while the soft error rate is exponentially increasing with technology scaling. To this end, we analyze the effects of soft errors in the ILC and propose compensation methods to increase the reliability of the ILC systems against soft errors, in particular, techniques in a cross-layer manner. First, we discuss the impacts of a single soft error on the output in the ILC systems and show that the soft error significantly affects the performance of the learning control. For an intensive analysis, we have selected an example with 4-byte floating point number [6] and have shown that a soft error in the ILC systems can lead to a large number of errors in the value and incorrect outputs eventually as threats to the system stability and reliability. Indeed, a soft error can significantly reduce the speed of the learning curve in a linear discrete-time system. Therefore, it is required to design and develop mitigation techniques to protect the ILC systems from soft errors. Second, we propose novel compensation methods against soft errors for the ILC systems. We have made a few assumptions about the soft errors in the ILC systems and investigated the new detection and compensation methods. In particular, from a monotonic convergence of the output error, we have estimated the occurrence of the critical soft errors that significantly affect the system output and result in catastrophic consequences. We propose four compensation methods: (i) exact rollback method, (ii) recovery with the previous iteration value, (iii) recovery with the adjacent value, and (iv) recovery with the desired value. In addition, the methodology and effectiveness of these four compensation techniques have been evaluated on the basis of their advantages and disadvantages. Based on the assumptions that we have made for the soft errors in this study, the convergence of each method has been presented. Our extensive simulation results show that the negative impacts of the soft errors can be successfully canceled with the proposed compensation methods.

The remainder of this paper is organized as follows. In Section 2, related works are briefly summarized for the soft-error related problems and the ILC for discrete-time systems. In Section 3 and 4, the effect of the soft error in the ILC is discussed and a new learning method to compensate for the effect of the soft error is introduced, respectively. An illustrative example is described in Section 5 and the conclusion follows in Section 6.

2 Related works

In this section, the preliminary results for the researches on the soft-errors and the ILC are briefly discussed.

2.1 Soft errors

Several decades of technology scaling have brought us to a point where transistors have become extremely susceptible to even small fluctuations in the voltage levels, slight noise in the power supply, signal interference, and cosmic particle strikes [4]. Any of these effects can temporarily toggle the logic value of a transistor, and therefore, it is called a transient fault or soft error. These soft errors are not permanent and nondestructive, that is, resetting of the device restores the normal behavior. An energetic particle such as an alpha particle, a neutron, or a free proton, can form a diffusion region of a CMOS transistor and produce a charge that can result in toggling of the logic value of the gates or flip-flops. This phenomenon of change in the logic state is called a transient fault or a soft error. Soft errors can result in erroneous program states, incorrect outputs, and, eventually, system crashes.

Soft errors have already been proved to cause significant fiscal damages [22]. For example, SUN blamed the soft errors for the crash of their million-dollar line SUN flagship servers [14]. More recently, Hewlett Packard acknowledged that a large installed base of a 1024-CPU server system in the Los Alamos National Laboratory has been frequently crashing owing to soft errors [15]. In another incident, the soft errors brought a billion-dollar automotive factory to halt every month [10], [17]. Further, highly integrated reliability-sensitive embedded devices such as mobile health-care systems and anti-lock braking systems (ABSs) in the automotive engine control units (ECUs) are significantly threatened by the exponentially increasing soft error rates with technology scaling. Thus, it is necessary to combat the soft errors in embedded systems in both emerging and traditional computing environments.

Recently, several selective protection techniques have been proposed in order to combat soft errors in a cost-efficient manner by protecting the failure critical data only [11, 13, 18, 21]. For instance, partially protected caches(PPC) [11] utilizes the knowledge of the content and device hardware capabilities to selectively store the critical data in a more reliable hardware (e.g., protected cache) and non-critical data in a less reliable one (e.g., non-protected cache). In [3], a novel memory cell design method considering robustness to soft-errors has been proposed for a deep learning accelerator.

Cross-layer-based techniques have been presented to maximize protection at a minimal cost by coupling and cooperating different error control schemes across the system abstraction layers from the hardware layer to the application layer in mobile embedded systems [12]. In [8], a cross-layer approach has been introduced to robust face recognition against soft errors. However, these techniques have been proposed to mitigate the impacts of the soft errors on domain specific embedded systems or general computer systems, and not control systems, in particular.

2.2 ILC for linear discrete time system

Some of the preliminary results for the ILC are briefly discussed, considering the monotonic convergence of the output errors [5]- [9].

We consider a linear time invariant(LTI) system described by

$$\begin{aligned} x(i+1) &= Ax(i) + Bu(i) \\ y(i) &= Cx(i) \end{aligned} \tag{1}$$

where, $u \in \mathbb{R}^1$, $x = [x_1, \cdots, x_n]^T \in \mathbb{R}^n$, and $y \in \mathbb{R}^1$ are the input, the state, and the output of the system, respectively. A, B and C are matrices of appropriate dimensions. Let $u^d(i)$, $x^d(i)$, and $y^d(i)$ represent the input, the state, and the output corresponding to the desired trajectory. Also let the desired output $y^d(i), i \in [\sigma, N+\sigma-1]$, be given and $\mathbf{u}_{[i,j]} := [u(i), \cdots, u(j)]^T, \mathbf{y}_{[i,j]} := [y(i), \cdots, y(j)]^T$. Here σ denotes the relative degree, which means that $CA^{\sigma-1}B \neq 0$ and $CA^iB = 0, i < \sigma - 1$.

The transfer function of the system is represented as

$$G(z) = \frac{\beta_1 z^{n-1} + \cdots + \beta_n}{z^n + \alpha_1 z^{n-1} + \cdots + \alpha_n}. \tag{2}$$

Here, we assume that the relative degree, σ, is known as *a priori* (i.e., $\beta_1 = \cdots = \beta_{\sigma-1} = 0$).

Based on the relative degree, we can obtain the input-to-output relation as

$$y(i+\sigma) = CA^\sigma x(i) + CA^{\sigma-1}Bu(i). \tag{3}$$

In addition, the following input-to-output mapping is used for ILC:

$$\begin{aligned} \mathbf{y}_{[\sigma,N+\sigma-1]} &= \mathbf{H}x(0) + \mathbf{J}\mathbf{u}_{[0,N-1]}, \\ \mathbf{H} &= \left[\ (H_0)^T \ , \ \cdots \ , \ (H_{N-1})^T \ \right]^T, \\ \mathbf{J}_{d_0} &= \begin{bmatrix} J_0 & 0 & \cdots & 0 \\ J_1 & J_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ J_{N-1} & J_{N-2} & \cdots & J_0 \end{bmatrix}, \end{aligned} \tag{4}$$

where $H_l = CA^\sigma$ and $J_l = CA^{\sigma+l-1}B$. The time interval for the output of interest is $[\sigma, N+\sigma-1]$ for minimum phase systems. Also, at every iteration, we set $x^k(0) = x^d(0)$.

Considering (4), an input update law is given as

$$\mathbf{u}^{k+1}_{[0,N-1]} = \mathbf{u}^k_{[0,N-1]} + \mathbf{S}\mathbf{e}^k_{[\sigma,N+\sigma-1]}, \tag{5}$$

where $\mathbf{e}^k_{[l,m]} = \mathbf{y}^d_{[l,m]} - \mathbf{y}^k_{[l,m]}$ and $\mathbf{S} \in \mathbb{R}^{N \times N}$ is the learning gain matrix.

The monotonic convergence of the above ILC method is shown in the following lemma.

Lemma 1. *For the uncertain system (1), if the condition*

$$||I - \mathbf{J}\mathbf{S}|| \leq \rho < 1, \tag{6}$$

holds for all k, the output error $\mathbf{e}^k_{[\sigma,N+\sigma-1]}$ monotonically converges to 0 as $k \to \infty$. That is,

$$||\mathbf{e}^{k+1}_{[\sigma,N+\sigma-1]}|| \leq \rho ||\mathbf{e}^k_{[\sigma,N+\sigma-1]}||, \lim_{k\to\infty} ||\mathbf{e}^k_{[\sigma,N+\sigma-1]}|| = 0. \tag{7}$$

Proof: From (4) and (5), we can obtain

$$\begin{aligned} \mathbf{e}^{k+1}_{[\sigma,N+\sigma-1]} &= \mathbf{J}\mathbf{u}^d_{[0,N-1]} - \mathbf{J}\mathbf{u}^{k+1}_{[0,N-1]} \\ &= \mathbf{J}\mathbf{u}^d_{[0,N-1]} - \mathbf{J}(\mathbf{u}^k_{[0,N-1]} + \mathbf{S}\mathbf{e}^k_{[\sigma,N+\sigma-1]}) \\ &= [I - \mathbf{J}\mathbf{S}]\mathbf{e}^k_{[\sigma,N+\sigma-1]}. \end{aligned}$$

Taking the norms on the both sides and from (6), we can obtain (7). □

3 Effect of soft error in iterative learning control

In this section, we analyze the effect of soft errors on the ILC. It is shown that the learning performance is significantly affected by the soft errors in a linear discrete time system.

3.1 Illustrative example of ILC

Figure 1 shows the basic structure of the ILC [1]. As shown in Figure 1 and equation (5), the input of the next iteration is determined from the output error and input of the current iteration. Since a large amount of memory should be used for the ILC, it can be easily affected by the soft errors in the memory.

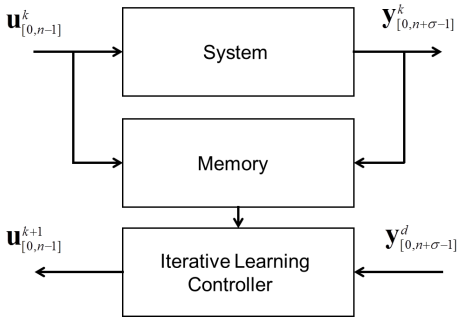


Figure 1: Basic ILC configuration for discrete time systems (modified from the figure in [1])

For the analysis of the effect of soft errors, let us consider an example of a linear discrete-time single-input and single-output(SISO) systems.

Let us consider the following linear discrete-time system:

$$\begin{aligned} \mathbf{x}(i+1) &= \begin{bmatrix} 0 & 0 & -0.1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{x}(i) + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(i) \\ y(i) &= \begin{bmatrix} 1 & 0.5 & 0.0625 \end{bmatrix} \mathbf{x}(i). \end{aligned} \tag{8}$$

This system has two minimum-phase zeros ($z = -0.25$) and the relative degree of the system is 1. The desired trajectory is given as follows:

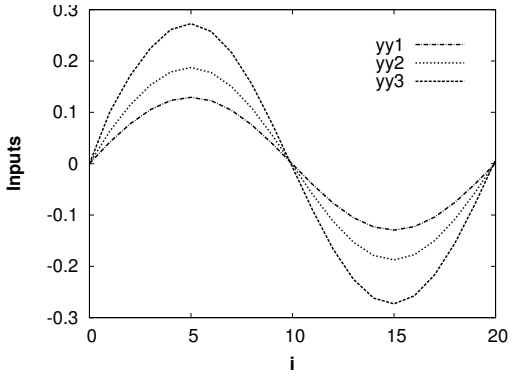
$$y^d(i) = \begin{cases} 0, & i = 0 \\ 0.2\sin(0.1\pi(i-1)), & 1 \le i \le 21. \end{cases} \tag{9}$$

Here, we set $N = 21$, $u(21) = 0$ and $\mathbf{S} = \alpha I_{21\times 21} = 0.1 I_{21\times 21}$. Since the system (8) and the learning gain α satisfy the conditions $||J_0|| > \sum_{i=1}^{N-1} ||J_i||$ with $1 > 0.7361$ and $|1 - \alpha J_0| < 1$ with $0.9 < 1$, which are sufficient conditions for the convergence in [16], the convergence condition $||I - 0.1\mathbf{J}|| < 1$ is satisfied.

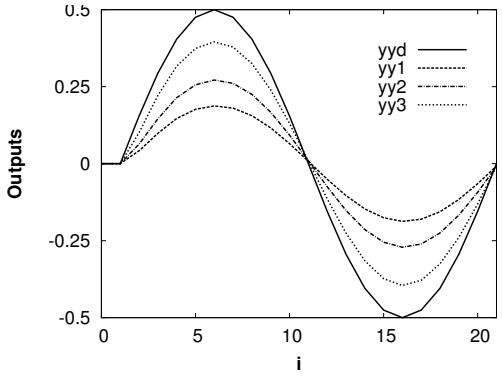
The input update law is expressed as

$$\mathbf{u}^{k+1}_{[0,20]} = \mathbf{u}^k_{[0,20]} + 0.1\mathbf{e}^k_{[1,21]}. \tag{10}$$

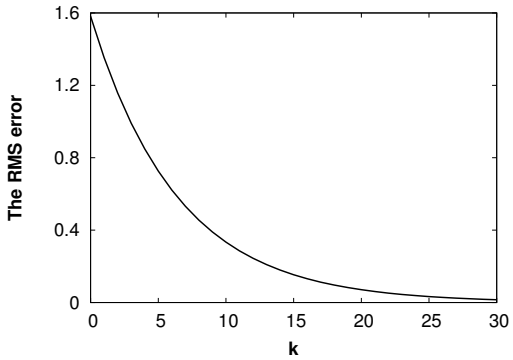
Figure 2 shows the outputs and inputs when there is no soft error. As shown in Figure 2(c), the root mean square(RMS) error for the output error approaches 0 as $k \to \infty$.



(a) Inputs for different values of k



(b) Desired output and the outputs for different values of k



(c) RMS error for different values of k

Figure 2: Outputs and inputs when there is no soft error

3.2 Effect of soft errors to ILC

Let us assume that a soft error occurs in the memory. If we assume that there is single bit error in the 4-byte floating point variable, the effect of the soft error is different, depending on the bit location among the 32 bits where a single soft error occurs. Table 1 summarized the structure of the 4-byte floating point variable [6].

Table 1: Binary format for floating point variable with single precision (32bits) [6]

Parts	Sign	Exponent	Fraction
Number of bits	1	8	23

Table 2 presents an example of the effect of the single soft error on the 4-byte floating point variable with a value of 0.2. As summarized in Tables 1 and 2, the soft errors in the sign bit, exponent part, and the upper part of the fraction lead to a large number of errors introduced and, eventually, can be critical to the control system. In real embedded systems, 1-bit error can be overcome by ECC(Error Correction Code). However when there are 3 or more bit errors, ECC cannot check those errors. Also the error values can be larger than that of 1-bit error. Assuming 1-bit error in MSB can show the effect of soft-errors in ILC. In order to show the effect of soft-errors, we assume just 1-bit error for the simulation without loss of generality.

By considering above mentioned aspects, let us discuss the effects of the soft error on the

Table 2: Effect of soft error on the 4-byte floating point variable with a value of 0.2

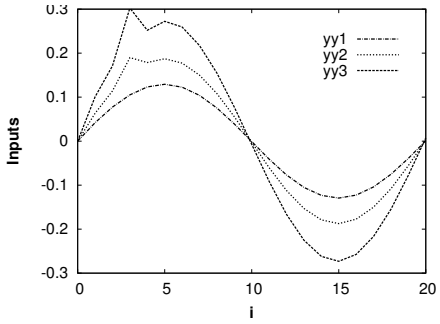
Part	Error bit	Approximate vaule (≈)
Sign	1(MSB)	-0.200000
Exponent	2	6.805647×10^{35}
	3	0.000000
	4	0.000000
	5	0.000003
	6	0.000781
	7	0.012500
	8	0.800000
	9	0.400000
Fraction	10	0.137500
	11	0.231250
	12	0.215625
	13	0.192188
	14	0.196094
	15	0.201953
	16	0.200977
	17	0.199512
	18	0.199776
	19	0.200122
	20	0.200061
	21	0.199969
	22	0.199985
	23	0.200008
	24	0.200004
	25	0.199998
	26	0.199999
	27-32	0.200000

ILC of the previous example for system (8). For the previous example, let us assume that there are soft errors in the memory. To analyze the effect of soft errors, we also assume the followings:

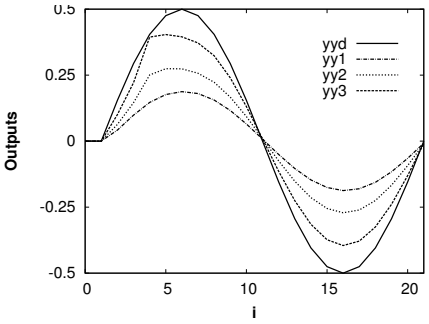
- 4-byte floating point variables are used for the storage and processing of all the data.
- 1-bit soft error occurs in $y^k(4)$ in the most significant bit(MSB) of the 4-byte memory.
- A soft error occurs at every 4-th iteration corresponding to the iteration number k.

With this setting as an example, since the MSB of the 4-byte floating number variable is the sign bit, the sign of $y^k(4)$ changes at every 4-th iteration. The learning is performed under these assumptions for the soft errors.

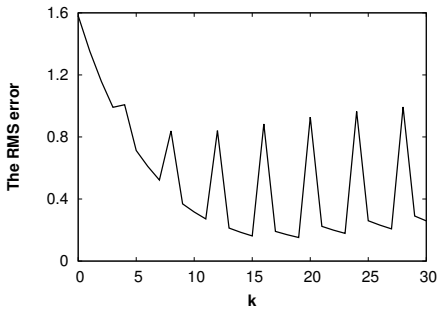
Figure 3 shows the effects of a soft error for the previous example. As shown in Figure 3, a soft error can significantly degrade the performance of the learning control. In other words, the soft errors reduce the learning speed and affect the convergence of the output error in the ILC.



(a) Inputs for different values of k



(b) Desired output and the outputs for different values of k



(c) RMS error for different values of k

Figure 3: Outputs and inputs when there are soft errors induced

4 Cross-layer approach to compensate for the soft error

In this section, we propose new detection and compensation methods for the soft errors in the ILC by obtaining the monotonic convergence with the cross-layer approach, as described in Figure 4. As summarized in Table 2, depending on the location of the soft error, its effect on the control system varies. The errors in the upper part of the 4-byte data format are critical to the control systems, while the other errors are negligible.

Similarly, with regard to the monotonic convergence of the output error, it is not required to detect all the soft errors. In this study, we detect the soft error that prohibits the monotonic convergence, and further, we compensate for the soft error with other relevant data available.

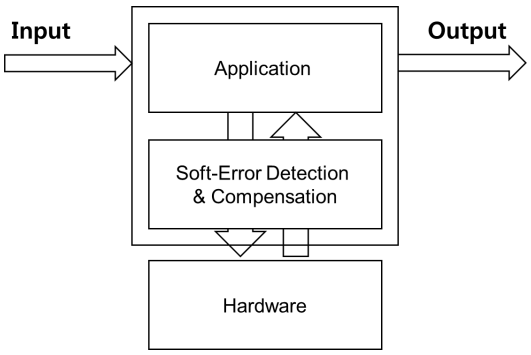


Figure 4: Compensation of soft error in a cross-layer manner

For further analysis, some assumptions are made by considering the soft error imposed on the system as follows:

- **(A1)** A soft error occurs in the system output, that is, y^k.
- **(A2)** At iteration k, only one soft error occurs. Hence soft-error occurs once at iteration k.
- **(A3)** There is no contiguous soft error corresponding to iteration k. In other words, when there is a soft error at iteration k, there is no soft error at $k+1$ and there may be a soft error at $k+2$.

With regard to (**A1**), soft errors can occur in the system, restored memories of the ILC, or the learning controller. Different from other variables, the system output y^k can have soft errors within all the components such as the system, restored memory, and learning controller. By considering these aspects, we make an assumption (**A1**).

Using (**A2**), we can detect the location of the soft error in the 4-byte floating point variable. Further, using (**A3**), we can derive the monotonic convergence with the output error of the previous iteration.

It should be noted that depending on the compensation method used, we can neglect or alleviate the related assumptions.

4.1 Detection of critical soft errors using cross-layer approach

If condition (6) is satisfied, $||\mathbf{e}^{k+1}_{[\sigma,N+\sigma-1]}|| \leq \rho||\mathbf{e}^{k}_{[\sigma,N+\sigma-1]}||$ should be guaranteed for some $\rho < 1$.

However, as shown in Figure 3, and Tables 1 and 2, depending on the location of the soft error, the monotonic convergence cannot be guaranteed.

Among all the soft errors, we only detect those errors affecting the monotonic convergence. Therefore, when $||\mathbf{e}^{k+1}_{[\sigma,N+\sigma-1]}|| \geq ||\mathbf{e}^{k}_{[\sigma,N+\sigma-1]}||$, we can conclude that a critical soft error has occurred. After detecting a soft error, we can find the location and compensate for the soft error.

Considering these facts, we propose a soft error detection algorithm to find the location of error j and iteration k for the ILC as follows :

Detection of critical soft errors in a cross-layered manner

- Step 1: For the learning curve, validate whether $||\mathbf{e}^k||_\infty \geq ||\mathbf{e}^{k-1}||_\infty$.
- Step 2: If the error $||\mathbf{e}^k||_\infty$ is larger than or equal to $||\mathbf{e}^{k-1}||_\infty$, it is determined that a critical soft error occurs at iteration k.
- Step 3: Determine j, which makes $||e^k(j)|| = ||\mathbf{e}^k||_\infty$, that is, the location of the largest value at iteration k.

The following lemma shows that we can detect the location of the soft error with the assumptions (**A1**)-(**A3**).

Lemma 2. *(Determining time j and iteration k, where soft error occurs.)*
*Let us assume that a soft error satisfying (**A1**)-(**A3**) occurs in the process of the ILC in the uncertain system (1). Using the above algorithm, the location of the soft error, that is, time j and iteration k can be determined.*

Proof: If there is no soft error, (7) holds for $\mathbf{e}^k$ and $\mathbf{e}^{k-1}$. However, if there is a critical soft error, we can obtain $||\mathbf{e}^k|| \geq ||\mathbf{e}^{k-1}||$. Thus, we can conclude that a critical soft error occurs at iteration k. □

When there is a critical soft error at iteration k, let us assume that the soft error occurs in j, that is, $e^k(j)$ contains the soft error.

Let us denote $\tilde{e}^k(j)$ as the real output error when there is no soft error, and $\tilde{\mathbf{e}}^k$ by replacing $e^k(j)$ with $\tilde{e}^k(j)$ in $\mathbf{e}^k$. From the definition of the ∞-norm, we can obtain the following equations:

$$\begin{aligned} &||\mathbf{e}^k||_\infty \\ &= \max\{||e^k(\sigma)||, \cdots, ||e^k(j)||, \cdots, ||e^k(N+\sigma-1)||\} \end{aligned} \tag{11}$$

$$\begin{aligned} &||\tilde{\mathbf{e}}^k||_\infty \\ &= \max\{||e^k(\sigma)||, \cdots, ||\tilde{e}^k(j)||, \cdots, ||e^k(N+\sigma-1)||\} \end{aligned} \tag{12}$$

Further, according to the relationship among $\mathbf{e}^k$, $\tilde{\mathbf{e}}^k$, and $\mathbf{e}^{k-1}$, the following equations hold:

$$||\mathbf{e}^k||_\infty \geq ||\mathbf{e}^{k-1}||_\infty, ||\tilde{\mathbf{e}}^k||_\infty < ||\mathbf{e}^{k-1}||_\infty. \tag{13}$$

Therefore, to satisfy (13), the following equation should be guaranteed by using (11) and (12).

$$||\mathbf{e}^k||_\infty = ||e^k(j)|| \tag{14}$$

Therefore, if we select the maximum value among the output errors at iteration k, we can determine j.

4.2 Compensation of soft errors

After detecting a soft error in $y^k(j)$, we can compensate for the soft error in several ways. For instance, as shown in Figure 5, we can use the stored data for $y^k(j)$, the value of the previous iteration $y^{k-1}(j)$, the adjacent value $y^k(j-1)$, or the desired output $y^d(j)$.

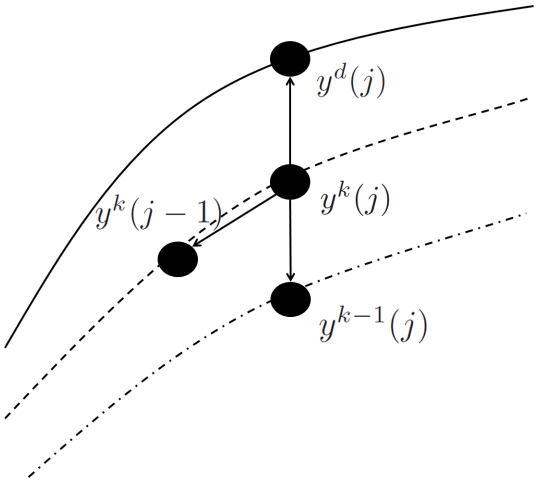


Figure 5: Recovery of the soft error induced $y^k(j)$ with related data

In general, for the resilient design of a soft error, the exact rollback method has been widely used. After periodically copying the value of $\mathbf{y}^k$ to the buffers, we can restore the value when an error is detected. Although we need additional buffer spaces for the rollback, it is a simple

and effective technique to reduce the memory errors. If advanced error-detection methods can be supported, it can be used without assumptions (**A2**) and (**A3**).

However, with regard to (**A1**), this method cannot be used when the soft error occurs within the system, that is, before copying the data to the buffer. For the soft errors in the memory or learning controller, the rollback method can be used irrespective of the variables.

The value of the previous iteration $y^{k-1}(j)$ can be used to compensate for $y^k(j)$. Since the learning is performed according to the iteration number k, it is easy to prove the monotonic convergence. In addition, it can be used when there is a soft error within the system or before copying the data to the buffer.

Further, we can use the adjacent value $y^k(j-1)$. For real-time control, this approach can be used for the compensation of soft errors. In this example, we do not need additional buffers. If more advanced error-detection methods can be supported, it can be used, relaxing assumptions (**A2**) and (**A3**).

Otherwise, the desired value $y^d(j)$ can be applied for the compensation. Since the error $e^k(j)$ becomes zero, monotonic convergence can be derived.

Based on these facts, we can summarize the compensation method as follows:

Compensation method 0 - exact rollback

- Step 1: Restore the measured output value $\mathbf{y}^k$ to an additional buffer at iteration k.
- Step 2: When a soft error is detected at iteration k and time j, recover it with $y^k(j)$ from the buffer.

Compensation method 1- using the value of the previous iteration $y^{k-1}(j)$

- Step 1: Restore the output value $\mathbf{y}^{k-1}$ to the memory at iteration $k-1$.
- Step 2: When a soft error is detected at iteration k and time j, replace $y^k(j)$ with $y^{k-1}(j)$.

Compensation method 2- using the adjacent value $y^k(j-1)$

- Step 1: When a soft error is detected at iteration k and time j, replace $y^k(j)$ with $y^k(j-1)$.

Compensation method 3- using the desired value $y^d(j)$

- Step 1: When a soft error is detected at iteration k and time j, replace $y^k(j)$ with $y^d(j)$.

With the presented compensation methods, monotonic convergence is proved in the theorems below. For compensation methods 1 and 3, convergence is given based on (**A1**)-(**A3**), whereas, for the compensation method 2, we need another condition for the magnitude of the compensated error.

Theorem 3. *(Convergence of the ILC when using compensation method 1)*
*Let us assume that a soft error satisfying (**A1**)-(**A3**) occurs at the process of the ILC in the uncertain system (1). Let us apply the input update law (5), the critical soft error detection method, and compensation method 1 to the uncertain system (1). If the condition (6) holds for all k, the output error $||e^k_{[\sigma,N+\sigma-1]}||_\infty$ converges to 0 as $k \to \infty$.*

Proof: Let us consider that a soft error occurs in $y^k(j)$. That is, $||\mathbf{e}^k||_\infty \geq ||\mathbf{e}^{k-1}||_\infty$ and $||\mathbf{e}^k||_\infty = ||e^k(j)||$.

From **Lemma 2**, we can detect j and k. Further, we compensate for the data using $y^{k-1}(j)$ instead of $y^k(j)$. Let us denote the compensating value $y^{k-1}(j)$ as $\hat{y}^k(j)$ and the compensated k-th iteration data as $\hat{\mathbf{e}}^k$, respectively. From these values, we obtain $\hat{\mathbf{e}}^k = [e^k(\sigma), \cdots, \hat{e}^k(j), \cdots, e^k(N+\sigma-1)]$, where $\hat{e}^k(j) = y^d(j) - \hat{y}^k(j) = y^d(j) - y^{k-1}(j) = e^{k-1}(j)$.

since $||\hat{e}^k(j)|| = ||e^{k-1}(j)||$, from (13) and (14), we can obtain

$$||\hat{\mathbf{e}}^k||_\infty \leq ||\mathbf{e}^{k-1}||_\infty. \tag{15}$$

Further, since there is no soft error at iteration $k+1$ from (**A3**), the following equation holds:

$$||\mathbf{e}^{k+1}||_\infty < \rho||\hat{\mathbf{e}}^k||_\infty, \rho < 1. \tag{16}$$

From (15) and (16), the output error approaches 0 as $k \to \infty$.

□

Theorem 4. *(Convergence of the ILC when using compensation method 2)*
*Let us assume that a soft error satisfying (**A1**)-(**A3**) occurs at the process of ILC in the uncertain system (1). Let us apply the input update law (5), the critical soft error detection method and compensation method 2 to the uncertain system (1). If the compensated error satisfies $||y^d(j) - y^k(j-1)|| \leq ||\mathbf{e}^{k-1}||_\infty$ and condition (6) holds for all k, the output error $||e^k_{[\sigma,N+\sigma-1]}||_\infty$ converges to 0 as $k \to \infty$.*

Proof: We can detect j and k with the detection method. Let us compensate for the data using $y^k(j-1)$ instead of $y^k(j)$. Let us denote the compensating value $y^k(j-1)$ as $\hat{y}^k(j)$ and the compensated k-th iteration error as $\hat{\mathbf{e}}^k$, respectively. From these values, we obtain $\hat{\mathbf{e}}^k = [e^k(\sigma), \cdots, \hat{e}^k(j), \cdots, e^k(N+\sigma-1)]$, where $\hat{e}^k(j) = y^d(j) - \hat{y}^k(j) = y^d(j) - y^k(j-1)$.

If $||\hat{e}^k(j)|| = ||y^d(j) - y^k(j-1)|| \leq ||\mathbf{e}^{k-1}||_\infty$, from (13) and (14), we can obtain

$$||\hat{\mathbf{e}}^k||_\infty \leq ||\mathbf{e}^{k-1}||_\infty. \tag{17}$$

Further, since there is no soft error at iteration $k+1$ from (**A3**), the following equation holds:

$$||\mathbf{e}^{k+1}||_\infty < \rho||\hat{\mathbf{e}}^k||_\infty, \rho < 1. \tag{18}$$

From (17) and (18), the output error approaches 0 as $k \to \infty$.

□

Theorem 5. *(Convergence of the ILC when using compensation method 3)*
*Let us assume that a soft error satisfying (**A1**)-(**A3**) occurs in the process of the ILC in the uncertain system (1). Let us apply the input update law (5), the critical soft error detection method and compensation method 3 to the uncertain system (1). If the condition (6) holds for all k, the output error $||e^k_{[\sigma,N+\sigma-1]}||_\infty$ converges to 0 as $k \to \infty$.*

Proof: We can detect j and k with the detection method. Let us compensate for the data using $y^d(j)$ instead of $y^k(j)$. Then, for the compensated data at iteration k, we obtain $\hat{\mathbf{e}}^k = [e^k(\sigma), \cdots, \hat{e}^k(j) = 0, \cdots, e^k(N+\sigma-1)]$.

Hence, from (13) and $\hat{e}^k(j) = 0$, we can obtain

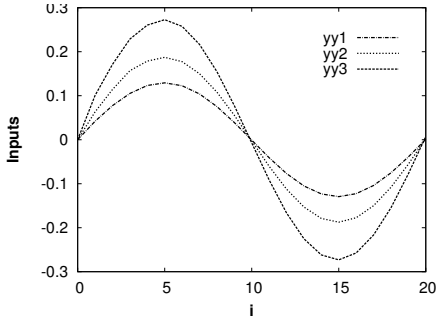
$$||\hat{\mathbf{e}}^k||_\infty \leq ||\tilde{\mathbf{e}}^k||_\infty < ||\mathbf{e}^{k-1}||_\infty. \tag{19}$$

Further, since there is no soft error at iteration $k+1$ from (**A3**), the following equation holds:

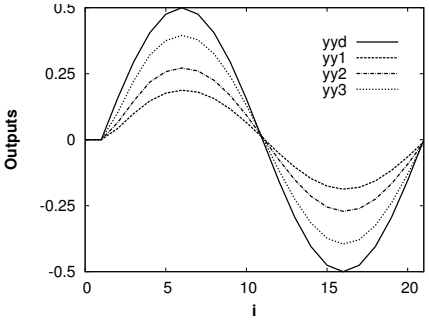
$$||\mathbf{e}^{k+1}||_\infty < \rho||\hat{\mathbf{e}}^k||_\infty, \rho < 1. \tag{20}$$

From (19) and (20), the output error approaches 0 as $k \to \infty$.

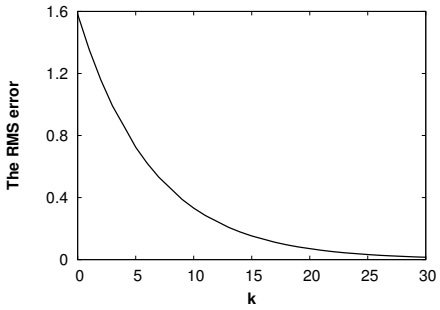
□



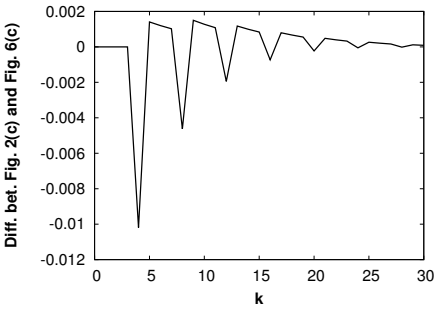
(a) Inputs for different values of k



(b) Desired output and the outputs for different values of k



(c) RMS error for different values of k



(d) Differences between the output errors when there is no soft error and when it is compensated using method 1

Figure 6: Outputs and inputs when soft error compensation method 1 is used

5 Simulation results

Let us apply the proposed methods to the case in Section 3.2. The soft error condition in the previous example satisfies assumptions (**A1**)-(**A3**) and we can apply the detection and compensation methods to this example.

Figure 6 shows the results of compensation method 1. As shown in Figure 6, we can observe that the output error converges to 0 as $k \to \infty$ and the compensation method works well although there is a slight performance degradation as compared with the case without soft errors. Figure 6(d) shows the differences between the output errors when there is no soft error and when the soft error is compensated using compensation method 1.

Figure 7 shows $||\mathbf{e}^k||_\infty$ trajectory when there is no soft error and when the induced soft error is compensated using the compensation methods. For compensation methods 1 and 3, we can observe that the error approaches 0 as $k \to \infty$, whereas for compensation methods 2, $||y^d(j) - y^k(j-1)|| \leq ||\mathbf{e}^{k-1}||_\infty$ cannot always be satisfied, and the monotonic convergence cannot be guaranteed.

In this example, when using compensation method 3, as compared to compensation method 1, $\hat{e}^k(j)$ becomes 0, and it reduces the speed of convergence. As shown in Figure 7, compensation method 1 is the best method in this example, among compensation methods 1, 2, and 3, considering the monotonic convergence in the ILC. Further, depending on the location and characteristics of the soft errors, we can combine the presented four compensation methods for achieving a better performance.

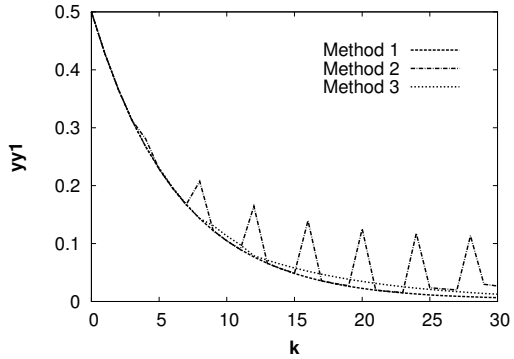


Figure 7: $||\mathbf{e}^k||_\infty$ according to k for the different compensation methods

6 Conclusion

In this paper, we have studied the effects of soft errors in the ILC and proposed novel compensation methods for the soft errors in the ILC. First, we have analyzed the effects of soft errors in the ILC. It has been shown that a soft error can affect the speed of the learning curve and make the controller lose the learning history. Second, based on the detection algorithm for the critical soft error that affects the monotonic convergence, we have proposed compensation methods for the ILC. The convergence of the proposed method has been proved under the assumptions for soft errors. From the experimental results, we have demonstrated the effectiveness of our proposed methods.

Furthermore, it is expected that several directions of researches could be developed. We could make more complicated assumptions such as multiple soft errors and extend the results to the various control applications. As an example in the ILC, depending on the location and characteristics of the soft errors, we could combine the four presented compensation methods for achieving better performance. These remain as future work.

Funding

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2018R1D1A1A09083894), and also supported by the National Research Foundation of Korea (NRF) funded by the Korean Government (MSIP) (NRF-2016R1A5A1012966).

Author contributions

The authors contributed equally to this work.

Conflict of interest

The authors declare no conflict of interest.

Bibliography

[1] Ahn, H.-S.; Chen, Y.; Moore, K. L. (2007). Iterative Learning Control: Brief Survey and Categorization, *IEEE Trans. Syst., Man and Cybern., Part C,* 37(6), 1099-1121, 2007.

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

[3] Azizimazreah, A., et al. (2018). Tolerating soft errors in deep learning accelerators with reliable on-chip memory designs, *2018 IEEE International Conference on Networking, Architecture and Storage (NAS)*, 1-10, 2018.

[4] Baumann, R. (2005). Soft errors in advanced computer systems, *Design & Test of Computers*, 22(3), 258-266, 2005.

[5] Bien, Z.; Xu, J.-X. (1998). *Iterative Learning Control Analysis, Design, Integration and Applications*, Kluwer Academic Publishers, 1998.

[6] IEEE Computer Society, *IEEE Standard for Floating-Point Arithmetic(IEEE Std 754)*, IEEE, 2008.

[7] Jeong, G.-M.; Choi, C.-H. (2002). Iterative learning control for linear discrete time nonminimum phase systems, *Automatica*, 38(2), 287–291, 2002.

[8] Jeong, G.-M., et al. (2016). Robust face recognition against soft-errors using a cross-layer approach, *International Journal of Computers Communications & Control*, 11(5), 657-665, 2016.

[9] Jeong, G.-M.; Ji, S.-H. (2017). Learning speed enhancement of iterative learning control with advanced output data based on parameter estimation, *International Journal of Computers Communications & Control*, 12(3), 323-329, 2017.

[10] Khudia,D. S.; Wright, G.; Mahlke, S. (2012). Efficient soft error protection for commodity embedded microprocessors using profile information, In *the 13th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, Tools and Theory for Embedded Systems (LCTES '12)*, 99–108, 2012.

[11] Lee, K., et al. (2006). Mitigating soft error failures for multimedia applications by selective data protection, *Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*, 411-420, 2006.

[12] Lee, K., et al. (2008). Mitigating the impact of hardware defects on multimedia applications: a cross-layer approach, *In: Proceedings of the 16th ACM international conference on Multimedia*, 319-328, 2008.

[13] Lee, K., et al. (2007). Partially protected caches to reduce failures due to soft errors in multimedia applications, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 17(9), 1343-1347, 2007.

[14] Lyons, D. (2000). *Sun screen: Soft error issue in sun enterprise servers*, Forbes, 2000.

[15] Michalak, S. E., et al.(2005). Predicting the number of fatal soft errors in los alamos national laboratory's ASC Q supercomputer, *IEEE Transactions on Device and Materials Reliability*, 5(3), 329-335, 2005.

[16] Moore, K. L.; Chen, Y.; Bahl, V. (2005). Monotonocally convergent iterative learning control for linear discret time systems, *Automatica*, 41(1), 1529–1537, 2005.

[17] Mukherjee, S. (2008). *Architecture Design for Soft Errors*, Morgan Kaufmann Publishers Inc., 2008.

[18] Nakka, N.; Pattabiraman, K.; Iyer, R. (2007). Processor-Level Selective Replication, *In: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 544-553, 2007.

[19] Tremblay, M.; Tami, Y. (1989). Support for fault tolerance in VLSI processors, *IEEE International Symposium on Circuits and Systems*, 388-392, 1989.

[20] Uchiyama, M. (1978). Formulation of high-speed motion pattern of mechanical arm by trial, *Trans. Soc. Instrum. and Contr. Eng.* (in Japanese), 14(6), 706–712, 1978.

[21] Vera, X., et al. (2009). Selective replication: A lightweight technique for soft errors, *ACM Transactions on Computer Systems*, 27(4), 1-30, 2009.

[22] Wang, N. J.; Quek, J.; Rafacz, T.; Patel, S. (2004). Characterizing the effects of transient faults on a high-performance processor pipeline, *2004 Intl. Conf. on Dependable Systems and Networks*, 61-70, 2004.