2025 IEEE 64th Conference on Decision and Control (CDC)
December 10-12, 2025. Rio de Janeiro, Brazil

Self-optimization of nonlinear iterative learning control and repetitive control*

Leontine Aarnoudse[1], Alexey Pavlov[1], and Tom Oomen[2]

2025 IEEE 64th Conference on Decision and Control (CDC) | 979-8-3315-2627-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/CDC57313.2025.11312565

***Abstract*— Nonlinear iterative learning control (ILC) and nonlinear repetitive control (RC) approaches introduce additional design freedom compared to linear time-invariant (LTI) approaches. Since the actual performance improvements depend on the parameters used in the nonlinearity, the aim of this paper is to optimize these parameters during the learning process. With optimal parameters, the nonlinear algorithms can outperform their LTI counterparts, for example by achieving fast attenuation of repeating disturbances without amplifying non-repeating disturbances. In this paper, we present the algorithm for the automatic learning/tuning process and validate it using simulations of an industrial flatbed printer.**

I. Introduction

Learning-based control approaches such as iterative learning control (ILC) and repetitive control (RC) can lead to high performance for systems with repeating disturbances by learning to attenuate these disturbances completely. The main difference between ILC and RC is that ILC considers separate experiments with system state resets, and learns a feedforward signal that is updated offline after each iteration [1]. RC considers periodic disturbances during continuous operation without resets, and a periodic signal generator is included in the closed loop to reject these disturbances [2].

Most ILC and RC approaches are linear and respectively iteration- or time-invariant, which leads to certain design trade-offs. An example where these methods are limited is the case where repeating and non-repeating disturbances occur simultaneously. The repeating disturbances can be compensated through learning, but attempting to learn compensating signals for the non-repeating disturbances leads to amplification of these disturbances [3]–[5]. In both ILC and RC, a common method to deal with this situation is the use of low-pass robustness filters. This does not only reduce the amplification of varying disturbances at high frequencies, but also the attenuation of repeating disturbances. Alternatively, a learning gain can be used to average over iterations or periods, leading to limited amplification of varying disturbances at the cost of a reduced convergence speed.

Many methods have been developed to combat the limitations of linear iteration- or time-invariant learning control. For example, in [6] an iteration-varying ILC strategy is developed that uses P-type or D-type ILC updates based on Kalman filtering, which requires accurate system and disturbances knowledge. In [7], similar knowledge is used in a Wiener-filtering approach to find an optimal ILC update. Other ILC methods are based on stochastic approximation, such as [8] which uses stochastic gradient descent with decreasing step sizes based on perturbation experiments, or [3] which systematically reduces the learning gain in a model-based ILC update. In the case of repetitive control, additional design freedom is often implemented through higher-order RC [9], [10] or the inclusion of observers [11].

Additional design freedom in ILC and RC can also be introduced through a static nonlinearity in the learning filter. This relates to the idea of variable-gain controllers in feedback systems, see, e.g., [12], [13]. Nonlinear frequency-domain ILC was first introduced in [14], [15]. In [16], [17], the idea is extended to lifted nonlinear ILC and nonlinear RC, enabling tuning for robustness and monotonic convergence using the notion of discrete-time nonlinear convergent systems [18]. Nonlinear ILC and RC achieve fast convergence and small errors in the presence of both varying and repeating disturbances, thus alleviating the trade-off mentioned earlier.

Nonlinear ILC and RC can lead to improved performance compared to linear iteration- or time-invariant approaches, but the performance depends strongly on the selection of certain design parameters. For example, the amplification of iteration-varying disturbances can be avoided by using a deadzone nonlinearity, but this requires selecting the correct deadzone width. In [16], some straightforward guidelines are provided for the design of the filters and the linear and nonlinear gains in ILC and RC. The selection of the deadzone width is more involved and requires knowledge of the disturbances. For the case of variable-gain feedback control an iterative optimization method is developed in [13], but similar design approaches for nonlinear ILC and RC are lacking. Other methods for the performance optimization of nonlinear systems or control schemes often consider the steady-state behavior, see, e.g., [19], and are not applicable to ILC and RC where transient behavior, such as the convergence speed, is also an important performance criterion.

Although significant steps have been taken to increase the design freedom in ILC and RC through nonlinearities, at present the parameter selection is ad hoc and not systematic. Therefore, the aim of this paper is to develop a method to optimize certain design parameters automatically over iterations. The contribution consists of the following elements:

- A new self-optimizing nonlinear ILC algorithm is de-

*This work was supported by the Research Council of Norway (RCN) through the project EXTREME EFFICIENCY: Data-driven optimization of industrial processes in time-varying environments (RCN project number 345272).

[1]The authors are with the Dept. of Geoscience, NTNU Norwegian University of Science and Technology, Trondheim, Norway. leontine.i.m.aarnoudse@ntnu.no

[2]Tom Oomen is with the Dept. of Mechanical Engineering, Control Systems Technology, Eindhoven University of Technology, Eindhoven, The Netherlands. He is also with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands.

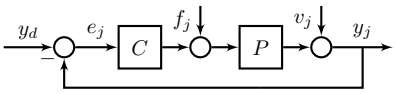

Fig. 1: Feedback control scheme with plant P, controller C, reference y_d, feedforward f_j and disturbance v_j.

veloped (Section III).

- The case of self-optimizing nonlinear RC, which is fundamentally different because of the lack of system resets, is considered (Section IV).
- The method is applied to the self-optimization of the deadzone width in ILC in the presence of both varying and repeating disturbances and validated through simulations (Section V).

Notation: For a vector x and a matrix P, $\|x\|_2 = \sqrt{\sum_{i=-\infty}^{\infty} |x_i|^2} < \infty$ denotes the ℓ_2-norm for $x \in \ell_2$. The set of real-rational, causal and stable transfer function is denoted by $\mathcal{RH}_\infty$, and $\mathcal{RL}_\infty$ denotes the set of rational transfer functions. The sets of real, natural, and integer numbers are denoted by $\mathbb{R}$, $\mathbb{N}$ and $\mathbb{Z}$, respectively.

II. Problem formulation

In this section, (non)linear ILC is introduced. Then, the importance of parameter tuning is illustrated using the example of a deadzone nonlinearity. Finally, the considered problem is formulated.

A. Linear and nonlinear iterative learning control

Consider a single-input, single-output linear time-invariant system with a plant P and a feedback controller C, as shown in Fig. 1. The error of the closed-loop system is given by

$$e_j = S(y_d - v_j) - Jf_j, \tag{1}$$

with sensitivity $S = (1 + PC)^{-1}$, process sensitivity $J = (1+PC)^{-1}P$, reference y_d, feedforward f_j and disturbance v_j for iteration $j \in \mathbb{N}$. It is assumed that y_d is iteration-invariant, while v_j may vary over iterations. ILC is used to iteratively update the feedforward, typically according to

$$f_{j+1} = Q(f_j + \alpha L e_j), \tag{2}$$

with learning filter $L \in \mathcal{RL}_\infty$ which approximates J^{-1}, zero-phase low pass robustness filter $Q \in \mathcal{RL}_\infty$, and learning gain $\alpha \in (0, 1]$. The ILC update is illustrated in Fig. 2, where $\varphi = 0$ corresponds to the linear case.

The ILC algorithm can be extended by a static nonlinearity to create additional design freedom and improve the performance. The static nonlinearity φ satisfies an incremental sector condition given by

$$0 \leq \tfrac{\varphi(a)-\varphi(b)}{a-b} \leq \gamma, \; \forall\, a, b \in \mathbb{R}. \tag{3}$$

The nonlinear ILC update is of the form

$$f_{j+1} = Q(f_j + \alpha L e_j + L\varphi(e_j)), \tag{4}$$

as illustrated in Fig. 2. Regarding the convergence of this nonlinear ILC scheme, the following result from [16] holds.

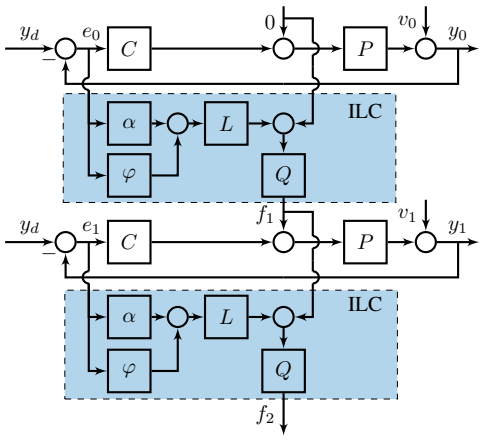

Fig. 2: Nonlinear ILC scheme with learning filter L, robustness filter Q, learning gain α and nonlinearity φ.

Lemma 1. *The sequence of errors $\{e_j\}$ according to* (1) *with feedforward update* (4)*, nonlinearity φ satisfying* (3) *and $\gamma, \alpha > 0$ converges monotonically in the ℓ_2-norm to a unique steady-state solution if*

$$\left\| Q\left(1 - \left(\alpha + \frac{\gamma}{2}\right) JL\right) \right\|_{\mathcal{L}_\infty} + \frac{\gamma}{2} \|QJL\|_{\mathcal{L}_\infty} < 1. \tag{5}$$

Nonlinear ILC algorithms of this form have been developed for frequency-domain ILC and lifted ILC, and similar ideas have also been applied to nonlinear repetitive control, see [16], [17]. These earlier results show that convergence can be ensured under mild conditions and a well-chosen nonlinearity can increase the performance significantly.

B. Parameter tuning in nonlinear ILC

The performance of nonlinear ILC depends on certain parameters. This is illustrated using the example of a deadzone nonlinearity, which can alleviate the trade-off between fast convergence and low converged errors in ILC as shown in [16], [17]. If a system contains both iteration-varying and iteration-invariant disturbances, often a small learning gain α is used to create an averaging effect over iterations. This avoids amplification of iteration-varying disturbances, but it also leads to slow convergence. Through nonlinear ILC, different learning gains can be applied to iteration-varying and iteration-invariant disturbances based on their amplitude characteristics. The deadzone nonlinearity is given by

$$\varphi(e_j(k)) = \begin{cases} 0, & \text{if } |e_j(k)| \leq \delta \\ \left(\gamma - \frac{\gamma\delta}{|e_j(k)|}\right) e_j(k), & \text{if } |e_j(k)| > \delta. \end{cases} \tag{6}$$

Here $\delta \in \mathbb{R}$ denotes the width of the deadzone, and $\gamma \in \mathbb{R}$ is a nonlinear learning gain, as illustrated in Fig. 3. The main idea of this nonlinear algorithm is that learning is only applied to error values larger than the threshold value δ, where δ is chosen such that $\varphi(Sv_j) \approx 0$. The parameters Q, L, α and γ all influence the stability of the nonlinear learning algorithm, as shown in Lemma 1, and their selection is relatively straightforward: Q and L can be chosen as they would be in linear ILC, α should be small, and γ should be

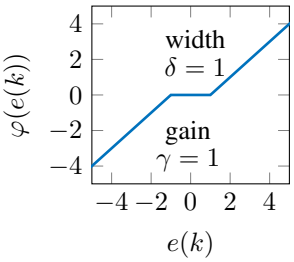

Fig. 3: Deadzone nonlinearity with width δ and gain γ.

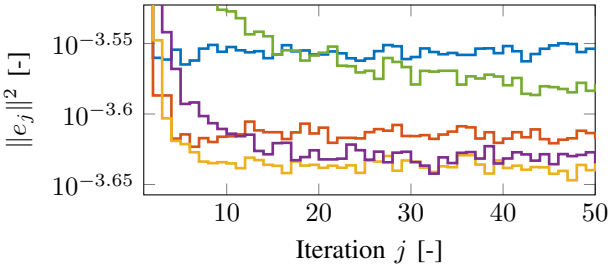

Fig. 4: Influence of the deadzone width (averaged over 20 realizations). $\delta = 1 \times 10^{-6}$ (—), 3×10^{-6} (—), 5×10^{-6} (—), 7×10^{-6} (—) and 1×10^{-5} (—).

close to 1 while still meeting the convergence criterion, see also the more detailed analysis in [16].

The main design freedom in nonlinear ILC lies in the selection of the deadzone width δ, which does not influence the stability. In general, $\varphi(Sv_j)$ should be approximately zero, but neither S nor v_j is known exactly. Fig. 4 shows the influence of different deadzone widths on the convergence speed and converged error 2-norm for the simulation example from Section V. The figure shows that there is a optimal deadzone width, and that other values for the width lead to increased error values or slow convergence.

C. Problem formulation

ILC can be extended by static nonlinearities, which increase the design freedom. This implies that obtaining good performance still requires the suitable selection of certain parameters. The aim of this paper is to develop an automatic optimization method that enables tuning of the relevant parameters while the nonlinear ILC algorithm is running.

III. Self-optimizing nonlinear ILC

In this section, a method for the optimization of certain parameters in the static nonlinearity in nonlinear ILC is developed. First, the generic case is considered and second, the approach is applied to optimizing the deadzone width.

A. Optimizing the nonlinearity in nonlinear ILC

Consider the nonlinear ILC update (4). The aim is to optimize a parameter p_φ of the nonlinearity $\varphi(e_j, p_\varphi)$. It is assumed that the parameter that is optimized does not influence the stability of the ILC algorithm, such as for example the width of a deadzone nonlinearity. If the parameter to be optimized can render the system unstable, constrained optimization methods need to be considered instead.

Consider the criterion

$$\mathcal{J}(p_\varphi) = \hat{e}_{j+1}^{\mathsf{T}}(p_\varphi)\hat{e}_{j+1}(p_\varphi), \tag{7}$$

where $\hat{e}_{j+1}(p_\varphi)$ denotes the estimated error at the next iteration, given by

$$\hat{e}_{j+1}(p_\varphi) = \hat{S}y_d - \hat{J}f_{j+1}(p_\varphi). \tag{8}$$

Here the unknown sensitivity and process sensitivity functions have been replaced with respectively the estimates $\hat{S}$ and $\hat{J}$, and $f_{j+1}(p_\varphi)$ is given by (4) with nonlinearity $\varphi(e_j, p_\varphi)$. Note that the estimates $\hat{S}$ and $\hat{J}$ are typically available in ILC, since they are also used to design the learning filter L. To minimize the criterion $\mathcal{J}(p_\varphi)$, at each iteration the parameter p_φ is updated according to

$$p_{\varphi,j+1} = p_{\varphi,j} - \beta\frac{\partial\mathcal{J}(p_\varphi)}{\partial p_\varphi}, \tag{9}$$

i.e., gradient descent with step size $\beta > 0$. The gradient $\frac{\partial\mathcal{J}(p_\varphi)}{\partial p_\varphi}$ of $\mathcal{J}$ with respect to p_φ is given by

$$\begin{aligned}
\frac{\partial\mathcal{J}(p_\varphi)}{\partial p_\varphi} &= \frac{\partial\hat{e}_{j+1}^{\mathsf{T}}\hat{e}_{j+1}}{\partial p_\varphi} = \\
&= \frac{\partial}{\partial p_\varphi}\left(y_d^{\mathsf{T}}\hat{S}^{\mathsf{T}}\hat{S}y_d - 2f_{j+1}^{\mathsf{T}}\hat{J}^{\mathsf{T}}\hat{S}y_d + f_{j+1}^{\mathsf{T}}\hat{J}^{\mathsf{T}}\hat{J}f_{j+1}\right) \\
&= \left(\frac{\partial}{\partial f_{j+1}}\left(-2f_{j+1}^{\mathsf{T}}J^{\mathsf{T}}\hat{S}y_d + f_{j+1}^{\mathsf{T}}\hat{J}^{\mathsf{T}}\hat{J}f_{j+1}\right)\right)^{\mathsf{T}}\frac{\partial f_{j+1}}{\partial p_\varphi} \\
&= \left(-2\hat{J}^{\mathsf{T}}\hat{S}y_d + 2\hat{J}^{\mathsf{T}}f_{j+1}\right)^{\mathsf{T}}\frac{\partial f_{j+1}}{\partial p_\varphi} \\
&= \left(-2\hat{J}^{\mathsf{T}}\hat{e}_{j+1}\right)^{\mathsf{T}}\frac{\partial f_{j+1}}{\partial p_\varphi}.
\end{aligned} \tag{10}$$

The form of the derivative $\frac{\partial f_{j+1}}{\partial p_\varphi}$, with f_{j+1} according to (4), depends on the specific nonlinearity and parameter. Equations (7)-(10) provide a general update law for tuning any parameter in the nonlinearity φ that does not affect the stability of the ILC algorithm. In the next section, we will derive an exact formula for (10) for the case of tuning the parameter δ, i.e., the width of the deadzone nonlinearity (6).

B. Self-optimization of the deadzone width

To compute the derivative $\frac{\partial f_{j+1}}{\partial\delta}$ of the next feedforward signal f_{j+1} with respect to the deadzone width, the deadzone nonlinearity in (6) is rewritten as

$$\varphi(e_j(k),\delta) = \varphi_1(e_j(k),\delta)e_j(k) + \delta\varphi_2(e_j(k),\delta), \tag{11}$$

$$\varphi_1(e_j(k),\delta) = \begin{cases} 0, & \text{if } |e_j(k)| \le \delta \\ \alpha, & \text{if } |e_j(k)| > \delta, \end{cases} \tag{12}$$

$$\varphi_2(e_j(k),\delta) = \begin{cases} 0, & \text{if } |e_j(k)| \le \delta \\ -\alpha\text{sign}(e_j(k)), & \text{if } |e_j(k)| > \delta. \end{cases} \tag{13}$$

Then, the following theorem holds.

Theorem 2. *Consider the feedforward update* (4) *with deadzone nonlinearity* φ *according to* (11)*. The derivative with respect to the deadzone width* δ *is given by*

$$\frac{\partial f_{j+1}}{\partial\delta} = QL\varphi_2(e_j,\delta). \tag{14}$$

Consequently, the gradient of criterion (7) *with respect to the deadzone width is given by*

$$\frac{\partial \mathcal{J}(\delta)}{\partial \delta} = \left(-2\hat{J}^{\mathsf{T}}\hat{e}_{j+1}(\delta)\right)^{\mathsf{T}} QL\varphi_2(e_j,\delta). \tag{15}$$

Proof. The derivative $\frac{\partial f_{j+1}}{\partial \delta}$ is given by

$$\begin{aligned}\frac{\partial f_{j+1}}{\partial \delta} &= \frac{\partial}{\partial \delta}\left(Q(f_j + \alpha L e_j + L\varphi_1(e_j,\delta)e_j + \delta L\varphi_2(e_j,\delta))\right)\\ &= \frac{\partial}{\partial \delta} QL\left(\varphi_1(e_j,\delta)e_j + \delta\varphi_2(e_j,\delta)\right)\\ &= QL\left(\frac{\partial \varphi_1(e_j,\delta)}{\partial \delta}e_j + \frac{\partial \varphi_2(e_j,\delta)}{\partial \delta}\delta + \varphi_2(e_j,\delta)\right). \end{aligned}\tag{16}$$

The partial derivatives $\frac{\partial \varphi_1(e_j,\delta)}{\partial \delta}$ and $\frac{\partial \varphi_2(e_j,\delta)}{\partial \delta}$ result from step functions and are given by Dirac delta functions, which are non-zero valued at $e_j = \delta_j$ and $e_j = -\delta_j$. Next, it is shown how these partial derivatives cancel each other, see also [13]. The nonlinearities φ_1 and φ_2 in (12) and (13) are equivalent to (indices j and k are omitted for brevity)

$$\begin{aligned}\varphi_1(e,\delta) = \lim_{z\to 0}\Big\{\gamma - \Big(\tfrac{\gamma}{\pi}\arctan\Big(\tfrac{2\pi(e+\delta)}{z}\Big) - \\ \Big(\tfrac{\gamma}{\pi}\arctan\Big(\tfrac{2\pi(e-\delta)}{z}\Big)\Big)\Big)\Big\},\end{aligned}\tag{17}$$

$$\begin{aligned}\varphi_2(e,\delta) = \lim_{z\to 0}\Big\{-\Big(\tfrac{\gamma}{\pi}\arctan\Big(\tfrac{2\pi(e+\delta)}{z}\Big) - \\ \Big(\tfrac{\gamma}{\pi}\arctan\Big(\tfrac{2\pi(e-\delta)}{z}\Big)\Big)\Big)\Big\},\end{aligned}\tag{18}$$

with deadzone width $\delta > 0$ since $\delta = 0$ implies linear ILC. The partial derivatives of φ_1 and φ_2 to e are given by

$$\frac{\partial \varphi_1(e,\delta)}{\partial e} = \lim_{z\to 0}\left\{\frac{-2\gamma z}{z^2+4\pi^2(e+\delta)^2} + \frac{2\gamma z}{z^2+4\pi^2(e-\delta)^2}\right\}, \tag{19}$$

$$\frac{\partial \varphi_2(e,\delta)}{\partial e} = \lim_{z\to 0}\left\{\frac{-2\gamma z}{z^2+4\pi^2(e+\delta)^2} - \frac{2\gamma z}{z^2+4\pi^2(e-\delta)^2}\right\}. \tag{20}$$

These partial derivatives are equal to zero for $e \neq \pm\delta$. For $e = \delta$, it holds that

$$\begin{aligned}&\frac{\partial \varphi_1(e,\delta)}{\partial e}e + \frac{\partial \varphi_2(e,\delta)}{\partial e}\delta = \\ &= \lim_{z\to 0}\left\{-\frac{2\gamma\delta z}{z^2+16\pi^2\delta^2} + \frac{2\gamma\delta z}{z^2} - \frac{2\gamma\delta z}{z^2+16\pi^2\delta^2} - \frac{2\gamma\delta z}{z^2}\right\}\\ &= \lim_{z\to 0}\left\{-\frac{4\gamma\delta z}{z^2+16\pi^2\delta^2}\right\} = 0.\end{aligned}\tag{21}$$

For $e = -\delta$ a similar result can be obtained. Finally, we obtain that

$$\begin{aligned}&\frac{\partial \varphi_1(e_j,\delta)}{\partial \delta}e_j + \frac{\partial \varphi_2(e_j,\delta)}{\partial \delta}\delta = \\ &= \left(\frac{\partial \varphi_1(e_j,\delta)}{\partial e_j}e_j + \frac{\partial \varphi_2(e_j,\delta)}{\partial e_j}\delta\right)\frac{\partial e_j}{\partial \delta} = 0.\end{aligned}\tag{22}$$

Substituting (22) in the derivative (16) shows that

$$\frac{\partial f_{j+1}}{\partial \delta} = QL\varphi_2(e_j,\delta). \tag{23}$$

Combining (10) and (23) gives the full derivative. □

Using Theorem 2 and (9), the update of nonlinearity parameter δ becomes

$$\delta_{j+1} = \delta_j + \beta\left(2\hat{J}^{\mathsf{T}}\hat{e}_{j+1}(\delta)\right)^{\mathsf{T}} QL\varphi_2(e_j,\delta) \tag{24}$$

The error estimate $\hat{e}_{j+1}$ in (15) is computed according to (8) using feedforward estimate $\hat{f}_{j+1}(\delta_j)$, which uses the previous deadzone value δ_j, and the available models $\hat{S}$ and $\hat{J}$. In Section V, the efficiency of this algorithm is demonstrated in simulations of an industrial flatbed printer.

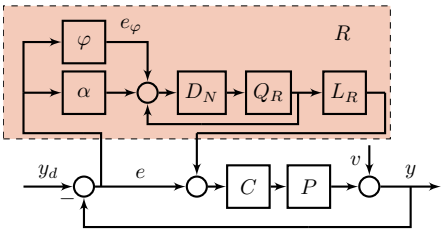

Fig. 5: Nonlinear repetitive control scheme.

IV. Extension to nonlinear RC

In this section, the idea of self-optimizing nonlinear repetitive control is considered. First, (non)linear RC is introduced and second, the optimization of the parameters is discussed.

A. Linear and nonlinear repetitive control

RC is based on the internal model principle [20] and learns a model of the periodic disturbances that is included in the feedback controller. Consider the closed-loop system of Fig. 1 with an initial error according to (1) with $f_j = 0$. RC is typically implemented as shown in Fig. 5, where $\varphi = 0$ corresponds to the linear case. The linear repetitive controller is given by

$$R(z) = \frac{\alpha L_R(z) z^{-N} Q_R(z)}{1 - z^{-N} Q_R(z)}, \tag{25}$$

with z^{-N} the z-domain representation of the delay operator D_N, i.e., $D_N(z) = z^{-N}$. The buffer length N is chosen to correspond to the disturbance frequency. Similar to ILC, $\alpha \in (0,1]$ is a learning gain, and $L_R \in \mathcal{R}$ and $Q_R \in \mathcal{R}$ are respectively learning and robustness filters. In RC, these filters can have finite preview while the repetitive controller $R \in \mathcal{RH}_\infty$ is still causal, by embedding their preview in z^{-N}. Typically, L is chosen to approximate T^{-1}, with $T = PC(1+PC)^{-1}$, for example using ZPTEC [21]. Robustness filter Q is typically chosen as a finite impulse response low-pass filter. The error with RC is given by

$$e = (1 + PC(1+R))^{-1}(y_d - v) = S_R e_0, \tag{26}$$

with modifying sensitivity $S_R = (1+TR)^{-1}$.

A nonlinearity φ can be included in the repetitive controller as shown in Fig. 5. The input e of the nonlinearity is written as a function of its output signal $e_\varphi = \varphi(e)$ and the disturbances contained in e_0, see (1), as follows.

$$e = T_R e_\varphi + S_R e_0, \tag{27}$$

with S_R the modifying sensitivity and $T_R = (1+TR)^{-1}TR$ the complementary sensitivity of the linear RC loop, see, e.g., [22] for a derivation. The system can be rewritten in state-space form as a cascade of a linear system S_R and a Lur'e system consisting of the linear system $-T_R$ with the static

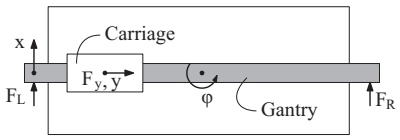

Fig. 6: Schematic illustration of a flatbed printer. In this paper, a model of the y-translation of the carriage is used.

nonlinearity φ in feedback:

$$\left.\begin{aligned} x(k+1) &= Ax(k)+Bu(k) \\ y(k) &= Cx(k) \end{aligned}\right\} -T_R \tag{28a}$$

$$u(k) = -\varphi(y(k)+w(k))$$

$$\left.\begin{aligned} n(k+1) &= An(k)+Be_0(k) \\ w(k) &= Cn(k)+e_0(k) \end{aligned}\right\} S_R \tag{28b}$$

This notation leads to the following convergence result [16].

Lemma 3. *Given a minimal realization (A,B,C) of $-T_R$ with (A,B) controllable and (A,C) observable. The nonlinear RC system* (27) *is exponentially convergent, as defined in [18, Definition 1], for any input $e_0(k)$ bounded on $\mathbb{Z}$ if $\rho(A)<1$, φ satisfies* (3) *for some γ, and the small-gain condition $\sup_{\omega\in[0,2\pi)}|T_R(e^{i\omega})|<\frac{1}{\gamma}$ holds.*

B. Self-optimizing nonlinear RC

Similar to nonlinear ILC, the performance of nonlinear RC depends strongly on the selection of certain parameters. The main difference between nonlinear ILC and RC when considering the optimization of the parameters of the nonlinearity is that ILC is inherently an iterative method, whereas RC runs continuously. The iterative gradient descent approach developed in Section III can therefore not be applied directly while RC is running. Instead, an initialization process is considered, which is suitable for applications where RC will be used continuously for a long time, or where it is implemented on multiple machines.

Repetitive control is implemented as a feedback controller and can be represented by a serial connection of state space systems (28). Therefore, the method developed in [13] can be applied, which relies on iterative optimization over separate experiments. For RC, one experiment should consists of multiple periods of the repetitive signal, such that both the convergence speed and the converged error are taken into account. The iterative scheme is then aimed at minimizing

$$\mathcal{J}_R(p_\varphi) = \hat{e}_{j+1}^{\mathsf{T}}(p_\varphi)\hat{e}_{j+1}(p_\varphi), \tag{29}$$

with $\hat{e}_{j+1}$ the estimated error of the next multiple-period experiment, based on a combination of system models and the measured error signal e_j of the previous experiment.

V. Printer Example

In this section, nonlinear frequency-domain ILC is applied to a model of an industrial flatbed printer. The deadzone width is updated iteratively according to (24), and the results are compared to different constant deadzone widths.

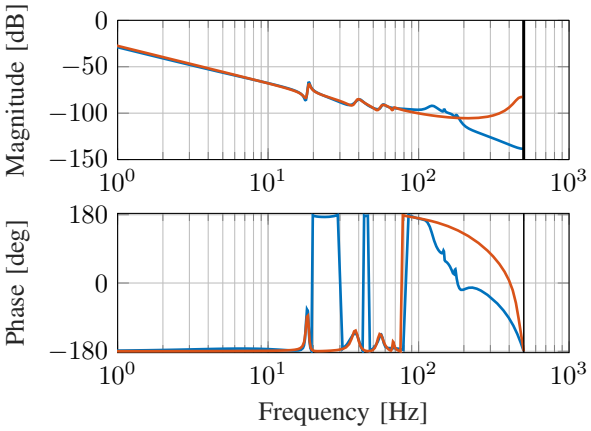

Fig. 7: Bode diagram of the 'true' simulated system P (—) and a low-order approximation $\hat{P}$ (—).

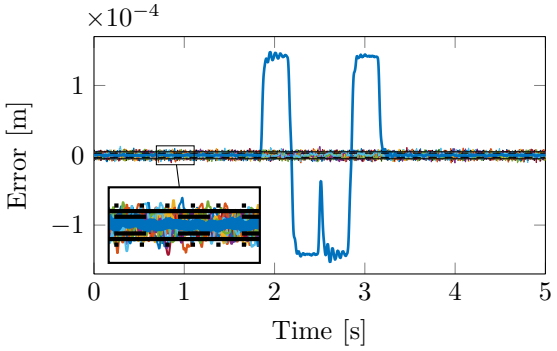

Fig. 8: Mean of the error signal over 20 iterations (—) and the noise estimates $\widehat{Sv}_j$ with three possible deadzone widths: 3×10^{-6} (- -), 5×10^{-6} (—), and 8×10^{-6} (·····).

A. Setup

Consider the translation of the carriage of a flatbed printer, illustrated in Fig. 6 and represented by a 20th-order model. In addition to this high-order model P, a 12th-order model $\hat{P}$ is also available, with a model mismatch at high frequencies, see Fig. 7. The controller C is a PD-type controller, and the fourth-order reference consists of a forward and backward translation. Gaussian white noise with a variance of 0.005 is added to the plant input. The error signal is illustrated in Fig. 8, which shows both the averaged error over 20 realizations, and 20 estimates of the noise realization $\widehat{Sv}_j$.

To apply frequency-domain ILC, the learning filter L is designed using stable inversion, see, e.g., [23], of the process sensitivity $\hat{J}$. This approximation is based on the low-order model $\hat{P}$. Since there is a model mismatch at high frequencies, the robustness filter Q is chosen as a first-order, zero-phase low pass filter with a cutoff frequency of 100 Hz. The gains are chosen as $\alpha=0.01$ and $\gamma=1$. All results are averaged over 20 realizations.

B. Results

In Fig. 4 in Section II, it is shown that a deadzone that is smaller than the optimal width of approximately 5×10^{-6} retains fast convergence but leads to a high converged error, whereas a deadzone that is wider than 5×10^{-6} leads to a reduced convergence speed. The results obtained by the self-

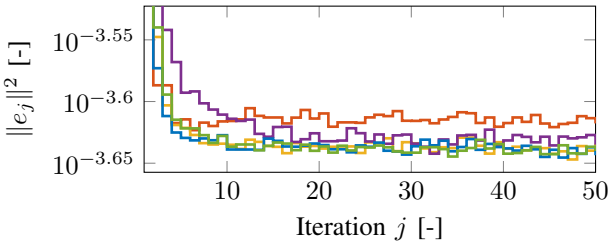

Fig. 9: A self-optimizing deadzone width performs as good as the optimal one, even when the initial value is far from optimal. Results are averaged over 20 realizations for constant $\delta = 3 \times 10^{-6}$ (—), 5×10^{-6} (—), 7×10^{-6} (—), and optimization with $\delta_1 = 1 \times 10^{-6}$ (—) and 1×10^{-5} (—).

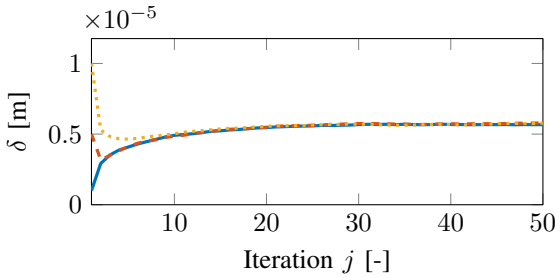

Fig. 10: Deadzone width δ over iterations for initial values of $\delta = 1 \times 10^{-6}$ (—), $\delta = 5 \times 10^{-6}$ (—) and $\delta = 1 \times 10^{-5}$ (—), averaged over 20 realizations.

optimizing nonlinear ILC algorithm with step size $\beta = 2 \times 10^{-4}$ are shown in Fig. 9. Two different values for the initial deadzone width δ_1 are used: 1×10^{-6} and 1×10^{-5}. Even though these initialization values are far from the optimum, self-optimizing nonlinear ILC performs as good as nonlinear ILC with a constant $\delta = 5 \times 10^{-6}$, recovering both the low converged error and the high convergence speed.

In Fig. 10, the deadzone width over iterations is shown for $\delta_1 = 1 \times 10^{-6}$, 5×10^{-6} and 1×10^{-5}. For all initialization points, the width converges quickly to the same value of around 5×10^{-6}. The figure also shows that for $\delta_1 = 5 \times 10^{-6}$, δ initially reduces before increasing. A possible explanation is that in earlier iterations, the iteration-invariant part of the error is still large, so that a smaller deadzone width can be beneficial in earlier iterations.

VI. Conclusion

The optimal parameters for the nonlinearity in nonlinear ILC and RC can be found automatically using a gradient descent algorithm that combines measured data with system models. As an example, a case of ILC with repeating and non-repeating disturbances is considered, where a deadzone nonlinearity can lead to both fast convergence and low converged errors. The optimal deadzone width is found within a small number of iterations, without requiring a priori knowledge of the disturbances. Simulations of an industrial flatbed printer demonstrate that the self-optimizing algorithm performs as well as nonlinear ILC with an a priori known optimal deadzone width in terms of convergence speed and converged error. Future research is aimed at further exploring possible applications of nonlinear ILC and RC.

Acknowledgment

The authors wish to thank Marcel Heertjes for early conversations on this topic.

References

[1] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Syst.*, vol. 26, no. 3, pp. 96–114, 2006.

[2] R. W. Longman, "On the theory and design of linear repetitive control systems," *Eur. J. Control*, vol. 16, no. 5, pp. 447–496, 2010.

[3] M. Butcher, A. Karimi, and R. Longchamp, "A statistical analysis of certain iterative learning control algorithms," *Int. J. Control*, vol. 81, no. 1, pp. 156–166, jan 2008.

[4] T. Oomen and C. R. Rojas, "Sparse iterative learning control with application to a wafer stage: Achieving performance, resource efficiency, and task flexibility," *Mechatronics*, vol. 47, pp. 134–147, 2017.

[5] X. Chen and M. Tomizuka, "New repetitive control with improved steady-state performance and accelerated transient," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 2, pp. 664–675, mar 2014.

[6] S. S. Saab, "Stochastic P-type/D-type iterative learning control algorithms," *Int. J. Control*, vol. 76, no. 2, pp. 139–148, jan 2003.

[7] A. Deutschmann-Olek, G. Stadler, and A. Kugi, "Stochastic Iterative Learning Control for Lumped-and Distributed-Parameter Systems: A Wiener-Filtering Approach," *IEEE Trans. Automat. Contr.*, vol. 66, no. 8, pp. 3856–3862, aug 2021.

[8] H. Chen and C. Hanfu, "Almost sure convergence of iterative learning control for stochastic systems," *Sci. China (Series F)*, vol. 46, no. 1, pp. 67–79, 2003.

[9] M. Steinbuch, S. Weiland, and T. Singh, "Design of noise and period-time robust high-order repetitive control, with application to optical storage," *Automatica*, vol. 43, no. 12, pp. 2086–2095, dec 2007.

[10] G. Pipeleers, B. Demeulenaere, J. De Schutter, and J. Swevers, "Robust high-order repetitive control: Optimal performance trade-offs," *Automatica*, vol. 44, no. 10, pp. 2628–2634, oct 2008.

[11] D. De Roover, O. H. Bosgra, and M. Steinbuch, "Internal-model-based design of repetitive and iterative learning controllers for linear multivariable systems," *Int. J. Control*, vol. 73, no. 10, pp. 914–929, jul 2000.

[12] M. Heertjes and M. Steinbuch, "Stability and performance of a variable gain controller with application to a dvd storage drive," *Automatica*, vol. 40, no. 4, pp. 591–602, apr 2004.

[13] M. F. Heertjes and H. Nijmeijer, "Self-tuning of a switching controller for scanning motion systems," *Mechatronics*, vol. 22, no. 3, pp. 310–319, 2012.

[14] M. Heertjes and T. Tso, "Nonlinear iterative learning control with applications to lithographic machinery," *Control Eng. Pract.*, vol. 15, no. 12, pp. 1545–1555, dec 2007.

[15] M. Heertjes, R. Rampadarath, and R. Waiboer, "Nonlinear Q-filter in the learning of nano-positioning motion systems," in *2009 Eur. Control Conf.* Budapest, Hungary: IEEE, aug 2009, pp. 1523–1528.

[16] L. Aarnoudse, A. Pavlov, and T. Oomen, "A design framework for nonlinear iterative learning control and repetitive control: Applied to three mechatronic case studies," *Control Eng. Pract.*, vol. 149, no. 15698, p. 105976, 2024.

[17] ——, "Nonlinear iterative learning control for discriminating between disturbances," *Automatica*, vol. 171, p. 111902, 2025.

[18] A. Pavlov and N. Van De Wouw, "Steady-state analysis and regulation of discrete-time nonlinear systems," *IEEE Trans. Automat. Contr.*, vol. 57, no. 7, pp. 1793–1798, 2012.

[19] A. Pavlov, B. G. Hunnekens, N. V. Wouw, and H. Nijmeijer, "Steady-state performance optimization for nonlinear control systems of Lur'e type," *Automatica*, vol. 49, no. 7, pp. 2087–2097, jul 2013.

[20] B. Francis and W. Wonham, "The Internal Model Principle of Linear Control Theory," *Automatica*, vol. 12, pp. 457–465, 1976.

[21] M. Tomizuka, "Zero phase error tracking algorithm for digital control," *J. Dyn. Syst. Meas. Control*, vol. 109, pp. 65–68, 1987.

[22] W. S. Chang, I. H. Suh, and T. W. Kim, "Analysis and design of two types of digital repetitive control systems," *Automatica*, vol. 31, no. 5, pp. 741–746, may 1995.

[23] T. Sogo, "On the equivalence between stable inversion for nonminimum phase systems and reciprocal transfer functions defined by the two-sided Laplace transform," *Automatica*, vol. 46, no. 1, pp. 122–126, jan 2010.