Iterative learning control in high-performance motion systems: from theory to implementation

Martin Goubej*, Sven Meeusen†, Noud Mooren† and Tom Oomen†
*NTIS Research Centre, University of West Bohemia, Pilsen, Czech Republic
†Department of Mechanical Engineering, Technische Universiteit Eindhoven, Eindhoven, Netherlands
mgoubej@ntis.zcu.cz,s.meeusen@student.tue.nl,n.f.m.mooren@tue.nl,T.A.E.Oomen@tue.nl

***Abstract*—Iterative learning control (ILC) enables a perfect compensation for systems that perform the same task over and over again. The aim of this paper is to demonstrate practical applicability of two various state-of-the-art ILC algorithms to point-to-point positioning systems. A simple Frequency domain ILC approach is exploited focusing on systems with exactly repeating motion tasks. Furthermore, flexible ILC is employed to enable learning also for non-repeating tasks. Particular steps providing a seamless transfer from theory and algorithms to practical implementation in a real-time environment by means of industrial-grade SW and HW are given. They may serve as a practical example of a workflow suitable for a wide range of motion control applications. Potential benefits of the learning-type control in comparison with conventional feedback and feedforward control are discussed as well.**

***Index Terms*—iterative learning control, motion control, frequency-domain ILC, basis-function ILC, advanced feedforward control, real-time systems**

I. Introduction

Model-based feedforward and learning control are essential to achieve high-performance for motion systems. Manually tuned feedforward control is able to compensate for known exogenous disturbances such as the reference signal. Furthermore, many positioning systems perform repeating motion tasks where learning algorithms such as Iterative Learning Control (ILC) can be used to improve performance automatically. ILC enables updating the feedforward signal by learning from previous motion tasks, and thereby compensating for the repeating parts of the error [1]. This method is successfully applied to many high-precision motion systems, see, e.g., wafer scanners [2], wire-bonding equipment [3] and printing systems [4].

Control theory for ILC is well-developed [1], [5]–[7], however, it is not applicable to motion systems that perform varying motion tasks. Standard ILC algorithms require re-learning of the feedforward signals if the reference changes which is not desired. Recent developments have extended standard ILC algorithms towards learning for varying references [8]–[10]. In this approach, the feedforward signal is parameterized as function of the reference and basis function. This allows to extrapolate the feedforward towards varying references, enabling learning feedforward for a wide range of motion systems performing also varying tasks.

Although important steps have been made in tailoring ILC to motion applications, its implementation in industrial control hardware instead of rapid prototyping equipment requires additional attention. Most of the state-of-the-art methods are not well known by practicing control engineers or they are considered too theoretical, potentially dangerous and inapplicable when using standard industrial-grade HW equipment and corresponding SW development tools. The goal is to show that the ILC design framework offers some strong theoretical fundamentals allowing to guarantee stability and convergence and it can be applied relatively simply using existing software environments and standard components. A case study of a generic, possibly varying, point-to-point positioning task is introduced to demonstrate practical applicability of the proposed methods and to provide some guidelines for real-time implementation. Performance benefits are also studied in comparison with conventional feedback and feedforward control strategies.

The paper is organized as follows. Section II introduces some theoretical background of two specific ILC methods which we consider especially useful for generic motion systems due to their inherent properties and ease of use. Section III discusses the necessary ingredients required for their successful implementation in real-time control environment. Section IV presents the flexible manipulator problem which was chosen as a representative use-case. It describes the workflow used in the ILC design phase and presents one of the possibilities of real-time implementation. Achievable closed-loop performance is compared to the conventional feedback or feedback plus model-based feedforward control structures. Concluding remarks and suggestions follow in the last section.

II. Iterative learning control theory

In this section, the standard control framework is introduced. Furthermore, frequency domain ILC (FD-ILC) and flexible or basis-functions ILC (BF-ILC) are introduced and notions of convergence are provided.

A. ILC Setup

Consider the system depicted in Fig. 1 consisting of an LTI plant P and stabilizing feedback controller C_{fb}. All the signals included in the figure have subscript j which refers to a repetition of the signal r, also denoted as a task.

978-1-7281-0303-7/19/$31.00 ©2019 IEEE

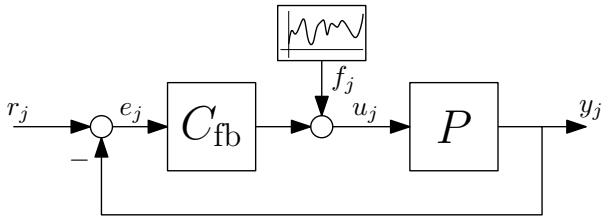

Fig. 1. Control Setup for Frequency Domain ILC.

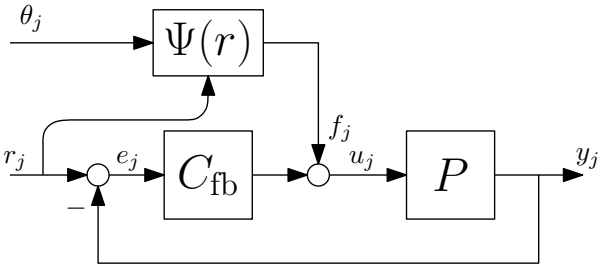

Fig. 2. Control Setup for Flexible ILC.

B. Frequency domain ILC

Frequency-domain ILC is a convenient method from a design point of view as shown in this section, however, it may be restrictive since it is assumed that $r_j = r_{j+1}$, i.e., the tasks are exactly repeating. The goal of the ILC algorithm is to compute the feedforward signal f_j, such that the error converges in the task domain, i.e., $e_\infty = \lim_{j\to\infty} e_j = 0$, where the error at task j is given by

$$e_j = Sr_j - PSf_j, \tag{1}$$

and $S = (I + PC_{fb})^{-1}$.

Typically, the following update is used to compute the feedforward signal for the next task,

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

where e_j is the error in the previous task, f_j is the previous feedforward signal and Q and L are to be designed filters. Note that L and Q can be non-causal since the update depends on signals from the previous task, i.e., $L, Q \in \mathcal{RL}_\infty$.

To gain insight in the design of Q and L consider the closed-loop error propagation by substituting (1) in (2) resulting in,

$$e_{j+1} = Q(1 - PSL)e_j + (1 - Q)Sr. \tag{3}$$

Furthermore, it can be shown that the error converges monotonically w.r.t. its 2-norm if the following condition is satisfied

$$|Q(e^{j\omega})(1 - P(e^{j\omega})S(e^{j\omega})L(e^{j\omega}))| < 1, \forall\omega \in [0, 2\pi] \tag{4}$$

for more details see, e.g., [1], [11].

Design of the FD-ILC follows along the following three steps as outlined in [12], [13];

(1) Obtain an approximate model of PS denoted with $\hat{PS}$
The aim is to find an approximate parametric model of PS that is required in the subsequent steps. This can be a parametric fit obtained from frequency domain measurement data.

(2) Design the learning filter $L \approx \hat{PS}^{-1}$.
To obtain the optimal performance, i.e., have $e_\infty \to 0$ one must satisfy that

$$L = (PS)^{-1} \quad \text{and} \quad Q = 1 \tag{5}$$

such that the error (3) becomes zero. However, because of model errors and inversion errors obtaining an exact inverse is not possible in practice, hence the aim is to approximate $(PS)^{-1}$, i.e., design $L \approx (\hat{PS})^{-1}$. For non-minimum phase system, additional care must be taken, see [14].

(3) Design the robustness filter Q.
In this final step, the robustness filter Q is designed to compensate for model error and assure that the convergence condition (4) is satisfied for all frequencies. Usually a low-pass type characteristics is needed to accomplish that. In this condition, the filter $L(e^{j\omega})$ is designed using the parametric model, and the true frequency response data of the plant will be used to check convergence. Hence, if Q deviates from 1 then essentially learning is cut-off for that frequency to ensure convergence of the algorithm at the expense of larger error after convergence.

C. Flexible ILC

The previously described FD-ILC algorithm is relatively easy to design and allows to check for monotonic convergence based of FRF data. The main disadvantage of this method is that the reference must be the same each task, otherwise performance may even deteriorate and the learning has to start over again. In this section, a brief description of BF-ILC will be provided which enables learning if $r_j \neq r_{j+1}$ without having to re-learn, see as observed in the experimental results in Fig. 11.

Consider the control setup used for flexible ILC as depicted in Fig. 2, where the feedforward signal is parameterized as function of the reference through Ψ in combination with feedforward parameters θ_j. Instead of optimizing the signal f_j, the basis functions Ψ are fixed and the parameters θ_j are optimized. The feedforward signal becomes

$$f_j = \Psi(r)\theta_j \tag{6}$$

where θ_j are the parameters in task j, and

$$\Psi = \begin{bmatrix}\psi_1 & \psi_2 & \dots & \psi_m\end{bmatrix} \in \mathbb{R}^{N\times m} \tag{7}$$

contains the m basis functions and N is the length of the task. To optimize the feedforward parameters the following cost function is minimized

$$\mathcal{J}(\theta_{j+1}) = \|e_{j+1}\|^2_{W_e} + \|f_{j+1}\|^2_{W_f} + \|f_{j+1} - f_j\|^2_{W_{\Delta f}}. \tag{8}$$

A weight can be set on the error signal, feedforward signal and the learning rate by the weighting matrices W_e, W_f and $W_{\Delta f}$ respectively. Next, the update of the feedforward parameters is computed by solving $\frac{\partial\mathcal{J}(\theta_{j+1})}{\partial\theta_{j+1}} = 0$ to which the solution is given by

$$\theta_{j+1} = Q\theta_j + Le_j, \tag{9}$$

where

$$L = \Gamma \Psi^\top J^\top W_e, \tag{10}$$

$$Q = \Gamma \Psi^\top (J^\top W_e J + W_{\Delta f}) \Psi, \tag{11}$$

and $\Gamma = \left(\Psi^\top (J^\top W_e J + W_f + W_{\Delta f}) \Psi\right)^{-1}$, and $J \in \mathbb{R}^{N \times N}$ is the convolution matrix of PS. Note that this method is also often referred to as optimal ILC since the matrices Q and L follow from an optimization problem rather than manual tuning. Furthermore, it is worth mentioning that in this case the matrices Q and L can be time-varying which is not the case in the frequency domain implementation where Q and L are LTI filters.

III. Real-time implementation aspects

The ILC framework is built upon some strong theoretical results giving essential statements about stability and performance of the whole learning process. This is an important prerequisite for application in industry which calls for guaranteed solutions and repeatable behavior due to safety reasons. Although quite theoretically complex concepts may be required to study ILC design methods and their properties, e.g. by means of operator theory, functional analysis, 2D systems, anti-causal filtering, optimal control etc., the final algorithm suitable for practical implementation may be relatively simple and straightforward to use.

The key problem from the implementation point of view is to correctly define the behavior of the learning process and the way of its interaction with the standard feedback loop. This can be achieved by using three basic components available in most of the software environments for control system design:

- **Finite state machine**
  The ILC algorithm uses multiple realizations of a repetitive task to recursively improve the learned feedforward signal which is injected into the feedback loop. A state machine which switches between the ”learning” and ”trial” modes of operation and ensures their proper synchronization is needed. Most of the SW tools intended for industrial control contain a support for the definition of finite-state automata. The usual form is the Sequential function chart (SFC), a graphical programming language based on Petri nets, which is one of the five programming languages defined by the IEC 61131-3 standard for PLC programming. It is missing in some simple platforms but it can be substituted by basic logic circuits, memory elements and flip-flops available in more low-level languages. An example of the control logic used for the real-time implementation is shown in the simplified diagram in Fig. (7).
- **1D array**
  The ILC methods work with the stored samples of a performance variable (usually the tracking error) which must be stored in memory for further processing. A simple 1D buffer (array) may serve for this purpose. When dealing with memory problems, the learning step may be realized in other device than the control HW itself, e.g. in a supervisory PLC or a dedicated computer which communicates the signals necessary for the feedforward adaptation in both directions.
- **Functional block/subroutine with for-cycles definition**
  This part takes care of the sequential processing of the sampled data. It implements the learning and robustness filters introduced in Section II which are used for automatic adaptation of the feedforward signal in a recursive manner. A simple for-cycle and basic algebraic operations are sufficient for this purpose.

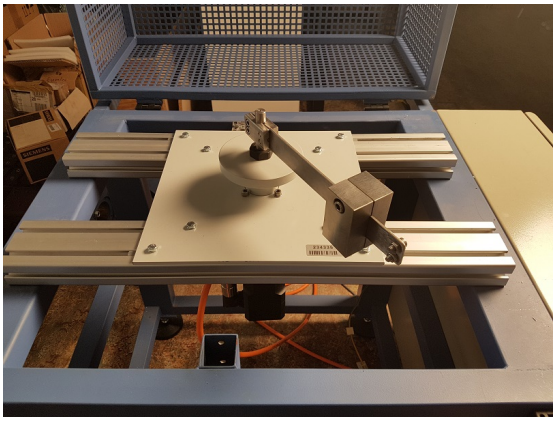

Fig. 3. Flexible arm testbed - mechanical setup used for the experiments

The REXYGEN software tool was chosen for implementing the ILC algorithms in the provided use-case study. It serves as an example of an industry-oriented platform for rapid control system development [15] providing the tooling for graphical programming of control algorithms with a minimum of hand coding. Most of the application software can be realized by means of provided standard functional blocks library. The functional blocks are connected via oriented information links in the same manner as in the well-known Matlab/Simulink environment or Functional block diagrams language in the PLC world. Specific functionalities may be implemented in a C-like scripting language by means of a user-programmable REXLANG block. The next section provides the guidelines for the design and implementation phases. They are directly transferable to other SW platforms supporting the above mentioned elementary components.

IV. Case study: optimization of repeating manipulation task

A mechanical testbed representing a generic motion axis was chosen for the purpose of demonstration of the proposed algorithms and design methodology (Fig. 3). The mechanical part consists of a flexible coupling, bearing housing, flywheel inertia and removable flexible arm with adjustable load mass. There is one degree of freedom allowing the arm to rotate around the vertical axis. The setup is driven by an electrical drive (500W permanent magnets synchronous servomotor) controlled by a TGDrives frequency inverter realizing the current control loop by means of standard field oriented

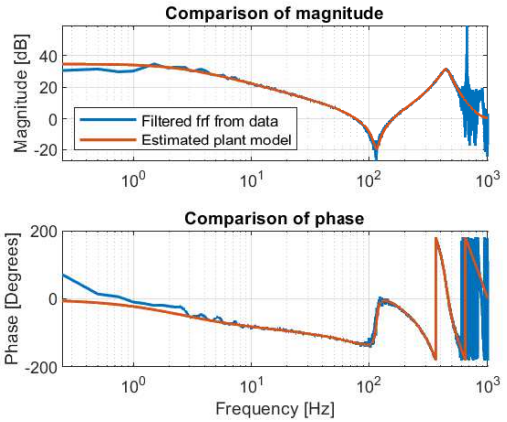

Fig. 4. Identified plant and estimated model

control algorithm. A supervisory control system implements the position control loop, the ILC algorithms and auxiliary logic control of the whole testbed. Standard B&R industrial PC running real-time Linux and REXYGEN control software was used [15].

The testbed can exhibit diverse dynamic characteristics based on the particular configuration of its mechanical elements. In our particular case, a single resonance dynamics was chosen and the problem of point-to-point positioning aiming at highest achievable accuracy of reference trajectory (9) tracking was chosen for the evaluation of the developed ILC strategies.

A. Model-based control design

A model of the controlled plant is obtained by conducting experiments where a sequence of pseudo-random signals with a fixed amplitude is used as input to the system. This excitation input displays similar characteristics as a white noise sequence, implicating that the generated signals are a series of uncorrelated successive samples. Nonparametric frequency response function (FRF) model is derived using the Welch's method of windowed and averaged periodograms. Using this data, a model fit is made, which can be seen in Fig. 4. The position controller is chosen as a 2-DoF PID controller with the control law given in the standard ISA form as:

$$U(s) = K\left[bW(s) - Y(s) + \frac{1}{T_i s}E(s) + \frac{T_d s}{\frac{T_d}{N}s + 1}(cW(s) - Y(s))\right], \tag{12}$$

where U, W, Y, E denote the Laplace images of controller output, setpoint reference, measured output and tracking error and K, T_i, T_d, N, b, c are the controller parameters which were tuned using loop-shaping techniques.

B. Model-based feedforward control

Conventional feedforward control can be applied in order to reduce the tracking error, thus increasing the error tracking performance of the system. Ideally, the feedforward signal should resemble the setpoint reference filtered through the inverse of the plant dynamics $f = P^{-1}r$, as this will lead to perfect tracking. However, due to e.g. model errors, inexact or unstable inverse this cannot be achieved in practice. The error norm is reduced by a factor of ten as shown in Fig. 10 when using an approximate stable inverse of the plant model for filtering the reference signal.

C. Advanced feedforward using ILC

The tracking performance can be improved further by applying the mentioned learning feedforward methods.

1) FD-ILC: For the FD-ILC method it is chosen to implement ILC in serial realization which is beneficial when dealing with integrator windup problem due to the sensor noise (see e.g. [16]). To apply FD-ILC to the system, the learning filter L should be constructed as the inverse of the complementary sensitivity $L = T^{-1}$ instead of the plant sensitivity PS. This inversion can be derived by using the Zero Phase Error Tracking Control algorithm (ZPETC) [17] which ensures that the resulting filter is stable and a desirable zero-phase property of the product $\angle TL = 0\ \forall\omega$ is achieved. The learning filter can be split into a causal part L_c and a finite preview term:

$$L = z^{p+d}L_c, \tag{13}$$

where d corresponds to the pure delay in T and p corresponds to the number of its unstable zeros. In Figure 5, T, L_c and L are presented in a bode diagram, which shows the phase effects of the non-causality of L. To realize the non-causal filtering of the error signal, the signal is first filtered with the causal filter, and afterwards shifted in time to achieve non-causality. This is realized by performing the time shift as in:

$$x_{nc} = [x_c(p+d+1), ..., x(N), 0_{p+d}] \tag{14}$$

where x_c is the error filtered by the causal filter L_c. By applying the time shift, x_{nc} now becomes the error signal filtered through the non-causal filter L. To achieve a similar signal length, the signal is padded with zeros equal to the amount of preview samples.

The robustness filter Q can be implemented as a discrete FIR system. This ensures that a simple discrete convolution operation can be used in order to filter the signal. Non-causal filtering can be achieved since the whole sequence of relevant signals f_j, e_j in (2) is known in advance in each learning update. This allows to exploit a desirable zero-phase property of the Q filter, see e.g. [18]. For the implementation of the robustness filter in this work, a lowpass filter is used to filter out unwanted high frequencies and a bandstop to filter out unwanted bending mode dynamics due to the flexible coupling at the motor. This filter is represented in Figure 6.

The previously described steps can be performed in real-time relatively easy with a few function blocks to implement the ILC algorithm. A finite state machine as visualized in Figure 7 regulates the calculations in the ILC domain, and applies the calculated feedforward in the trial domain. Figure 8 presents three function blocks that are used to perform the ILC algorithm. The upper left function block is a state space representation of the causal filter L_c from (13). The

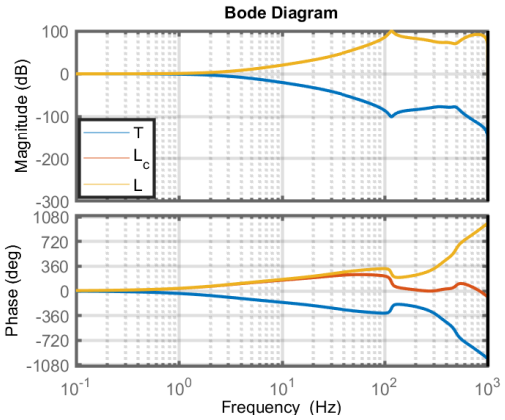

Fig. 5. Causal and non-causal learning filter from ZPETC

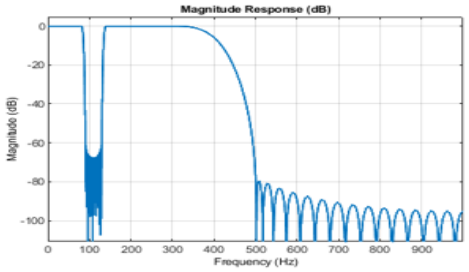

Fig. 6. Robustness filter Q

ATMT function block acts as a finite state machine that regulates the switch between the learning and trial domain. The REXLANG function block acts as a data buffer, and performs the time shift to achieve non-causal filtering, and the filtering through Q. Subsequently, after these calculations are performed, the learned feedforward is stored and injected to the feedback loop (PIDU block in Fig. 8) during the motion task.

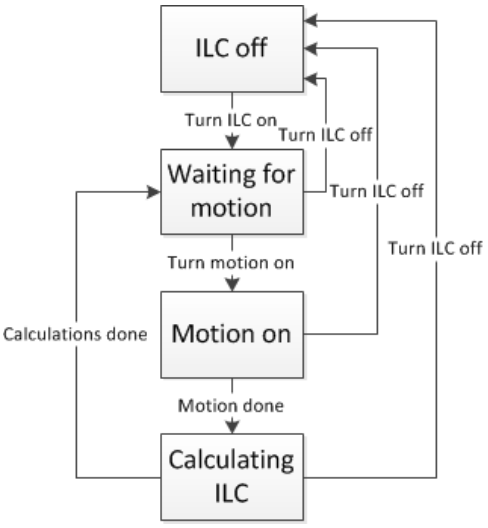

Fig. 7. Visualization of the finite state machine

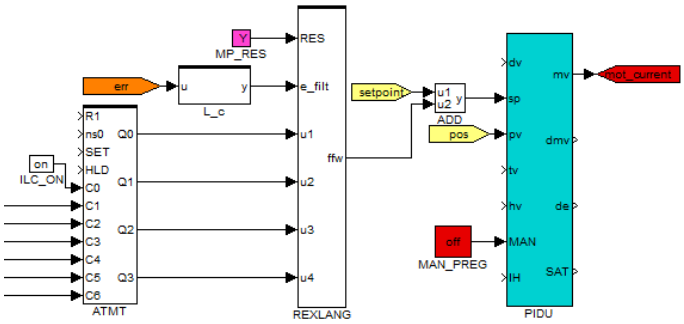

Fig. 8. Implementation of the FD-ILC algorithm in REXYGEN system

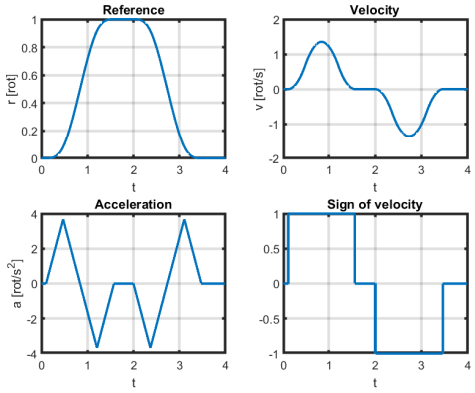

Fig. 9. Basis Φ for BF-ILC

2) BF-ILC: In order to design the basis-functions ILC an appropriate basis Ψ has to be chosen. In this work, Ψ is chosen to contain the following functions of the reference r (see e.g. [19] for the detailed discussion):

$$\Psi = [r \quad \dot{r} \quad \ddot{r} \quad sign(\dot{r})] \tag{15}$$

Figure 9 presents a visualization of the basis Ψ. The second, third and fourth term in this basis compensate for the viscous friction, lack of force to accelerate and the coulomb friction respectively. The first term compensates for the reference, which is needed due to the usage of the 2-DoF PID controller. This basis Ψ is used in order to construct L and Q as in (10) and (11). The learning update is performed trough matrix calculations (9), where the Q and L matrices are loaded into the memory before the first iteration. The feedforward action is generated subsequently by means of chosen basis functions as $f_j = \Psi\theta_j$. In real-time, these steps are performed similarly as previously described for the FD-ILC.

D. Comparison of the proposed control strategies

Figure 10 compares the two different ILC control methods to conventional feedforward and conventional feedback control. It can be seen that both FD-ILC and BF-ILC improve the performance in terms of minimizing the norm of the tracking error. The FD-ILC outperforms the BF-ILC in case of exactly repeating task. This is mainly due to the limited class of feedforward action which can be generated as a linear combination of the chosen basis-functions (6). Perfect compensation is unattainable if the basis does not match the disturbance

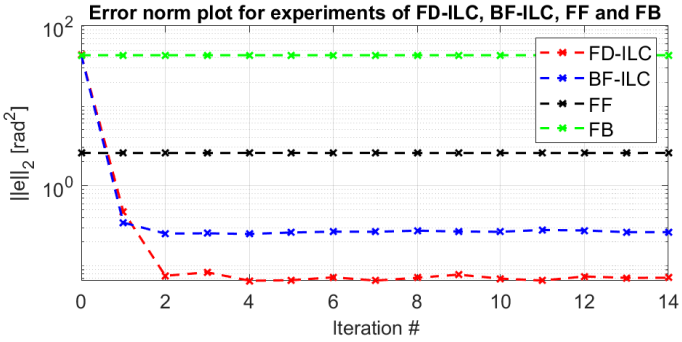

Fig. 10. Comparison of the 2-norm of the error for experiments with BF-ILC, FD-ILC, feedforward and only feedback.

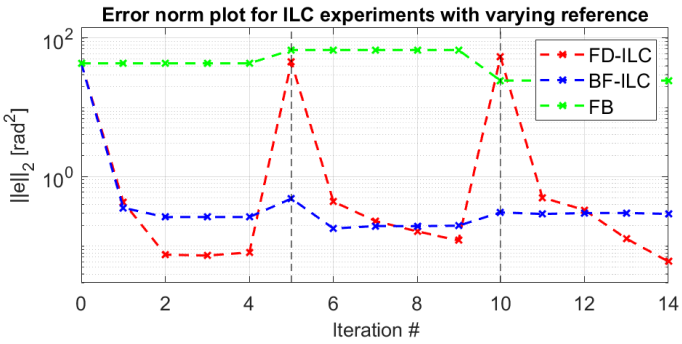

Fig. 11. Comparison of the 2-norm of the error for experiments with BF-ILC, FD-ILC and only feedback for varying references. The reference changes at the 5th and 10th iteration.

and plant characteristics exactly. FD-ILC is not subject to this problem. Figure 11 presents the results of the experiments with iterations that are subject to varying references. The point-to-point trajectory was modified twice by changing the final position and maximum velocity/acceleration limits (see trials Nr. 5 and 10 in Fig. 11). The norm of the error for the FD-ILC increases significantly, it may even exceed the value achieved without the feedforward compensation. The BF-ILC shows only a slight increase of the error norm at the iterations where the reference is changed. The results indicate that FD-ILC is preferable for constant tasks. If robustness against varying tasks is required, the BF-ILC method achieves much better results at the cost of worse steady state performance. Concluding from these results, the employment of both ILC methods shows a significant improvement in terms of the tracking error.

V. Conclusions

The paper deals with two iterative learning control methods which are employed in a generic motion control scenario aiming at optimization of the achieved tracking performance. It is shown that the ILC algorithms can be implemented quite easily with the use of few basic building blocks which are readily available in existing industrial HW and SW platforms. Experimental case study demonstrates achievable performance improvement compared to the conventional feedback and model-based feedforward control. The scope of the ILC framework is much broader than indicated here and involves other important aspects such as multi-variable systems, parameter varying dynamics, optimal design or more complex basis function parameterizations. Some of these issues are addressed in the provided list of references.

Acknowledgment

This work was supported from the H2020 ECSEL JU project I-MECH under grant agreement Nr. 737453 [20] and from ERDF under project InteCom No. CZ.02.1.01/0.0/0.0/17_048/0007267.

References

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

[2] 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.

[3] F. Boeren, A. Bareja, T. Kok, and T. Oomen, "Frequency-domain ILC approach for repeating and varying tasks: With application to semiconductor bonding equipment," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 6, pp. 2716–2727, Dec 2016.

[4] J. Bolder, T. Oomen, and M. Steinbuch, "On inferential iterative learning control: with example to a printing system," in *2014 American Control Conference*. IEEE, 2014, pp. 1827–1832.

[5] H.-S. Ahn, K. L. Moore, and Y. Chen, *Iterative learning control: robustness and monotonic convergence for interval systems*. Springer Science & Business Media, 2007.

[6] Y. Wang, F. Gao, and F. J. Doyle III, "Survey on iterative learning control, repetitive control, and run-to-run control," *Journal of Process Control*, vol. 19, no. 10, pp. 1589–1600, 2009.

[7] G. Pipeleers and K. L. Moore, "Unified analysis of iterative learning and repetitive controllers in trial domain," *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 953–965, 2014.

[8] D. Hoelzle, A. Alleyne, and A. W. Johnson, "Basis task approach to iterative learning control with applications to micro-robotic deposition," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 5, pp. 1138–1148, 2011.

[9] J. van de Wijdeven and O. H. Bosgra, "Using basis functions in iterative learning control: analysis and design theory," *International Journal of Control*, vol. 83, no. 4, pp. 661–675, 2010.

[10] J. Bolder, T. Oomen, S. Koekebakker, and M. Steinbuch, "Using iterative learning control with basis functions to compensate medium deformation in a wide-format inkjet printer," *Mechatronics*, vol. 24, no. 8, pp. 944–953, 2014.

[11] D. De Roover and O. H. Bosgra, "Synthesis of robust multivariable iterative learning controllers with application to a wafer stage motion system," *international Journal of Control*, vol. 73, no. 10, pp. 968–979, 2000.

[12] N. Strijbosch, L. Blanken, T. Oomen *et al.*, "Frequency domain design of iterative learning control and repetitive control for complex motion systems," in *IEEJ International Workshop on Sensing, Actuation, Motion Control, and Optimization*, 2018, pp. 1–2.

[13] T. Oomen, "Learning in machines," *Mikroniek*, 2018 nr 6.

[14] J. van Zundert and T. Oomen, "On inversion-based approaches for feedforward and ilc," *Mechatronics*, vol. 50, pp. 282–291, 2018.

[15] REX Controls s.r.o., "REXYGEN SW tools," *www.rexygen.com*, 2019.

[16] T. Bolder and T. Oomen, "Inferential iterative learning control: A 2d-system approach," *Automatica 71*, 2016.

[17] M. Tomizuka, "Zero phase error tracking algorithm for digital control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 109, no. 1, pp. 65–68, 1987.

[18] D. Elliott, *Handbook of Digital Signal Processing: Engineering Applications*. Academic Press, 1988.

[19] P. Lambcrechts, M. Boerlage, and M. Steinbuch, "Trajectory planning and feedforward design for electromechanical motion systems," *Control Engineering Practice 13*, 2005.

[20] M. Čech, K. Ozols, and A.-J. Beltman, "I-MECH Smart system integration for mechatronic applications," *IEEE ETFA, In Press*, 2019.