Iterative Learning Control for Performance Optimisation

Bing Chu

Abstract— **Iterative learning control (ILC) is a popular design methodology to achieve high performance trajectory tracking of systems operating in a repetitive manner. This paper further extends the applicability of ILC by showing that ILC can solve a more general problem of optimising some system performance for which trajectory tracking is just a special case. The problem is formulated in a general Hilbert space setting using system operators and a gradient based algorithm is proposed as a solution. The algorithm's convergence and robustness properties are analysed in detail and numerical simulations are provided to demonstrate the effectiveness of the proposed method.**

I. Introduction

Iterative learning control (ILC) is a design methodology to achieve high performance tracking trajectory of systems working in a repetitive manner, and to do so, by learning during the information collected from the previous executions of the task (named trials). The idea is motivated by human learning: by observing how well you perform a task and properly adjusting your actions according to your previous performance, you will be able to improve your ability to perform the same task next time. Originating from robotics, ILC has now attracted extensive research interest and has been proven to be extremely successful in achieving high system performance across a wide range of applications from industrial robotic manipulators, chemical batch processes, medical equipment to rehabilitation [1].

In classic ILC design problem, the reference signal $r(t)$ is defined at every time instant t over the whole trial interval $[0,T]$. This however is not always the case in practical applications. As an example, when an robot arm performs a 'pick and place' task, we will be mainly interested in the output at the pick up and place down positions; what happens between these positions is of less concern. Motivated by these applications, the idea of point-to-point ILC is developed where the design objective is to find an control input such that the output tracks a given reference trajectory defined not on the whole trial interval, but only at a finite number of time instants. A number of the point-to-point ILC design algorithms have been proposed in the literature, see [2]–[9] for more details.

In point-to-point ILC, there are more flexibilities in choosing the control input. Theoretically, there will be infinite number of control inputs that can achieve perfect tracking at the required intermediate time points. In principle, these extra degrees of freedom can be further exploited to improve some other system performance. Along this line, an auxiliary optimization problem is studied in [10], [11] where the system is required to track a given point-to-point reference and at the same time to minimise some quadratic auxiliary system performance. Two switching algorithms are proposed in [10] and an inverse-model based approach is developed in [11]. More recently, a similar problem is studied in [12]. A multi-objective learning framework is developed by including a number of (possibly competing) performance metrics, e.g. point-to-point tracking performance, control energy etc., into a single optimisation based controller design. These results suggest that in addition to trajectory tracking, ILC also has the potential to improve some system performance of interest.

Inspired by the above work, this paper shows that the idea of ILC can be applied more generally to solve a performance optimisation problem, for which the classic trajectory tracking is just a special case, thus further extending the applicability of ILC. The idea can be exploited in principle in both classic ILC setting and point-to-point ILC problem but for simplicity this paper focuses on the former. The main contributions of the paper are summarised as follows:

- mathematical formulation of the idea of using ILC for performance optimisation into an abstract optimisation framework in Hilbert spaces (Section II). Formulating the problem in general Hilbert spaces has the clear advantage that it can include many situations of interest, e.g. linear time invariant or time varying state space models, linear differential models etc., providing a general platform for algorithm analysis and design.
- derivation of a gradient based method to solve the above problem and a detailed analysis of its convergence properties (Section III). It is shown the proposed algorithm has the appealing property that it will converge monotonically in the performance index to the optimal solution of the performance optimisation problem, i.e. the best possible performance that can be achieved. Implementation of the proposed algorithm is also discussed.
- rigorous robustness analysis of the proposed algorithm against model uncertainties (Section IV). The analysis results reveal that the proposed algorithm is capable of converging to the best performance even if the model used is different from the real plant. This feature is of great interest in practice as an accurate model is generally difficult to obtain. The model uncertainties that can be tolerated by the proposed algorithm is also characterised analytically.
- validation of the proposed method using a model of an

Bing Chu is with School of Electronics and Computer Science, University of Southampton, Highfield, Southampton SO17 1BJ, UK `b.chu@ecs.soton.ac.uk`

electro-mechanical rotatory non-minimum phase system (Section VI). The results clearly demonstrate the effectiveness and advantages of the proposed algorithm.

II. Problem Formulation

In this section, the idea of using ILC for performance optimisation is formulated using an abstract operator form representation in Hilbert spaces. For simplicity, a formulation is presented for discrete time, linear time-invariant systems but the abstract problem setting applies to more general linear systems including many situations of interest such as (but not limited to) discrete time, linear time-varying systems, continuous time linear time-invariant and time-varying state space models, linear differential models, and differential delay models etc.

Consider the following single-input, single-output discrete time, linear invariant system

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t) \\ y_k(t) &= Cx_k(t) \end{aligned} \tag{1}$$

where t is the time index (i.e. sample number), k is the trial number and $u_k(t)$, $x_k(t)$, $y_k(t)$ are respectively the input, state and output of the system on trial k. The system is working in a repetitive manner to perform the same task defined over a finite duration $t \in [0, N]$ and the system state is reset to the identical initial condition $x_k(0) = x_0$, $k = 1, 2, \cdots$ between trials, i.e. when $t = N+1$ the time is reset to $t = 0$ and state to x_0.

Without loss of generality assume the relative degree of the system is unity (i.e. the generic condition $CB \neq 0$ is satisfied, other cases can be handled in a similar way), then system model (1) on the k^{th} trial can be expressed in an equivalent 'lifted-system' form representation [13]

$$y_k = Gu_k + d, \tag{2}$$

where the $N \times 1$ vectors of input and output time series u_k, y_k are defined as

$$\begin{aligned} u_k &= \left[\begin{array}{cccc} u_k(0), & u_k(1), & \cdots, & u_k(N-1) \end{array}\right]^T \\ y_k &= \left[\begin{array}{cccc} y_k(1), & y_k(2), & \cdots, & y_k(N) \end{array}\right]^T, \end{aligned} \tag{3}$$

G is the linear system operator mapping from the input space $\mathcal{U} = \mathbb{R}^N$ with inner product defined as

$$\langle u, v \rangle_R = u^T R v$$

and associated induced norm

$$\|u\|_R = \sqrt{\langle u, u \rangle_R} = \sqrt{u^T R u}$$

to the output space $\mathcal{Y} = \mathbb{R}^N$ with inner product

$$\langle x, y \rangle_Q = x^T Q y$$

and associated induced norm

$$\|y\|_Q = \sqrt{\langle y, y \rangle_Q} = \sqrt{y^T Q y}$$

(where Q, R are positive definite matrices of compatible dimensions). In particular, in matrix from G can be represented as

$$G = \begin{bmatrix} CB & 0 & \cdots & 0 & 0 \\ CAB & CB & \ddots & 0 & 0 \\ CA^2B & CAB & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & CB & 0 \\ CA^{N-1}B & \dots & \dots & CAB & CB \end{bmatrix} \tag{4}$$

and the $N \times 1$ vector d represents the effect of initial conditions

$$d = \left[\begin{array}{cccc} CAx_0, & CA^2x_0, & \cdots, & CA^Nx_0 \end{array}\right]^T. \tag{5}$$

Note that matrix G is nonsingular as $CB \neq 0$. The design problem can now be stated as follows.

ILC for Performance Optimisation: The ILC algorithm design problem can now be stated as finding a control updating law

$$u_{k+1} = f\left(y_{k+1}, \cdots, y_{k-s}, u_k, \cdots, u_{k-r}\right) \tag{6}$$

where $s, r > 0$ are integers, such that the system input and output has the asymptotic property that a performance index $J(u, y)$ defined as

$$J(u, y) = g(u, y) \tag{7}$$

in which $g(u, y)$ is a continuous function of the input u and output y, is minimised, i.e. when $k \to \infty$

$$\begin{aligned} (u_k, y_k) &\to (u^*, y^*) \\ &= \arg\min_{(u,y)} \{J(u, y) = g(u, y) : y = Gu + d\} \end{aligned} \tag{8}$$

The above definition describes a large class of problems including many situations of interest. As a special case, the classic trajectory tracking problem can be included into the above framework by choosing the performance index $J(u, y)$ to be minimised as

$$J(u, y) = \|r - y\|_Q^2, \tag{9}$$

i.e. by choosing $g(u, y) = \|r - y\|_Q^2$ in (7), where r is the given reference signal. Minimising the performance $J(u, y)$ in this case requires that the tracking error $r - y$ is as small as possible and ideally zero (if the reference is within the range of the system operator, i.e. there exists an input signal to generate the given reference, which is true for the system operator G defined in (4)).

In this paper the following quadratic performance index is considered

$$J(u, y) = \|y - y_d\|_Q^2 + \rho\|u - u_d\|_R^2 \tag{10}$$

where y_d and u_d represent some desired operating points (note that they do not need to satisfy the system dynamics $y_d = Gu_d + d$;); $\rho \geq 0$ is a weighting scalar. Note that (10) has been used extensively in the classic optimal control theory to characterise system performance. The design objective is to find a control updating law (6) such that the above performance index is minimised as $k \to \infty$ by the designed input u_k and the system output y_k.

Note that when accurate model information is available, a solution of the above performance optimisation problem can be obtained directly (either analytically or numerically), in which case there is no benefit of using ILC to learn from the previous system performance. For example for the performance index (10) considered in this paper, this is just a finite horizon linear quadratic optimal control problem which admits a combined feedforward and feedback solution using Riccati equations [14]. However, when there is no exact model information available for use, the problem becomes much difficult as we are trying to minimise a performance index $J(u, y)$ in which u and y are constrained by some *unknown* system dynamics G_p that is different from the system model G we have, i.e. we are looking for the solution of the following optimisation problem

$$(u^*, y^*) = \arg \min_{(u,y)} \big\{ J(u, y) = \|y - y_d\|_Q^2 + \rho \|u - u_d\|_R^2 : y = G_p u + d \big\} \quad (11)$$

in which G_p is unknown. A possible solution is to solve the above problem using the (inaccurate) system model G and apply the obtained input to the system. However, the achieved system performance can be very different from the optimal one. In what follows, we will show that even under this difficult situation, we can still achieve the optimal system performance by using the idea from ILC.

III. A Gradient Based ILC Algorithm for Performance Optimisation

In this section, a gradient based ILC algorithm is proposed to solve the above performance optimisation problem. Its implementation procedures and convergence properties are discussed in detail.

A. Description of the Algorithm

The proposed gradient ILC algorithm is given as follows:

Step 1 Set iteration number $k = 0$, choose an initial input u_0, implement the input to the system and record the output y_0

Step 2 Update the input for the next iteration using a gradient based algorithm

$$u_{k+1} = u_k - \beta[G^*(y_k - y_d) + \rho(u_k - u_d)] \quad (12)$$

where $\beta > 0$ is a learning gain satisfying

$$0 < \beta < 2/(\rho + \|G\|^2),$$

G^* is the Hilbert adjoint operator mapping from $\mathcal{Y}$ to $\mathcal{U}$ with a matrix form representation

$$G^* = R^{-1} G^T Q \quad (13)$$

and $\|G\|$ denotes the norm of the operator G; implement the input u_{k+1} and record the system output y_{k+1}

Step 3 Set $k \leftarrow k + 1$, goto Step 2.

Note that the key step in updating the input u_{k+1} is to compute

$$G^*(y_k - y_d) = R^{-1} G^T Q(y_k - y_d). \quad (14)$$

This can be implemented by either off-line computation using the system model G, or on-line experiment as follows.

Introduce a time reversal matrix F as follows

$$F = \begin{bmatrix} 0 & \cdots & \cdots & 0 & 1 \\ 0 & \cdots & \cdots & 1 & 0 \\ 0 & \cdots & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & \cdots & 0 \end{bmatrix} \quad (15)$$

The time reversal matrix has the property that when operating on a signal, e.g. u_k, the resulting signal is just the time reversal of u_k, i.e.

$$F u_k = \begin{bmatrix} u_k(N-1), & u_k(N-2), & \cdots, & u_k(0) \end{bmatrix}^T.$$

It is clear that $F = F^T, F^2 = I$. Using the time reversal matrix F the matrix G^T can be written as

$$G^T = FGF$$

and thus (14) is equivalent o

$$G^*(y_k - y_d) = R^{-1} FGFQ(y_k - y_d) \quad (16)$$

Therefore to update u_{k+1} :

(i) compute $Q(y_k - y_d)$ and $\rho(u_k - u_d)$
(ii) calculate $FQ(y_k - y_d)$ by reversing $Q(y_k - y_d)$ in time
(iii) apply $FQ(y_k - y_d)$ to the system experimentally and record the output $GFQ(y_k - y_d)$
(iv) reverse the output to obtain $FGFQ(y_k - y_d)$; compute $R^{-1} FGFQ(y_k - y_d)$;
(v) update u_{k+1} by computing

$$u_{k+1} = u_k - \beta[\underbrace{R^{-1} G^T Q(y_k - y_d)}_{\text{Term 1}} + \underbrace{\rho(u_k - u_d)}_{\text{Term 2}}]$$

(note that both terms in the above have now been obtained)

Note that compared to the off-line computation method, the online implementation involves an experiment in Step (iii) between two trials. This might seem to be more complicated but as will be seen later (in Section IV), this implementation leads to improved robustness properties and therefore should be the preferable choice if possible.

B. Convergence Properties

The proposed algorithm has some very nice convergence properties, as shown in the follow theorem:

Theorem 1: The proposed algorithm converges monotonically in the performance index $J(u, y)$, i.e.

$$\|y_{k+1} - y_d\|_Q^2 + \rho \|u_{k+1} - u_d\|_R^2 \le \|y_k - y_d\|_Q^2 + \rho \|u_k - u_d\|_R^2, \quad k \ge 0 \quad (17)$$

to the unique solution of the performance optimisation problem, i.e.

$$\lim_{k\to\infty}(u_k, y_k) = (u^*, y^*)$$
$$= \arg\min_{(u,y)} \left\{ \|y - y_d\|_Q^2 + \rho\|u - u_d\|_R^2 : y = Gu + d \right\}.$$

Proof. To show the monotonicity of the performance index, denoting

$$\Delta u_k = u_{k+1} - u_k = -\beta[G^*(y_k - y_d) + \rho(u_k - u_d)]$$

and substituting the updating law into the performance index gives

$$\begin{aligned} J(u_{k+1}, y_{k+1}) - J(u_k, y_k) &= \|y_{k+1} - y_d\|_Q^2 \\ + \rho\|u_{k+1} - u_d\|_R^2 &- \|y_k - y_d\|_Q^2 - \rho\|u_k - u_d\|_R^2 \\ &= \beta^2 \left(\rho\|\Delta u_k\|_R^2 + \|G\Delta u_k\|_Q^2\right) - 2\beta\|\Delta u_k\|_R^2 \end{aligned} \quad (18)$$

which is a quadratic function of β (if $\Delta u_k \neq 0$). It is easy to find that if

$$0 < \beta < 2/(\rho + \|G\|^2),$$

the value of the above quadratic function will be negative and thus

$$J(u_{k+1}, y_{k+1}) \leq J(u_k, y_k),$$

i.e. the performance index decreases monotonically.

To prove that the algorithm converges to the solution of the performance optimisation problem, we use a slightly different approach (the same approach will be used later in robustness property analysis).

Note that the updating law can be further written as

$$\begin{aligned} u_{k+1} - u_d &= u_k - u_d - \beta[G^*(y_k - y_d) + \rho(u_k - u_d)] \\ &= u_k - u_d - \beta[G^*Gu_k + G^*d - G^*y_d + \rho(u_k - u_d)] \\ &= [I - \beta(\rho I + G^*G)](u_k - u_d) + \beta G^*(y_d - Gu_d - d) \end{aligned}$$

As $0 < \beta < 2/(\rho I + \|G\|^2)$, the above is a contraction and thus as $k \to \infty$, $u_k - u_d$ converges to the unique fixed point

$$u_\infty - u_d = G^*(\rho I + GG^*)^{-1}(y_d - Gu_d - d) \quad (19)$$

Similarly it can be shown that $y_k - y_d$ converges to

$$y_\infty - u_d = -(\rho I + GG^*)^{-1}(y_d - Gu_d - d)$$

Denote

$$\lambda = (\rho I + GG^*)^{-1}(y_d - Gu_d - d).$$

It is seen that the proposed algorithm converges to

$$u_\infty - u_d = G^*\lambda,\ y_\infty - y_d = -\lambda,$$

which is a stationary point of the Lagrangian

$$\mathcal{L}(u, y, \lambda) = \|y - y_d\|_Q^2 + \|u - u_d\|_R^2 + \langle \lambda, y - Gu - d \rangle_Q$$

This together with the fact that the performance index is strict convex implies the algorithm converges to the desired unique solution of the performance optimisation problem, which complete the proof. ■

The above theorem shows that the proposed algorithm converges monotonically in the performance index to the best performance that can be achieved. Moreover, from (18) in the proof, it can be seen that the performance improvement from trial to trial is a quadratic function of the learning gain β. Therefore, if β is chosen to be closer to the minimum of the quadratic function, e.g. close to $1/(\rho + \|G\|^2)$, faster convergence of the performance index should be expected. This will be further illustrated later in the example section.

Remark 1: Computing the operator norm $\|G\|$. In order to choose the learning gain β, we will need to know $\|G\|$, i.e. the norm of the operator G. This can be calculated as follows using the model

$$\begin{aligned} \|G\| &= \sup_{u\neq 0} \frac{\|Gu\|_Q}{\|u\|_R} = \sup_{u\neq 0} \sqrt{\frac{\|Gu\|_Q^2}{\|u\|_R^2}} = \sup_{u\neq 0} \sqrt{\frac{u^TG^TQGu}{u^TRu}} \\ &= \sup_{v\neq 0} \sqrt{\frac{v^TR^{-1/2}G^TQGR^{-1/2}v}{v^Tv}} = \bar{\sigma}(Q^{1/2}GR^{-1/2}) \end{aligned} \quad (20)$$

in which $\bar{\sigma}(Q^{1/2}GR^{-1/2})$ is the largest singular value of matrix $Q^{1/2}GR^{-1/2}$. An upper bound of it can be estimated using the $\|G(z)\|_\infty$, details of which are omitted here for brevity.

IV. Robustness of the Proposed Algorithm

The previous section proposes a gradient based ILC algorithm and shows that the proposed algorithm will converge monotonically in the performance index to the optimal solution under the ideal situation that the available model is an accurate representation of the plant dynamics, which, however, is rarely the case in practice. In this section, the convergence performance of the proposed algorithm is examined when there exist model uncertities/mismatches. In particular, the uncertainties that can be tolerated by the algorithm are characterised in terms of the following definition which is a modified version of the so-called 'robust monotonic convergence' proposed in [15].

Robust Monotonic Convergence: An ILC algorithm has the property of robust monotone convergence with respect to a performance index in the presence of a defined set of model uncertainties if, and only if, for every choice of control on the first trial and for any choice of model uncertainty within the defined set, the resulting sequence of performance index converges monotonically.

Suppose the real plant G_p is different from the system model G and represented by

$$G_p = (I + \Delta G)G$$

where ΔG is the multiplicative model mismatch/uncertainty which is an $N \times N$ matrix in 'lifted form' representation. It turns out that the robustness properties of the proposed algorithm depends on how the algorithm is implemented. When the algorithm is implemented online using experiments on the plant between trials, the following theorem shows that the proposed algorithm can still converge to the optimal solution of the performance optimisation problem.

Theorem 2: The proposed algorithm using online implementation is robust monotonically convergent in the performance index $J(u, y)$, i.e.

$$\|y_{k+1} - y_d\|_Q^2 + \rho\|u_{k+1} - u_d\|_R^2 \leq \|y_k - y_d\|_Q^2 + \rho\|u_k - u_d\|_R^2, \quad k \geq 0 \quad (21)$$

to the best performance that can be achieved, i.e.

$$\lim_{k\to\infty} (u_k, y_k) = (u^*, y^*) = \arg\min_{(u,y)} \left\{\|y - y_d\|_Q^2 + \rho\|u - u_d\|_R^2 : y = G_p u + d\right\} \quad (22)$$

if

$$0 < \beta < 2/(\rho + \|G_p\|^2). \quad (23)$$

Proof. The theorem can be proved using similar techniques as in the proof of Theorem 1 by noting that

$$\Delta u_k = u_{k+1} - u_k = -\beta[G_p^*(y_K - y_d) + \rho(u_k - u_d)]$$

where G_p^* is the adjoint operator of the real plant. The details are omitted here for brevity. ■

From the above theorem, it can be seen that the proposed algorithm can still converge monotonically to the unique minimal solution of the performance optimisation problem (22), even when the exact plant model G_p is not available. This is clearly different from the normal optimisation problem where the optimal solution is sensitive to the model accuracy. In our algorithm, this has been possible as the system works in a repetitive manner thus the algorithm could learn from the system's previous behaviours to compensate for the modelling mismatches. Also note that a sufficient condition for (23) to hold is

$$\|\Delta G\| < \frac{1}{\|G\|}\sqrt{\frac{2}{\beta} - \rho}$$

From this, it can be seen that, for a larger set of model uncertainties to be tolerated, the learning gain β has to be smaller. A very small learning gain β will lead to slow convergence in the performance index which clearly demonstrates the classic tradeoff between performance and robustness.

Robustness properties of the algorithm using off-line computation can be analysed in a similar way - details are omitted here for space reasons. Unfortunately, using off-line implementation the algorithm does not converge to the optimal solution to the performance optimisation problem, which is clearly different from the case of using online implementation where the optimal solution is guaranteed. This is not surprising as when using online experiments more information about the real plant is utilised and therefore better performance can be achieved. On the other hand, using off-line computation saves the effort of conducting real experiments - a compromise between the experimental effort and achieved performance has to be made.

V. Validation of the Proposed Design

In this section, numerical simulations are presented to demonstrate the effectiveness of the proposed design.

A. A non-minimum phase test facility

A model of an electro-mechanical non-minimum phase test facility consisting of a rotary mechanical system of inertias, dampers, torsional springs, a timing belt, pulleys and gears, has been used to evaluate the proposed method. The test facility is a demanding platform which has been used in the assessment of a wide variety of ILC schemes (see, for example [16], [17]). A 1000 pulse/rev encoder records the output shaft position and a standard squirrel cage induction motor supplied by an inverter, operating in Variable Voltage Variable Frequency (VVVF) mode, drives the load. The plant uses a PID loop in order to act as a pre-stabilizer, and the resulting closed-loop system constitutes the system to be controlled using ILC. The system can be represented using the non-minimum-phase, continuous time plant transfer-function

$$G(s) = \frac{149.36(4 - s)}{s^4 + 21.5s^3 + 170.28s^2 + 368.52s + 663.82} \quad (24)$$

which was identified in previous work [16] using a frequency response test method and shown to be a reasonably good representation of the system dynamics. The model is sampled using a zero-order hold and a sampling time of 0.1s to yield a discrete time state space model to be used in the simulation.

The design objective is to design an input signal such that the output shaft position is as close as possible to the desired operating position $r(t)$ given below

$$r(t) = \begin{cases} 5\cos(\pi t), & 0 \leq t \leq 1.5 \\ 0, & 1.5 < t \leq 4.5 \\ 5\sin(\pi t - \pi/2) & 4.5 < t \leq 6 \end{cases} \quad (25)$$

over a duration of 6s repeatedly, at the same time, minimising the input energy required. This is formulated into a performance optimisation problem with

$$J(u, y) = \|y - r\|_Q^2 + \|u\|_R^2,$$

i.e. choosing $y_d = r, u_d = 0$ and $\rho = 1$ in the general performance index (10). For simplicity, the matrices Q and R are chosen to be $Q = I$ and $R = 0.1 \times I$ where I is an identity matrix with appropriate dimensions.

B. Performance of the algorithm

The simulations study a practical scenario where the accurate model information is not available. Suppose that during the identification process, the nondominant poles $-9.6789 \pm 5.4546i$ of the model (24) were not identified at all and only the dominant poles $-1.0711 \pm 2.0569i$ were included; furthermore, the DC gain of the model was underestimated by 50%. The resulting model is denoted as G_1 with transfer function

$$G_1(s) = \frac{1.344(4 - s)}{s^2 + 2.142s + 5.378}$$

representing a situation where significant model uncertainties are present.

The simulation compares the performance of four scenarios:

1) Online implementation of the proposed algorithm with learning gains $\beta = 0.05$ and 0.1, respectively
2) Off-line implementation of the proposed algorithm with learning gains $\beta = 0.05$ and 0.1, respectively
3) Calculating the optimal solution by directly solving the optimal control problem using the available (inaccurate) system model and apply the obtained input to the plant, compute the performance index - this represents a classic (perhaps the most commonly used) solution
4) Assuming the accurate model is known, solving the optimal control problem and compute the performance index - this represents the best possible result that can be achieved.

The results are shown in Figure 1. From the figure, it is

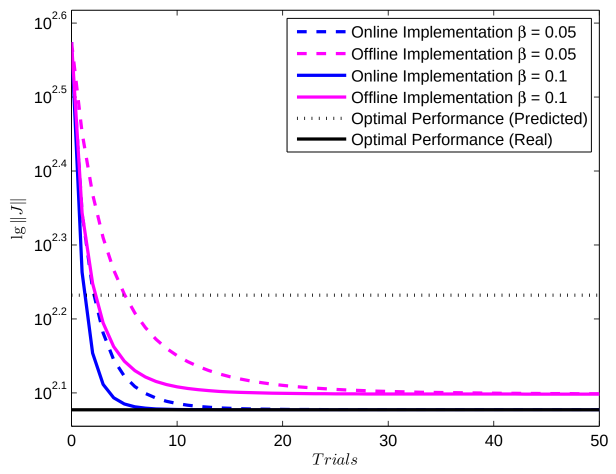

Fig. 1. Simulation: performance index convergence results over 50 trials

clear that using the classic solution based on an (inaccurate) system model, the best performance that can be obtained is 170.6. Using the proposed method, significant improvements in the performance are achieved. In particular, using off-line implementation the proposed algorithm converges to 125.4 in the performance index and using online implementation the proposed algorithm converges to 119.6, which is the optimal performance that can be ever achieved (calculated using the assumed known accurate system model (24)). Note that this is accomplished when there is no exact model information available, which is of great interest in practical applications.

VI. Conclusion

Iterative learning control is a popular control design method for high performance trajectory tracking of systems operating in a repetitive manner. This paper further extends the applicability of ILC by showing that it can be used to solve a general performance optimisation problem for which the trajectory tracking is just a special case. The problem is formulated into an optimisation framework using an abstract Hilbert space setting. A gradient based method is proposed as a solution and its convergence and robustness properties are analysed in detail. It has been shown that the proposed algorithm will converge to the best solution of the performance optimisation problem, even when the model available is different from the real plant, which is of particular practical interest. Numerical simulations using an electro-mechanical non-minimum phase system model verify the effectiveness of the proposed design.

The paper focuses on linear discrete time invariant systems but the method applies more generally to linear discrete time varying systems and linear continuous time systems; the detailed realisations of the results, however, might change. Theoretically, the results in this paper can also be extended to point-to-point ILC problem and to the case where there exist system constraints. These topics constitute part of our future research and will be reported separately.

References

[1] David H Owens. *Iterative Learning Control: An Optimization Paradigm*. Advances in Industrial Control. Springer, 2016.

[2] Y.Wang and Z. Hou. Terminal iterative learning control based station stop control of a train. *International Journal of Control*, 84(7):1263–1277, 2011.

[3] H. Ding and J. Wu. Point-to-point control for a high-acceleration positioning table via cascaded learning schemes. *IEEE Transactions on Industrial Electronics*, 54(5):2735–2744, 2007.

[4] J. Park, P. H. Chang, H. S. Park, and E. Lee. Design of learning input shaping technique for residual vibration suppression in an industrial robot. *IEEE/ASME Transactions on Mechatronics*, 11(1):55–65, 2006.

[5] J. van de Wijdeven and O. Bosgra. Residual vibration suppression using hankel iterative learning control. *International Journal of Robust Nonlinear Control*, 18(10):1034–1051, 2008.

[6] D. H. Owens, C. T. Freeman, and T. V. Dinh. Norm-optimal iterative learning control with intermediate point weighting: Theory, algorithms, and experimental evaluation. *IEEE Transactions on Control Systems Technology*, 21(3):999–1007, 2013.

[7] C. T. Freeman and Y. Tan. Iterative learning control with mixed constraints for point-to-point tracking. *IEEE Transactions on Control Systems Technology*, 21(3):604–616, 2013.

[8] P. Janssens, G. Pipeleers, and J. Swevers. A data-driven constrained norm-optimal iterative learning control framework for LTI systems. *IEEE Transactions on Control Systems Technology*, 21(2):546–551, 2013.

[9] T. D. Son, H. S. Ahn, and K. L. Moore. Iterative learning control in optimal tracking problems with specified data points. *Automatica*, 49(5):1465–1472, 2013.

[10] D. H. Owens, C. T. Freeman, and B. Chu. Multivariable norm optimal iterative learning control with auxiliary optimisation. *International Journal of Control*, 86(6):1026–1045, 2013.

[11] D. H. Owens, C. T. Freeman, and B. Chu. An inverse-model approach to multivariable norm optimal iterative learning control with auxiliary optimisation. *International Journal of Control*, 87(8):1646–1671, 2014.

[12] I. Lim and K. L. Barton. Pareto iterative learning control: Optimized control for multiple performance objectives. *Control Engineering Practice*, 26:125–135, 2014.

[13] J. Hatonen, D.H. Owens, and K.L. Moore. An algebraic approach to iterative learning control. *International Journal of Control*, 77(1):45–54, 2004.

[14] A. Bryson and Y.C. Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Abingdon, UK, 1975.

[15] D. H. Owens, J.J. Hatonen, and S. Daley. Robust monotone gradient-based discrete-time iterative learning control. *International Journal of Robust and Nonlinear Control*, 19(6):634–661, 2009.

[16] C.T. Freeman, P.L. Lewin, and E. Rogers. Experimental evaluation of iterative learning control algorithms for non-minimum phase plants. *International Journal of Control*, 78(11):826–846, 2005.

[17] Z. Cai, C. T. Freeman, P. Lewin, and E. Rogers. Iterative learning control for a non-minimum phase plant based on a reference shift algorithm. *Control Engineering Practice*, 16(6):633–643, 2008.