Int. J. Appl. Math. Comput. Sci., 2008, Vol. 18, No. 3, 279–293
DOI: 10.2478/v10006-008-0026-7

ITERATIVE LEARNING CONTROL — MONOTONICITY AND OPTIMIZATION

DAVID H. OWENS, STEVE DALEY

Department of Automatic Control and Systems Engineering
University of Sheffield, Mappin Street, Sheffield S1 3JD, UK
e-mail: d.h.owens@sheffield.ac.uk

The area if Iterative Learning Control (ILC) has great potential for applications to systems with a naturally repetitive action where the transfer of data from repetition (trial or iteration) can lead to substantial improvements in tracking performance. There are several serious issues arising from the "2D" structure of ILC and a number of new problems requiring new ways of thinking and design. This paper introduces some of these issues from the point of view of the research group at Sheffield University and concentrates on linear systems and the potential for the use of optimization methods and switching strategies to achieve effective control.

Keywords: iterative learning control, optimization, monotonicity, robust control.

1. Introduction

Iterative learning control is technique for improving tracking response in systems that repeat a given task over and over again (each repetition sometimes being called a *pass* or *trial*). It is assumed that a plant model

$$\begin{cases} x(t+1) = \Phi x(t) + \Gamma u(t), & x(0) = x_0, \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{1}$$

is given where A, B, C and D are matrices of appropriate dimensions, $u(\cdot)$ is the input variable, $x(\cdot)$ is the state variable, x_0 is the initial condition for $x(\cdot)$, and $y(\cdot)$ is the output variable. For notational simplicity, it is assumed that $D = 0$. Furthermore, a reference signal $r(t)$ is given over a finite time-interval $t \in [0, T]$. The control objective is to find a control input $u(t)$ so that the corresponding output $y(t)$ tracks $r(t)$ precisely. In contrast to the standard tracking problem, when the system (1) has reached the final time point $t = T$, the state x of the system is reset to the *same* initial condition x_0, and, after resetting, the system is again required to track the *same* reference signal $r(\cdot)$. Several important industrial applications fit into the ILC framework. Reported applications of ILC include robotics (Zilouchian, 1994; Norrlöf, 2002), chemical batch processing (Lee *et al.*, 1996*a*), and servo systems (Lee and Lee, 1993), to name but a few. For a detailed discussion on ILC applied to industrial problems, see (Longman, 2000).

Arimoto *et al.* (1984) originally formalized ILC for an academic community by suggesting that information from repetitions $1, 2, \ldots, k-1$ could be used to construct a new, improved input time series u_k, where k is the repetition number. He also demonstrated, in a simple way, that this can be done, in principle, in a way that ensures that the tracking error will ultimately go to zero as the number of repetitions increases. In summary, ILC has the property that experience from previous repetitions or iterations can be used to ensure that the ILC system will gradually learn the control action that will result in perfect tracking.

Remark 1. Note that in the ILC community it is now widely accepted that (Uchiyama, 1978) is the first publication to introduce the ILC concept. However, because this publication is written in Japanese, non-Japanese researchers were not aware of this publication when the ILC research initially started in the USA and Western Europe. Therefore, the publication (Arimoto *et al.*, 1984) was referenced as being the starting point for ILC research. However, it seems that there are earlier publications than Uchiyama's on topics related to ILC. For example, in (Cryer *et al.*, 1976), the authors (who worked for General Motors Truck and Bus) proposed the following discrete-time 'iterative control' (as they call it) in the frequency domain:

$$u_{k+1}(e^{j\omega T_s}) = u_k(e^{j\omega T_s}) + \beta G^{-1}(e^{j\omega T_s}) e_k(e^{j\omega T_s}), \tag{2}$$

where $\beta \in \mathbb{R}, 0 < \beta < 1$ is a design parameter and $G(e^{j\omega T_s})$ is the discrete-time Fourier transform of the im-

pulse response $g(t)$ of the plant in question. This is clearly an ILC algorithm and the authors apply this algorithm in the context of laboratory road simulation. Another even earlier reference to ILC concepts seems to be the USA Patent US3555252 – Learning control of actuators in control systems of 1971, see (Chen and Moore, 2000) for details.

Example 1. In order to clarify the difference between familiar feedback control and ILC, consider a dynamical system

$$(p^2 + 5p + 6)y(t) = (p + 1)u(t), \tag{3}$$

where $p := \frac{\mathrm{d}}{\mathrm{d}t}$ and $t \in [0, 6]$. This system is sampled at intervals of $T_s = 0.1$ seconds using zero-order hold resulting in a discrete-time plant model (Φ, Γ, C). The system is required to track a reference signal $r(t) = \sin(2\pi t/6)$ and the system is controlled with a PID-controller,

$$u(t) = K_p e(t) + K_I \int_0^t e(\tau)\,\mathrm{d}\tau + K_D \dot{e}(t), \tag{4}$$

where $K_p = 15$, $K_I = 8$ and $K_D = 0$, which is also sampled using zero-order hold with the same sampling time. Fig. 1 shows the output $y(t)$, implying that the system is only capable of tracking the reference signal to a moderate degree of accuracy. Note that the same tracking result is obtained during each repetition, because the PID-controller has fixed parameters. Fig. 2, on the other hand, shows the l_2-norm (Euclidean norm) of the tracking error $e_k(t) := r(t) - y_k(t)$ as a function of the iteration index (number or round) k with the ILC algorithm

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t + 0.1), \tag{5}$$

where $\gamma = 9$. Note that at first sight this algorithm seems to be non-causal, because $u_{k+1}(t)$ is a function of $e_k(t + 0.1)$. However, on iteration $k + 1$, $e_k(t)$ is *past* data and it is available for the whole range $t \in [0, T]$. In general, it is possible (and typically necessary) to make $u_{k+1}(t)$ a function of $e_k(s)$ for $s \geq t$. Based on Fig. 2 it seems that the tracking error does indeed tend to zero as $k \to \infty$, but the convergence is not monotonic in the sense that errors do increase in early repetitions but ultimately reduce to zero. In the following section it will be shown that this algorithm converges to a zero tracking error for an arbitrary time-invariant linear plant (assuming that $C\Gamma \neq 0$) if the learning gain γ satisfies the inequality $|1 - \gamma C\Gamma| < 1$.

♦

Since the initial work by Cryer, Uchiyama and Arimoto, ILC has established itself as an independent research topic in the control community. The review paper (Moore, 1998), for example, contains roughly 300 references on ILC. There have also been several books written on ILC, including (Moore, 1993; Bien and Xu, 1998; Chen and Wen, 1999; Xu and Tan, 2003). The number of Ph. D. theses written on ILC, including (Amann, 1996; de Roover, 1997; Norrlöf, 2000; Hätönen, 2004), is another indicator of ILC entering the main stream of control theory.

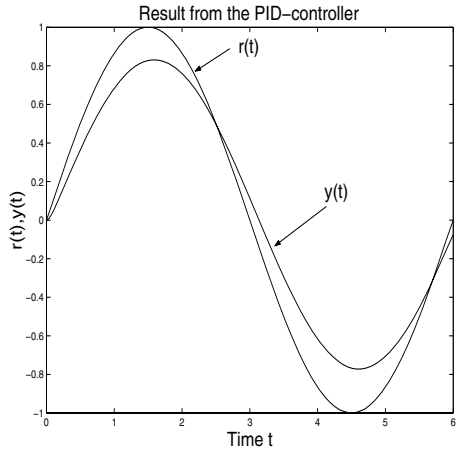

Fig. 1. Response for the PID-controller (4).

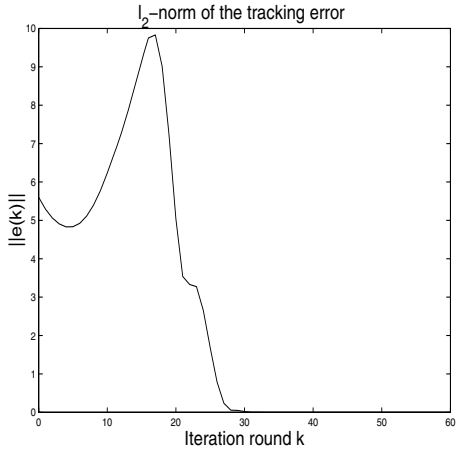

Fig. 2. Convergence behaviour of the Arimoto algorithm.

2. Formal definition of ILC

The *system model* is central to the analysis. To derive a precise mathematical definition of the ILC problem, consider the following standard continuous-time time-varying linear state-space model defined over a *finite* time-interval $t \in [0, T]$ (defining the length of time allocated for each task):

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)u(t), \qquad x(0) = x_0, \\ y(t) &= C(t)x(t), \end{aligned} \tag{6}$$

or the corresponding linear time-varying discrete-time system with sampling time $T_s = h$,

$$\begin{aligned} x(t+h) &= \Phi(t)x(t) + \Gamma(t)u(t), \qquad x(0) = x_0, \\ y(t) &= C(t)x(t), \end{aligned} \tag{7}$$

where the state $x(\cdot) \in \mathbb{R}^n$, the output $y(\cdot) \in \mathbb{R}^m$, the input $u(\cdot) \in \mathbb{R}^m$. The operators $A(\cdot), \Phi(\cdot), B(\cdot), \Gamma(\cdot)$ as well as $C(\cdot)$ in (6) and (7) are matrices of appropriate dimensions. Note that in (6) time t is a continuous parameter whereas in (7) it only takes discrete values $t = 0, h, 2h, \dots T$. In order to avoid technical difficulties in the analysis, it is typically assumed that matrices are either constant or, more generally, continuous with respect to time t.

The *control objective* is defined in terms of a reference signal $r(t)$. The task is to construct $u(t)$ so that $y(t)$ tracks $r(t)$ as accurately as possible. In addition, the system (6) (or the corresponding discrete-time model (7)) is required to follow the reference signal in a repetitive manner, i.e., after the system has reached the final time point $t = T$, the state of the system is reset to the initial condition x_0 and the system attempts to track the same reference signal $r(t)$ again.

Control systems design in ILC takes a quite specific form. If $u_k(t)$ is the input applied at trial $k \in \mathbb{N}$ and $e_k(t) := r(t) - y_k(t)$ is the resulting tracking error, the control design problem is to construct a control law (or an algorithm) expressed, for generality, as a functional relationship typified by the equation

$$u_{k+1}(t) = f\big(e_{k+1}(\cdot), \dots e_{k-s}(\cdot), u_k(\cdot), \dots, u_{k-r}(\cdot)\big). \tag{8}$$

The control law is required to have the property that $\lim_{k\to\infty} u_k = u^*$ (i.e., the control signal converges to the control signal that generates r exactly. As a consequence, $\lim_{k\to\infty} e_k = 0$ in a suitable topology. Furthermore, for real-world causality (and hence implementation) reasons, it is specifically required that the dependence of $u_{k+1}(t)$ on e_{k+1} only requires knowledge of its values on $0 < s \leq t$.

Existence of a solution. Note that in the problem definition it is assumed that there exists an input u^* which gives perfect tracking. If this is not the case, the problem can be modified to require that the algorithm should converge to a fixed point u^* where u^* is the solution of the optimization problem

$$u^* = \arg\min_{u\in\mathcal{U}} \|r - Gu\|^2, \tag{9}$$

where $\mathcal{U}$ is set of possible inputs and $\|\cdot\|$ is a suitable norm.

Assuming that perfect tracking is possible, we have $r = Gu^*$ and hence u^* can be formally obtained from the reference and initial condition data using the inverse of the plant model. Such an approach leads to an ILC algorithm with the property of convergence in one iteration. To see this in the linear case, consider the inverse algorithm for linear systems

$$u_{k+1} = u_k + G_e^{-1} e_k. \tag{10}$$

A simple calculation then indicates that

$$\begin{aligned} e_1 &= r - y_1 = r - Gu_1 \\ &= r - G[u_0 + G^{-1}e_0] = 0, \end{aligned} \tag{11}$$

independent of the choice u_0 (and hence e_0).

This algorithm is clearly a theoretical construct. It cannot be realized in practice due to the fundamental problem that an exact plant model is never available. 'Infinite' iteration must therefore be accepted as a practical reality.

It is ironic that theoretical studies of ILC are often based on assumed exact models of the plant. This is justified scientifically by the need to understand ILC in the nominal case and create benchmark algorithms with known properties. This approach requires that the robustness of the methods in the presence of modelling error be tested both through theoretical analysis and experimental work.

The original and modified ILC problem definition generate at least the following two fundamental design questions that have to be answered before any further algorithm design can take place:

- Is the algorithm design to be done with the continuous-time system (6) or with the discrete-time model (7)?
- What topology (norm or measure of error magnitude) $\|\cdot\|$ is to be used in the algorithm design?

The first question is in fact a very subtle one: Because in ILC the control algorithm uses information from previous repetitions, this information has to be recorded using suitable media. However, this is only possible with digital devices, and hence it seems to be natural to use the discrete-time model (7). However, as was demonstrated in previous work on ILC, see (Amann, 1996), convergence results obtained for discrete-time models can be sometimes over-optimistic and hide problems with robustness when the discrete-time algorithm is applied on a plant that is originally a continuous-time system. Thus it could be argued that the continuous-time model (7) is most useful in control design. This discussion indicates that, whatever model type is chosen, the theoretical convergence results have to be always checked with a simulation model which takes into account the hybrid nature of the problem, i.e., the controller is a discrete-time system and the plant in is a continuous-time system.

The answer to the second question could be the following one: select a normed space that is easy to work with. For example, in this paper, the infinite-dimensional

function space $L_2[0,T]$ and the finite dimensional Euclidean space of sequences (time series) on $[0,T]$ are used exclusively. Sequences (of ILC iterates) in these spaces are regarded as familiar square-summable sequences in the space l_2. This is due the fact that all such spaces are complete inner-product spaces, and optimization in these spaces is far simpler than in a general Banach space setting. The Hilbert space setting is quite general and, for linear systems, results in the rather complete theory of norm-optimal iterative learning control introduced by the first author, which will be discussed further on.

Note: Exponentially weighted spaces have a place in the toolbox of tractable ILC analysis but are not considered here for brevity. Other, analytically more difficult, spaces and norms can be considered if computational approaches are used to solve the problem.

Convergence is naturally the most important requirement for an ILC algorithm. However, additional requirements have been suggested in the research literature (see, for example, (Bien and Xu, 1998)), one of the most common ones being as follows:

(i) Complexity: Convergence should be achieved with a minimal amount of information about the plant.

(ii) Robustness: Convergence should be achieved even if there is uncertainty in the plant model.

(iii) Resetting: Some form of convergence should be achieved even if the resetting is not perfect, i.e., the initial condition for each iteration lies inside a ball of radius δ centred at x_0 or, in a more mathematical notation, $x_k(0) \in B_\delta(x_0) \in \mathbb{R}^n$ for $k = \mathbb{N}$.

Note: Some authors consider the first assumption as an essential part of ILC. However, in, e.g., robotics, either accurate models are available from the robot manufacturer or they can be obtained rather easily using modern identification techniques. Consequently, it seems unwise (and rather strange) to discard this plant information in ILC algorithm design.

3. ILC and two-dimensionality

As was explained in the previous section, the basic idea behind ILC is to use the repetitive nature of the problem definition to allow the system to learn the input function that results in asymptotically perfect tracking. The learning mechanism, however, introduces a new axis, namely, the iteration axis k. This results in a two-dimensional system, where the independent variables (t,k) are the finite time axis $t \in [0,T]$ and the infinite iteration axis $k \in \mathbb{N}$. As a first step towards a convergence/stability analysis, note that, due to the finite nature of the time axis, the output of a finite-dimensional linear time-varying system can never become unbounded in finite time. Hence, in contrast to classical feedback control, it can be expected that the properties of the ILC system along the time axis play an important but not dominant role in formal convergence analysis. The iteration axis, on the other hand, is infinite. As the number of iterations increases, it is intuitive that the infinite sequence ($k = 0, 1, 2, 3, \dots$) of 'output profiles' $y_k(t)$ for $t \in [0,T]$ can converge, diverge or oscillate, depending on the chosen learning mechanism (control law). Also, even if convergent, the individual signals may become unacceptably large before convergence finally occurs.

How then can the question of convergence/stability be approached mathematically? In answering this question the following example is useful.

Example 2. Consider the following ILC algorithm:

$$u_{k+1}(t) = u_k(t) + [Ke_k](t), \tag{12}$$

where K is a 'learning-gain operator', and the plant model (represented as a linear operator G plus an initial condition term z_0) has the form

$$y_{k+1}(t) = [Gu_{k+1}](t) + z_0(t) \tag{13}$$

for $t \in [0,T]$. In order to analyse the convergence properties of this algorithm, it is necessary to find how the tracking error $e_k(t) := r(t) - y_k(t)$ evolves as a function of the iteration index k. To find this 'evolution equation', apply the control algorithm (12) to the plant model $y_{k+1}(t) = [Gu_{k+1}](t) + z_0(t)$ to give

$$[Gu_{k+1}](t) + z_0(t) = [Gu_k](t) + z_0(t) + [GKe_k](t). \tag{14}$$

Using the process model (13) and the definition of the tracking error $e_k(t)$, this equation can be written equivalently as

$$e_{k+1}(t) = [Le_k](t), \tag{15}$$

where $L = (I - GK)$. Hence, L is the operator that maps $e_k(\cdot)$ to $e_{k+1}(\cdot)$, and it is clear that its mathematical properties fully define the properties of the ILC algorithm, including convergence.

In the previous example, L is often called the *learning operator* that maps the tracking error from trial to trial. In fact, most of the existing linear ILC algorithms in the research literature result in the error evolution equation $e_{k+1} = Le_k,\ k = 0, 1, 2, \dots$, or, more generally,

$$e_{k+1} = Le_k + b, \qquad k = 0, 1, 2, \dots. \tag{16}$$

It is important to analyse the conditions under which this kind of iterative process converges. This analysis can be set firmly in the context of *multi-pass* or *repetitive* systems theory developed since the mid-1970s by the first author and colleagues (based on well-known branches of

applied analysis), and it leads to a number of convergence conditions. The most general of these is a spectral radius condition

$$\rho(L) < 1, \tag{17}$$

where $\rho(\cdot)$ is the spectral radius of a given operator). The spectral radius condition is implied by a simpler contraction mapping condition (i.e., $\|L\| < 1$, where $\|\cdot\|$ is a suitable operator norm). In general, the spectral radius condition guarantees only *asymptotic* convergence whereas the contraction mapping condition guarantees *monotonic* convergence. Readers are invited to study these ideas as they provide an excellent introduction to the complexities of iterative theory from a control point of view and also add useful background details to what follows. For a more general stability analysis of 2D-systems, see (Edwards and Owens, 1982; Rogers and Owens, 1992).

4. Convergence properties

As a motivational example, consider the following discrete-time version of the Arimoto-law (see (Moore, 1993) for details):

$$u_{k+1}(t) = u_k + \gamma e_k(t+1), \tag{18}$$

which is used to control the system (1). In order to analyse the performance of this algorithm, note that, because in ILC the time-axis is *finite*, the system model (1) can be written equivalently in a "super-vector" form as $\vec{y}_k = G_e\vec{u}_k + \vec{d}$, where

$$G_e = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ C\Gamma & 0 & 0 & \dots & 0 \\ C\Phi\Gamma & C\Gamma & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C\Phi^{N-2}\Gamma & C\Phi^{N-3}\Gamma & \dots & \dots & 0 \end{bmatrix},$$

$$\vec{d} = \begin{bmatrix} Cx_0 \\ C\Phi x_0 \\ C\Phi^2 x_0 \\ \vdots \\ C\Phi^{N-1}x_0 \end{bmatrix}. \tag{19}$$

The elements $C\Phi^j B$ of the matrix G_e are the Markov parameters of the plant (1), and d is the initial condition response. Furthermore, the 'super-vectors' $\vec{u}_k$ and $\vec{y}_k$ are defined as

$$\begin{aligned} \vec{u}_k &:= [u_k(0)\ u_k(1)\ \dots\ u_k(N-1)]^{\mathrm{T}}, \\ \vec{y}_k &:= [y_k(0)\ y_k(1)\ \dots\ y_k(N-1)]^{\mathrm{T}}. \end{aligned} \tag{20}$$

The tracking error $\vec{e}_{k+1}$ is defined with the equation

$$\begin{aligned} \vec{e}_{k+1} &:= \vec{r} - \vec{y}_{k+1} = r - \left(G_e\vec{u}_{k+1} + \vec{d}\right) \\ &= \left(r - \vec{d}\right) - G_e\vec{u}_{k+1} \end{aligned} \tag{21}$$

and, consequently, the possibly non-zero initial condition response can be embedded into the reference signal via the map $r \to r - d$. From now on (without loss of generality), it is therefore assumed that $\vec{d} = 0$ or, equivalently, $x_{k+1} = 0$. The control law (18) can be written in the super-vector notation as $\vec{u}_{k+1} = \vec{u}_k + G_c\vec{e}_k$, where

$$G_c = \begin{bmatrix} 0 & \gamma & 0 & \dots & 0 \\ 0 & 0 & \gamma & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \gamma \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}. \tag{22}$$

Using similar algebraic manipulations as in Example 2, Section 3, it can be shown that the Arimoto law leads to an error evolution equation $\vec{e}_{k+1} = L\vec{e}_k$,

$$L = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1-\gamma C\Gamma & 0 & \dots & 0 \\ 0 & -\gamma C\Phi\Gamma & 1-\gamma C\Gamma & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & -\gamma C\Phi^{N-2}\Gamma & -\gamma C\Phi^{N-3}\Gamma & \dots & 1-\gamma C\Gamma \end{bmatrix}. \tag{23}$$

The condition for asymptotic convergence is that the spectral radius of L be strictly less than one (or, equivalently, the modulus of each eigenvalue of L be strictly less than one). Because L is a lower-triangular matrix, the eigenvalues λ_i of L are the diagonal elements of L, i.e., $\lambda_1 = 1$ and $\lambda_i = 1 - \gamma C\Gamma$ for $i = 2, \dots, N$. If $r(0) = Cx_k(0)$ for each k, it can be shown using a 'lifting technique' (see (Hätönen, Owens and Moore, 2004) for details) that the unity eigenvalue is associated with the initial conditions (unchanged by the control action), and hence the only effective eigenvalue is $\lambda = 1 - \gamma C\Gamma$ of the multiplicity of $N-1$ and thus $\rho(L) = |1 - \gamma C\Gamma|$.

This is a remarkable result, because the convergence depends only on C, Γ and γ (i.e., $|1 - \gamma C\Gamma| < 1$) and not on the internal dynamics of the plant as described by the matrix Φ. The practical problem is that the convergence is only asymptotic and, therefore, as Fig. 2 indicates, the l_2-norm (or any other norm) can grow to a very large value on some iterations before terminal convergence. Clearly, the tracking performance of the algorithm can be extremely poor when the l_2 norm is large. This is, in most practical applications of ILC, not acceptable, because a large

l_2 norm may relate to a low product quality, a loss of time or materials, or it might possibly result in permanent damage in the system being controlled. Therefore, even though the Arimoto algorithm and other theoretically convergent *model independent* algorithms are very appealing theoretically, their applicability to real ILC problems is questionable. There is a real need in the industry for algorithms that result in improved error performance and preferably have some form of monotonic convergence, e.g., $||e_{k+1}|| \leq ||e_k||, \quad \forall k \geq 0$. This is discussed in the next section.

5. Norm-optimal ILC. A benchmark ILC solution

In this section an optimization based design concept called Norm-Optimal Iterative Learning Control (NOILC) is introduced and its mathematical properties are derived. Note that using optimization to design ILC algorithms has generated a lot of interest in the ILC community, see, e.g., (Togai and Yamano, 1985; Gorinevsky, 1992; Tao *et al.,* 1994; Gunnarsson and Norrlöf, 2000).

The material presented in this section is based on the work of the first author and his team as initiated by the paper (Amann, 1996). Only the main result in terms of convergence analysis will be presented. In this section only the relevant propositions are presented. If the reader is interested in the actual proofs, he/she should consult references such as (Amann, 1996; Amann *et al.,* 1996; Amann *et al.,* 1998; Hätönen and Owens, 2004).

The starting point in NOILC is to write the plant model in the abstract operator form

$$y = Gu + z_0, \tag{24}$$

where G is the system input-output (convolution) operator, $u \in \mathcal{U}$ and $y, z_0 \in \mathcal{Y}$, where $\mathcal{Y}$ and $\mathcal{U}$ are the input and output spaces, respectively. In NOILC, both $\mathcal{U}$ and $\mathcal{Y}$ are chosen to be real Hilbert spaces. Furthermore, G is assumed to be a linear and bounded operator from $\mathcal{U}$ to $\mathcal{Y}$. In (24), z_0 describes the effect ofnon-zero initial conditions on the plant output and, as was shown in Section 4, it can be assumed that $z_0 = 0$ without loss of generality. As a motivation for the use of optimization, note that the ultimate goal of any ILC algorithm is to ensure that $e = 0$, i.e., to iteratively solve (in the limit) the optimization problem

$$\min_{u \in U} \|e\|^2 \tag{25}$$

subject to plant dynamics and the relation $e = r - Gu$. From now on it is assumed that the reference signal belongs to the range of the plant G, and hence an optimizing solution u^* satisfying $e^* = r - Gu^* = 0$ does exist.

Theoretically, if G and r are known, and r belongs to the range of G, and G is injective, the unique optimizing solution u^* can be solved directly with the equation $u^* = G^{-1}r$. In practice, however, G is never known precisely, and alternative ways of solving the optimization problem (25) have to be considered. In NOILC, optimal u^* is generated by solving the following *sequence* of optimization problems for iterates u_{k+1} :

$$\min_{u_{k+1} \in U} J_{k+1}(u_{k+1}), \tag{26}$$

where

$$J_{k+1}(u_{k+1}) = \|e_{k+1}\|^2 + \|u_{k+1} - u_k\|^2 \tag{27}$$

subject to the constraint equation $e_{k+1} = r - Gu_{k+1}$. The norms are assumed to be induced norms from the inner products $\langle \cdot, \cdot \rangle_{\mathcal{Y}}$ in $\mathcal{Y}$ and $\langle \cdot, \cdot \rangle_{\mathcal{U}}$ in $\mathcal{U}$, respectively; in other words, $\|e_{k+1}\|^2_{\mathcal{Y}} = \langle e_{k+1}, e_{k+1} \rangle_{\mathcal{Y}}$ and $\|u_{k+1} - u_k\|^2_{\mathcal{U}} = \langle u_{k+1} - u_k, u_{k+1} - u_k \rangle_{\mathcal{U}}$. The first term in (27) reflects the design criterion of $\|e_{k+1}\|^2$ being small during every repetition. The second term penalizes the input functions u_{k+1} that are 'too' different from the previous trial input u_k. There are at least two arguments that explain why this term should be useful in the cost function:

- The resulting algorithm will be of the form $u_{k+1} = u_k + \delta_{k+1}$, and hence the algorithm will contain the internal model of $r_k = r$, which is necessary for zero convergence (Hätönen, Owens and Moore, 2004).
- The input sequence $\{u_k\}$ will be smooth along the iteration axis k, and it will introduce elements of *caution* and *robustness* into the algorithm.

Note: In (Lee *et al.*, 1996*b*), the same cost function was considered, and this work was also published in 1996; therefore, the idea of using this particular cost function can be traced back to at least two independent sources. The work of (Amann, 1996) has the considerable benefit that a Hilbert space setting enables the general construction of ILC laws that apply to both continuous and sampled data plants plus plants with delays and many other complex dynamic (albeit linear) effects.

To illustrate the useful properties of the resulting algorithm, let u^*_{k+1} be the optimal solution for iteration $k+1$ and e^*_{k+1} the corresponding optimal tracking error for iteration $k+1$. With this notation the non-optimal choice $u_{k+1} = u_k$ gives the following interlacing result:

$$\|e^*_{k+1}\|^2 \leq J(u^*_{k+1}) \leq \|e^*_k\|^2, \tag{28}$$

and hence the algorithm will result in monotonic convergence, i.e., $\|e^*_{k+1}\| \leq \|e^*_k\|$. Note that for this interlacing result only the boundedness of G and the existence of an optimizing solution u^*_{k+1} (which can be non-unique) is required, and the linearity of G does not play any role! Hence it can be expected that the NOILC algorithm can also work for non-linear plant models. Initial results on this topic can be found from (Hatzikos *et al.*, 2004), where the authors use genetic algorithms to solve the NOILC optimization problem for non-linear plants.

5.1. Abstract convergence analysis. In order to show rigorously that the algorithm converges to the correct solution u^* in the linear case, the first step is to solve the optimization problem (26). This is achieved by calculating the Frechet derivative of (26) and setting it to zero, which is the necessary and sufficient condition for optimality. The Frechet derivative can be calculated from the equation

$$\langle J_{k+1}(u_{k+1}), \delta u_{k+1}\rangle = \frac{d}{d\epsilon} J_{k+1}(u_{k+1} + \epsilon \delta u_{k+1}) = 0, \tag{29}$$

and it turns out that the optimal solution is

$$u_{k+1} = u_k + G^* e_{k+1}, \tag{30}$$

where G^* is the adjoint operator of G in the chosen Hilbert space topologies. Multiplying (30) with G gives

$$y_{k+1} = y_k + GG^* e_{k+1}, \tag{31}$$

which then gives

$$(I + GG^*)e_{k+1} = e_k. \tag{32}$$

It can be easily shown that the operator $I + GG^*$ in (32) is positive and, hence, $(I + GG^*)$ is invertible while the error evolution equation becomes

$$e_{k+1} = (I + GG^*)^{-1} e_k = L e_k, \tag{33}$$

where $L := (I + GG^*)^{-1}$ is the 'learning operator'. The following proposition can be derived.

Proposition 1. *Assume that $r \in \mathcal{R}g\,(G)$, where $\mathcal{R}g\,(G)$ is the range of the operator G. Then the sequence $\{u_k\}$ satisfies $\lim_{k\to\infty} \|u_{k+1} - u_k\| = 0$, and the sequence e_k satisfies $\lim_{k\to\infty} \|e_k\| = 0$.*

Hence, in this general case, not only will the error reduce from trial to trial, it will also converge monotonically to zero. If GG^* is a strictly positive operator, the convergence is *geometric*.

Proposition 2. *If the operator GG^* satisfies $GG^* \geq \sigma^2 I$ for some $\sigma > 0$, then the following estimate holds:*

$$\|e_{k+1}\| \leq \frac{1}{1+\sigma^2}\|e_k\|, \tag{34}$$

and the norm of the tracking error converges geometrically to zero.

It was also shown in (Amann *et al.*, 1998) and (Amann *et al.*, 1996) that if the plant is a continuous-time system, σ^2 in Proposition 2 is in fact zero, whereas in the discrete-time case it is non-zero. In summary, continuous-time systems result in guaranteed monotonic convergence, whereas discrete-time systems result in guaranteed *geometric* convergence.

The analysis of the sequence $\{u_k\}$ shows that in the discrete-time case the sequence converges in norm to u^*, whereas in the continuous-time case a similar result does not yet exist (as it depends crucially on the reference signal r). However, the convergence to a fixed point u^* can be guaranteed also in the continuous-time case, if the following 'relaxed' algorithm is used:

$$u_{k+1} = \alpha u_k + G^* e_{k+1}, \tag{35}$$

where $\alpha \in (0, 1)$. A simple calculation indicates that the limit solves the linear quadratic optimal tracking problem

$$\min_{u\in U}\{J(u) = \|e\|^2 + (1-\alpha)\|u\|^2\}. \tag{36}$$

However, in this case the error converges to a non-zero vector, but the norm of this vector can be reduced by letting α approach unity.

5.2. Causal implementation. An example using optimal control theory. Despite the generality of the above analysis, it is essential that the formal results be translated into useful, implementable controllers, which use only past data, knowledge of the reference signal r and knowledge of the plant model G. Consider the continuous-time linear time-invariant plant model (possibly a MIMO system)

$$\begin{aligned} \dot{x}_k(t) &= Ax_k(t) + Bu_k(t), \qquad x_k(0) = x_0, \\ y_k(t) &= Cx_k(t), \end{aligned} \tag{37}$$

where $t \in [0, T]$, and without loss of generality assume that $x_0 = 0$. With these assumptions, the input-output behaviour of the dynamical system (37) can be equivalently written as $y(t) = [Gu](t)$, where $G : \mathcal{U} \to \mathcal{Y}$ is defined with the formula

$$[Gu](t) = \int_0^t Ce^{A(t-\tau)}Bu(\tau)\,\mathrm{d}\tau. \tag{38}$$

The inner products are selected to be

$$\langle u_1, u_2\rangle_{\mathcal{U}} = \int_0^{\mathrm{T}} u_1^{\mathrm{T}}(t)Ru_2(t)\,\mathrm{d}t \tag{39}$$

and

$$\langle y_1, y_2\rangle_{\mathcal{Y}} = \int_0^{\mathrm{T}} y_1^{\mathrm{T}}(t)Qy_2(t)\,\mathrm{d}t + y_1^{\mathrm{T}}(T)Fy_2(T), \tag{40}$$

where R, Q and F are symmetric positive definite matrices. These definitions, together with the constraint equation $e(t) = r(t) - [Gu](t)$, result in the following familiar differential equation for optimal u_{k+1}:

$$\begin{aligned} \dot{\psi}_{k+1}(t) &= -A^{\mathrm{T}}\psi_{k+1}(t) - C^{\mathrm{T}}Qe_{k+1}(t), \\ \psi_{k+1}(T) &= C^{\mathrm{T}}Fe_{k+1}(T), \\ u_{k+1}(t) &= u_k(t) + R^{-1}B^{\mathrm{T}}\psi_{k+1}(t). \end{aligned} \tag{41}$$

This equation is anti-causal because of the terminal condition $\psi_{k+1}(T) = C^{\mathrm{T}}Fe_{k+1}(T)$ and thus cannot be used to implement the algorithm. However, using the standard techniques of optimal control theory, a causal implementation of (41) has the form

$$\begin{aligned} u_{k+1}(t) &= u_k(t) - R^{-1}B^{\mathrm{T}}M(t), \\ M(t) &:= \left[K(t)(x_{k+1}(t) - x_k(t)) - \xi_{k+1}(t)\right]. \end{aligned} \tag{42}$$

It can be shown that this implementation is equivalent to (41) if $K(t)$ satisfies the Riccati-equation:

$$\begin{aligned} \dot{K}(t) = &-A^{\mathrm{T}}K(t) - K(t)A \\ &+ K(t)BR^{-1}B^{\mathrm{T}}K(t) - C^{\mathrm{T}}QC, \\ K(T) = &\, C^{\mathrm{T}}FC, \end{aligned} \tag{43}$$

and $\xi_{k+1}(t)$ satisfies the differential equation

$$\begin{aligned} \dot{\xi}_{k+1}(t) = &-(A - BR^{-1}B^{\mathrm{T}}K(t))^{\mathrm{T}}\xi_{k+1}(t) \\ &- C^{\mathrm{T}}Qe_k(t) \end{aligned} \tag{44}$$

which is computable in reverse time as it is driven by the tracking error from the previous trial k. In summary, the algorithm consists of the following steps:

(1) Select suitable values for R and Q and simulate $K(t)$ in reverse time using (43).

(2) Select an initial guess u_0 and feed it into the real plant. Record the input $u_0(\cdot)$, the corresponding tracking error $e_0(\cdot)$, and the state $x_0(\cdot)$, and set the iteration index k equal to one.

(3) Solve the so-called *predictive term* $\xi_k(\cdot)$ in reverse time using (44).

(4) Run a new experiment using the control law (42). Record the input $u_k(\cdot)$, the corresponding tracking error $e_k(\cdot)$, and the state $x_k(\cdot)$. Set $k \to k+1$ and go to Step 3.

Remark 2. It is important to note that in the implementation of the algorithm the Riccati equation for $K(t)$ has to be solved only once, whereas the predictive term $\xi_k(t)$ has to solved between each trial via numerical integration. Furthermore, the implementation requires a full knowledge of the state of the system and, if not available, a state observer is needed.

Remark 3. The inner product (40) might at first sight look slightly exotic. However, the addition of the term $y_1^{\mathrm{T}}Fy_2$ guarantees that the algorithm results in a uniform zero tracking error even if $CB = 0$, i.e., the relative degree of the plant is more than one. See (Amann, 1996; Amann *et al.,* 1998) for details.

A similar causal implementation exists also for linear time-varying discrete-time systems, see (Amann *et al.*, 1996) for details.

5.3. Predictive extensions. So far in this section the standard NOILC algorithm has been introduced and its convergence properties have been analysed. The power of the ideas lies in their great generality, i.e., providing a *guarantee* of monotone convergence to a zero error independent of the details of plant dynamics. Plant dynamics do, however, have an impact on the rate of convergence. For example, the presence of non-minimum-phase zeros is known to introduce initially fast convergence but infinitessimally slow terminal convergence. Consideration must be given to ensuring that the rate of convergence is acceptable for the application, see (Amann, 1996) and (Longman, 2000) for a more detailed discussion on this topic.

In optimal control terms, convergence speeds do increase as the weighting matrix R reduces. However, intuition suggests that faster convergence might be achieved if future tracking errors and future input differences are also added to the cost function. This was proposed by the first author in (Amann *et al.*, 1998) as part of a predictive receding horizon control design where the performance index is replaced by

$$\begin{aligned} &J_{k+1,n}(\vec{u}_{k+1}) \\ &\quad = \sum_{i=1}^{n} \lambda^{i-1}(\|e_{k+1,i}\|^2 + \|u_{k+1,i} - u_{k+1,i-1}\|^2), \end{aligned} \tag{45}$$

where $\vec{u}_{k+1} := [u_{k+1,1}\ u_{k+1,2}\ \ldots\ u_{k+1,n}]$. Here $u_{k+1,j}$ refers to the input for iteration $k+j$ calculated during iteration $k+1$ and $e_{k+1,j}$ refers to the (predicted) tracking error for iteration $k+j$ calculated during iteration $k+1$. Furthermore, in the following material the definition $\vec{e}_{k+1} := [e_{k+1,1}\ e_{k+1,2}\ \ldots\ e_{k+1,n}]$ is used. Finally, u_{k+1} refers to the input control applied to the real plant during iteration $k+1$, and $e_{k+1} = r - Gu_{k+1}$ is the resulting tracking error from this particular choice of the input function.

The proposed criterion (45) includes the error not only from the current trial but also the predicted error from the next $n-1$ trials (n is known as the prediction horizon) as well as the corresponding changes in the input. The weight parameter $\lambda > 0$ determines the importance of more distant (future) errors and incremental inputs. Furthermore, just as in generalized predictive control (Camacho and Bordons, 1998), a receding horizon principle is proposed in (Amann *et al.*, 1998). In this approach, at trial $k+1$ only the input $u_{k+1,1}$ is used as an input into the plant, and the predictive optimization is repeated again at the next trial.

By including more future signals into the performance criterion (45) (i.e., by increasing n), it is argued by the authors that, as n increases, the algorithm should become less 'short sighted', and faster convergence should be obtained when compared to the non-predictive algorithm resulting from the minimisation of (26). This was

rigorously proved in (Amann *et al.*, 1998) when the input $u_{k+1,1}$ was used as the 'true' input. In addition, the resulting algorithm from (45) has a causal implementation if the original plant can be described with a state-space representation. This implementation takes requires a multi-model set-up with $n-1$ plant models running in parallel with the plant.

Although little has yet been derived about the robustness of this algorithm, it is known that, as $n \to \infty$, the error sequence increasingly behaves in a way that satisfies (in the strong operator topology sense) the relation

$$||e_{k+1}|| \leq \frac{1}{\lambda}||e_k||, \qquad \forall k \geq 0, \tag{46}$$

suggesting that the choice of $\lambda > 1$ is crucial and increasing λ may tend to improve convergence rates. As a further recent extension of these ideas, it was noted in (Hätönen and Owens, 2004) that it is possible to use the receding horizon principle for any other input $u_{k+1,j}$ or, more generally, the implemented control input function can be taken as a positive convex combination of the inputs $u_{k+1,j}$. In this appoach the input used for iteration $k+1$ is given by

$$u_{k+1} = \sum_{j=1}^{n} \alpha_j u_{k+1,j}, \tag{47}$$

where $\alpha_j \geq 0$ and $\sum_{j=1}^{n} \alpha_j = 1$. The convex combination approach in fact leads into a quicker convergence rate when compared to the receding horizon approach, see (Hätönen and Owens, 2004) for details.

6. Parameter-optimal ILC

In the previous section the possibility of using optimization techniques in ILC were discussed, and it was shown how the NOILC approach and its predictive modification result in geometric convergence for an arbitrary linear (possibly time-varying) plant. The implementation of the different versions of the algorithm could, however, be a non-trivial task. This is due to the fact that both the feedback and feedforward term are generated by a time-varying non-causal dynamical system, and they have to be solved by using numerical integration. In addition, the feedback term requires a full knowledge of the state of the system, which either requires instrumentation that can measure the states (possibly resulting in an expensive measurement set-up) or the inclusion of a state-observer into the NOILC algorithm. Therefore, it is an important question whether or not there exist structurally (in terms of implementation and instrumentation) simpler algorithms that would still result at least in monotonic convergence, as is discussed in (Owens and Feng, 2003). A natural starting point for answering this question is the Arimoto-type ILC algorithm

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1) \tag{48}$$

because it is a feedforward algorithm (i.e., it does not require any additional instrumentation or other 'real-time components'), and the corrective term is simply a learning gain multiplied with the shifted tracking error from the previous trial. However, if γ is not selected 'optimally', Section 4 showed that the algorithm results only in asymptotic convergence and may exhibit poor transient performance. One possible improvement is to select γ by making it iteration varying. This iteration variation is constructed by solving 'optimal' γ_{k+1} for each iteration using a suitable cost function. Because the norm of the input difference is $\|u_{k+1} - u_k\|^2 = \gamma^2 \|e_{k+1}\|^2$, a parallel cost function to NOILC for this algorithm could be

$$J_{k+1}(\gamma_{k+1}) = \|e_{k+1}\|^2 + w\gamma_{k+1}^2, \tag{49}$$

where $w > 0$ is a weighting parameter introduced to limit the value of γ used. Note that w can be constant or "adaptive". The authors have introduced the form

$$\begin{aligned} &w = w_1 + w_2||e_k||^2, \\ &w_1 \geq 0,\ w_2 \geq 0,\ w_1 + w_2 > 0 \end{aligned} \tag{50}$$

to allow for varying "caution" as the algorithm progresses. The convergence analysis for this algorithm is presented in the following subsection. For the proofs of the propositions, see (Owens and Feng, 2003).

6.1. Convergence analysis of the simple POILC algorithm. As a starting point note that the constraint equation for the cost function (49) can be written as $y_k = G_e u_k$, where G_e is the *lifted* plant model

$$G_e = \begin{bmatrix} C\Delta & 0 & 0 & \dots & 0 \\ C\Phi\Gamma & C\Gamma & 0 & \dots & 0 \\ C\Phi^2\Gamma & C\Phi\Gamma & C\Gamma & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C\Phi^{N-1}\Gamma & C\Phi^{N-2}\Gamma & \dots & \dots & C\Gamma \end{bmatrix}, \tag{51}$$

and

$$\begin{aligned} u_k &:= [u_k(0)\ u_k(1)\ \dots\ u_k(N-1)]^{\mathrm{T}}, \\ y_k &:= [y_k(1)\ y_k(1)\ \dots\ y_k(N)]^{\mathrm{T}} \end{aligned} \tag{52}$$

are the lifted input and output super-vectors. Using this description of the plant model, it is easy to show that the optimal update law for the optimal γ_{k+1} is given by the following equation:

$$\gamma_{k+1} = \frac{e_k^{\mathrm{T}} G_e e_k}{w + e_k^{\mathrm{T}} G_e^{\mathrm{T}} G_e e_k}. \tag{53}$$

The next proposition shows with an optimal update for γ_{k+1} the algorithm results in monotonic convergence.

Proposition 3. *The algorithm (48), where γ_{k+1} is calculated using (53), results in the monotonic convergence*

property $\|e_{k+1}\| \leq \|e_k\|$. *In particular,* $\|e_{k+1}\| = \|e_k\|$ *if and only if either* $e_k = 0$ *or* γ_{k+1}*=0.*

Note that as in the NOILC case, monotonicity again arises from optimality and is independent of the plant model or the weighting parameter w. The next proposition illustrates the behaviour of the sequence of optimal learning gains $\{\gamma_k^*\}$ in the limit.

Proposition 4. *Suppose that* $w_1 > 0$. *The sequence of optimal learning gains* $\{\gamma_k^*\}$ *generated by (53) satisfies* $\lim_{k\to\infty} \gamma_k^* = 0$.

This result shows that the algorithm becomes more cautious as k increases, possibly enhancing the robustness properties of the algorithm.

Unfortunately, neither of the above propositions necessarily imply that the error $e_k \to 0$ as $k \to \infty$. It turns out that convergence to the zero error is dependent on the plant dynamics, this dependence being expressed in terms of positivity conditions, see the following proposition.

Proposition 5. *Consider the algorithm (48) with the update law (53) for* γ_{k+1}. *Suppose that* $G_e + G_e^T$ *is positive-definite; in other words,* $\exists \sigma \in \mathbb{R},\ \sigma > 0$ *so that* $v^T G_e v = v^T(\frac{G_e+G_e^T}{2})v \geq \sigma v^T v$ *for arbitrary* $v \in \mathbb{R}^N$. *Then the resulting error sequence satisfies* $\lim_{k\to\infty} e_k = 0$.

Remark 4. Note that if $w_1 = 0$ in (50), Proposition 5 can be modified to show that the algorithm leads to geometric convergence to a zero tracking error.

Remark 5. Note that in (Hätönen, 2004), it was shown that a sufficient condition for $G_e + G_e^{\mathrm{T}}$ to be positive-definite is that $zG(z) = zC(zI-\Phi)^{-1}\Gamma$ be a positive-real transfer function.

Note that if the matrix $G_e + G_e^{\mathrm{T}}$ is negative-definite, a similar analysis as in Proposition 5 shows that the algorithm will converge monotonically to a zero tracking error. However, if $G_e + G_e^{\mathrm{T}}$ is negative-definite, it can be always made positive-definite by multiplying G_e with -1. Therefore in this section (without loss of generality) the statement '$G_e + G_e^{\mathrm{T}}$ is not a positive-definite matrix' means that $G_e + G_e^{\mathrm{T}}$ is either a semi-definite or a non-definite matrix.

What happens if $G_e^{\mathrm{T}} + G_e$ is not a positive-definite matrix? This is a more complex question, but the following proposition shows that the algorithm can indeed converge to a non-zero solution.

Proposition 6. *Assume that* $G_e^T + G_e$ *is not a positive-definite matrix. Then, for any non-zero point in the limit set* $\{e_\infty : e_\infty^T G_e e_\infty = 0\}$ *initial control inputs* u_0 *exist such that the POILC algorithm (48) converges to* e_∞.

In summary, a zero tracking error in the limit is guaranteed when $G_e + G_e^{\mathrm{T}}$ is a positive-definite matrix, but in other cases the limiting tracking error can be (and can be seen to be) a non-zero vector. A more detailed convergence analysis of this algorithm can be found in (Owens and Feng, 2003) and, more recently, as part of the publication (Owens and Daley, 2008).

Remark 6. Positivity can be regarded as a property of the plant or as an objective of the control design analysis and/or implementation. In this context,

- The plant G may already contain control loops introduced to ensure that positivity conditions are satisfied, see (Hätönen, 2004; Owens and Feng, 2003); or
- A change of the norm to an exponentially weighted norm (Owens and Feng, 2003) may induce positivity at the expense of strict monotonicity of the Euclidean norm, see (Owens and Feng, 2003).

Remark 7. The existence of non-zero limits is established above and is a reality for *any* non-positive system. Further work is being done on the form and properties of these limit sets and their implications for practical POILC implementations. The analysis is complex only because the limit set lies in a high dimensional space (of dimension equal to the number of samples in the time interval of interest). The analysis is needed because most systems, in practice, are non-positive and hence non-zero limits will be the norm rather than the exception.

6.2. Limit sets, dynamics and switching strategies. The recent publication (Owens and Daley, 2008) provided considerable insight into the issue of convergence to non-zero errors in parameter optimal ILC. The essential structure of the results obtained can be summarised (where $M = G$ for consistency with the notation in the reference) as a decomposition of the limit set into two subsets, one S_∞^-, which attracts iterative trajectories, and one S_∞^+, which repels such trajectories.

Theorem 1.

1. *A point* $e \in S_\infty = \{e : e^T M e = 0\}$ *attracts local trajectories of the ILC algorithm if it lies in the attracting component* $S_\infty^- \subset S_\infty$ *defined by the equations*

$$e^T M e = 0, \mathcal{B}(e) = \frac{e^T(M+M^T)Me}{w + ||Me||^2} \in (0,2). \tag{54}$$

2. *If, however, it satisfies* $e^T M e = 0$ *and one of the inequalities*

$$\mathcal{B}(e) > 2 \quad , \quad \mathcal{B}(e) < 0, \tag{55}$$

then it repels local trajectories and is said to lie in the repelling component $S_\infty^+ \subset S_\infty$.

Note: The cases where $e^T Me = 0$ and $\mathcal{B}(e) = 0$ or $\mathcal{B}(e) = 2$ are not considered here. They lie on the boundary between the two sets defining attracting components and repelling components, respectively, and will require further analysis to resolve their characteristics.

In practice, points on the attracting component will attract error iterates leading to convergence to a non-zero limit error. In contrast, if the algorithm leads to an error time series on some iteration that is close to the repelling component, the algorithm will appear to be converging slowly but, in reality, it is simply "pausing" before it is repelled and further monotonic convergence – clearly non-desirable in practice as slow convergence is a waste of resource whether that is time or experimental cost. Examples can be found in the reference (Owens and Daley, 2008) which indicate that even simple non-positive plants can exhibit such behaviour.

Convergence to non-zero limit errors is undesirable unless that limit error is small. A conceptual solution to the problem is presented in (Owens and Daley, 2008) using switching. The essence of the idea is that, using the ILC law

$$u_{k+1} = u_k + \beta_{k+1} K_{k+1} e_k \tag{56}$$

and the POILC optimal parameter defined by

$$\beta_{k+1} = \frac{e_k^T M_{k+1} e_k}{w + ||M_{k+1} e_k||^2}, \quad M_{k+1} = GK_{k+1}, \tag{57}$$

the iteration dependent dynamical systems $\{K_{k+1}\}$ are elements of a well defined, sufficiently rich and numerous set of matrix representations of dynamical "filters". The algorithm is constructed to switch between members of this set in a deterministic manner with the resultant amazing property that the error sequence $\{||e_k||\}_{k\geq 0}$ will converge to zero even though the plant itself is non-positive.

The paper (Owens and Daley, 2008) provides a complete description of such deterministic switching algorithms and illustrates the wide range of successful switching strategies that can be considered. It also presents a statistical analysis of the algorithms that suggest that random switching is beneficial although the nonlinear nature of the problem makes exact theoretical analysis difficult.

6.3. Inverse and adjoint algorithms. So far the POILC approach results in zero tracking error only for some processes, namely, those where the symmetric part of the plant model G_e is a positive definite matrix. Consider now a more general algorithm (in the super-vector notation)

$$u_{k+1} = u_k + \gamma_{k+1} K e_k, \tag{58}$$

where K is an arbitrary matrix (normally representing a dynamical system or a set of such operations). A simple calculation shows that this update law is equivalent to the Arimoto algorithm $u_{k+1} = u_k + \gamma_{k+1} e_k$ when applied to the plant $G_e K$. Hence, if the series connection GK has a positive-definite symmetric component, the results from the previous section immediately imply monotonic convergence to a zero tracking error. The positive-definite condition on GK leads 'naturally' to the following design options:

$$K = \begin{cases} G^{-1} & \text{the "inverse algorithm",} \\ G^{\mathrm{T}} & \text{the "adjoint algorithm".} \end{cases} \tag{59}$$

In both cases, GK satisfies (assuming exact models) the desirable positivity condition noted previously. The convergence results for these algorithms can be found in (Hätönen *et al.*, 2003) and, more generally, in (Harte and Owens, 2005). Both algorithms exhibit theoretically good robustness against modelling uncertainties. Furthermore, both algorithms have been applied successfully to industrial-scale problems, see (Ratcliffe *et al.*, 2004; Daley *et al.*, 2004).

Robustness analysis for these algorithms is possible and was done for the inverse model algorithm in (Harte and Owens, 2005) and for the adjoint algorithm in a forthcoming paper authored by Owens and Daley to appear in the *International Journal of Robust and Nonlinear Control.* These results are based on a rigorous formulation of the frequency domain characteristics of ILC on finite time intervals and a definition of *robust monotonic convergence* that requires retention of the property of monotonic convegrence of the tracking error to zero despite the presence of the modelling error in plant dynamics. To summarize the ideas, suppose that $G(z) = G_0(z)U(z)$, where G_0 is a nominal model and U represents a multiplicative uncertainty. The two algorithms summarised above then become

$$K = \begin{cases} G_0^{-1} & \text{the "approximate inverse algorithm",} \\ G_0^{\mathrm{T}} & \text{the "approximate adjoint algorithm".} \end{cases} \tag{60}$$

The POILC analysis must then assume that $U = 1$ for the purposes of calculating a value of β_{k+1}. The analyses in the two papers mentioned above indicate that the resultant algorithm is robust monotone stable if (necessary and almost sufficient conditions)

$$|\frac{1}{\beta_0} - U(z)| < \frac{1}{\beta_0}, \quad \forall |z| = 1, \tag{61}$$

for the inverse algorithm and

$$|\frac{1}{\beta^*} - U(z)|G(z)|^2| < \frac{1}{\beta^*}, \quad \forall |z| = 1, \tag{62}$$

for the adjoint algorithm. The calculation of the required value of β^* based on the plant model and initial error data is described in the relevant reference.

These robustness results have simple frequency domain interpretations that are discussed in the references.

It is worth noting here that they clearly indicate the need for $U(z)$ to be positive real and for the "gain" to be sufficiently small. They also indicate the sensitivity of the adjoint algorithm to resonance in the nominal model along with the relative robustness of the adjoint algorithm to high frequency dynamics in the uncertainty U. These insights are very valuable but only apply to the two algorithms described. The possibility that better algorithms exist must be considered and the relevant theoretical convergence and robustness conditions established.

6.4. High-order algorithms. In the above, the description has been simplified by the use of simple feedback of the previous trial error e_k on trial $k+1$. Intuitively, if more data are used in POILC, then improved ILC performance should be obtained. With this in mind, a number of publications have investigated the so-called *high-order POILC*, see, for example, (Owens and Feng, 2003; Hätönen, 2004) and, more generally, the work (Hätönen, Owens and Feng, 2006). The idea can be illustrated by the control update law

$$u_{k+1} = \sum_{i=1,M_1} \alpha_{k+1,i} u_{k+1-i} + \sum_{i=1,M_2} \beta_{k+1,i} e_{k+1-i}, \tag{63}$$

where the parameter vectors $\alpha_{k+1} = [\alpha_1, \ldots, \alpha_{M_1}]^T$ and $\beta_{k+1} = [\beta_1, \ldots, \beta_{M_2}]^T$ are selected by minimizing an objective function of a typical form (with W_1 and W_2 positive definite and symmetric),

$$J = ||e_{k+1}||^2 + \alpha_{k+1}^{\mathrm{T}} W_1 \alpha_{k+1} + \beta_{k+1}^{\mathrm{T}} W_2 \beta_{k+1}. \tag{64}$$

These high order ideas are recognized as worthy of a future study but it is already known that their value is ultimately limited by the facts that (1) non-zero limit sets for the error exist in almost all cases when G is non-positive and (2) in the final phases of convergence, errors and inputs are almost co-linear. As a consequence, the benefits of high-order terms are concentrated in the fist few iterations and are ultimately lost.

7. Robustness issues

ILC algorithms are commonly applied in situations that are typified by

(a) imprecise knowledge of a plant model,

(b) non-linearities and parasitic dynamics,

(c) disturbances that change from trial to trial,

(d) imprecise resetting of the initial condition.

Although a number of results have been obtained in each case (see, e.g., Chen and Wen, 1999; Harte and Owens, 2005; Furuta and Yamakita, 1987), it is fair to say that theoretical knowledge and CAD tools relevant to the understanding of these problems are currently very limited. Much more work is needed in this area. The following comments indicate that the possible results for ILC may take specific forms. The observations are qualitative but based on a reasonable mathematical intuition.

Modelling errors: To illustrate the impact of modelling errors, consider the following ILC algorithm:

$$u_{k+1} = u_k + \beta K e_k, \tag{65}$$

which is trivially stable if, and only if, $G_e K$ has a spectrum in the open right-half-plane and β is sufficiently small. Clearly, the choice of K will affect the robustness of the scheme. To illustrate the ideas, suppose that $K = G_o^{-1}$, where G_o is an approximate model and a multiplicative model uncertainty U is defined by $G_e = UG_o$. In this situation, robustness is clearly connected to the spectrum of the matrix representation of the uncertainty and requires that the spectrum is 'sign-definite'. These observations indicate that robustness analysis is more complex than that needed in non-ILC situations (which often uses norm and hence direction insensitive methods) and can explain, to some degree, why progress in producing useful analysis and design tools for attacking plant uncertainty has been slow.

Optimization and the effect of disturbances: Finally, in an optimization framework, robustness with respect to bounded input disturbances is amenable to geometric interpretation. Clearly, convergence to the zero error is not, in general, possible, but a degree of learning can still take place. The basis for believing this is based on the observation that optimization methods generate descent directions for the norm of the error signal. Hence, provided the disturbance does not 'reverse' the descent direction, learning is still possible. Intuition says that the condition for learning is related to the error/disturbance signal-noise ratio. As a consequence, learning is achieved until the error and disturbance have a similar magnitude. Much more work is required to usefully quantify these ideas.

8. Experimental performance

As part of a collaborative programme, a multi-axis test facility was constructed at the University of Southampton to practically test ILC on a wide range of dynamic systems and in an industrial-style application involving the handling of payloads. The apparatus consists of a three-axis gantry robot supported above one end of a 6m long industrial plastic chain conveyor. The completed facility will also include a second robot at the other end of the conveyor and a payload return mechanism. The overall objective is to be able to use iterative learning control to place payloads (tins of peas) onto the conveyor at one end, move them along the conveyor, then remove them at the other end using the second robot. The return mechanism

will continuously cycle the payloads round the system, allowing the investigation of ILC algorithm long-term stability.

Fig. 3. Illustration of the gantry robot.

Each axis of the gantry robot has been modelled individually in both torque control and speed control modes of operation. The axes dynamics have been determined by performing a series of open loop frequency response tests. By performing tests over a wide range of frequencies, Bode plots of gain and phase against a logarithmic frequency scale can be produced. Furthermore, transfer functions were fitted into the Bode plots, resulting in a parametric model for each axis that can be used to build G_e for each of the axis.

Figure 4 shows the MSE of the tracking error multiplied by 1000 as function of iteration round for each axis using the adjoint algorithm from Section 6.2. The y-axes in the figure are in a logarithmic scale so that changes in the tracking error can be observed during later iteration rounds, where it becomes extremely small. From this figure it is clear that the algorithm converges to almost a zero tracking error, and the convergence speed is acceptable for this particular task. Figures 5–7 show $e_{300}(t) = r(t) - y_{300}(t)$ for each axis. These figures demonstrate further the accuracy of the tracking after learning.

9. Conclusions

This paper described the area of Iterative Learning Control (ILC) as a branch of control systems design. It was necessarily highly selective and concentrated on work originating in the Sheffield Group and with collaborators including problem formulation, new issues for design that arise out of the essentially $2D$ nature of the dynamics and control specifications, and the impact of plant dynamics on

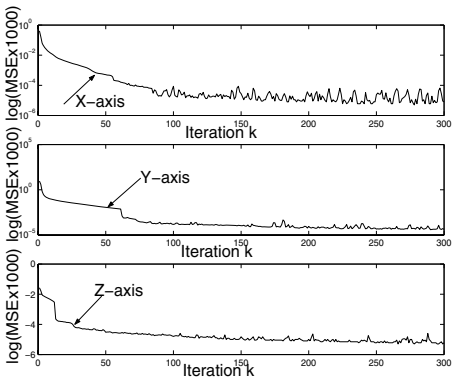

Fig. 4. Log mean squared error$\times 1000$ for each of the axes.

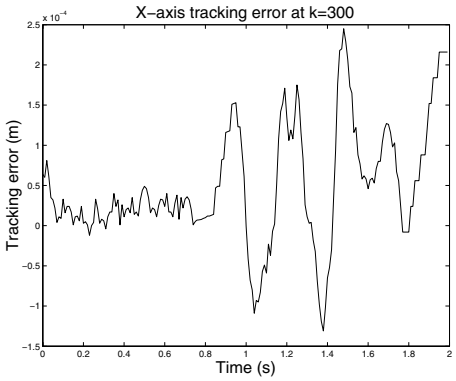

Fig. 5. Tracking error $r(t) - y_{300}(t)$ for the X-axis.

ILC performance. The stability (convergence) of learning dominates the analysis (just as stability dominates classical control design methods), and the need for monotonicity of the Euclidean norms of the signal time series is proposed as a sensible design objective.

The use of optimization methods has been put forward by Owens and co-workers and is seen to be a method of ensuring monotone convergence. It has the advantage that, by choosing quadratic objective functions, it releases classical methods of optimal control and optimization for use in this area. In several important cases (the so-called Norm Optimal ILC (NOILC)), ILC algorithms can be expressed in terms of familiar objects such as Riccatti equations and 'co-state' equations and consist of a combination of off-line (inter-trial) computations (used to form 'feedforward' signals) and state feedback.

ILC, as most control designs, does have to address the issues of the complexity of computation and implementation. One of the interesting facts to emerge from the work of Owens and colleagues is that monotone convergence is retained for simplified Parameter-Optimal ILC (POILC) approaches. These approaches offer the attractive feature of relative simplicity but, being sub-optimal

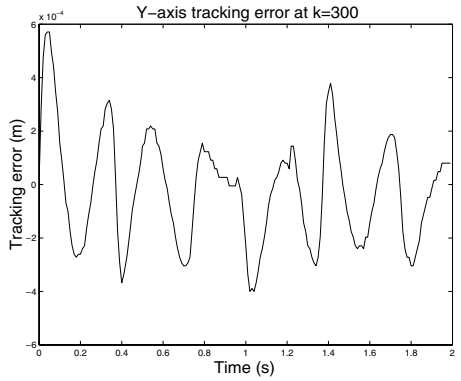

Fig. 6. Tracking error $r(t) - y_{300}(t)$ for the Y-axis.

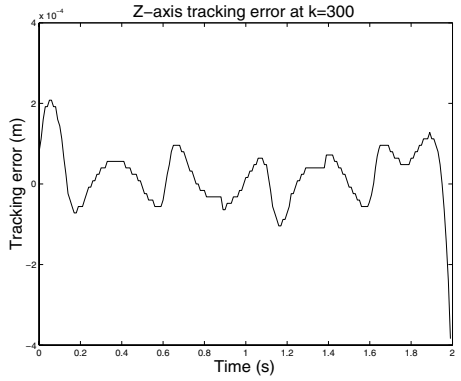

Fig. 7. Tracking error $r(t) - y_{300}(t)$ for the Z-axis.

in the NOILC sense, consideration must be given to the issues of convergence rates and whether or not the error actually does converge to zero. It is known that non-zero limit errors can occur but that a careful choice of the control structure does offer the possibility of either reducing their magnitudes or, under well defined conditions (that depend on plant dynamics), ensuring that $e = 0$ is the *only* possible limit. One proposed solution to this problem has been the use of carefully constructed switching strategies but, more generally, the form and nature of limit sets need to be more fully understood. In addition, the area requires that a more effective robustness theory be developed. Current work at Sheffield has made progress in this area, but much more needs to be done.

Finally, ILC offers theoretical, computational and applications challenges. Some solutions have been obtained and a number of new directions opened up. Practical studies have underlined the potential of these new algorithms and their robustness in practice. There are a multitude of open problems needing attention. This paper has only touched upon some of the issues. Interested readers will find more information in the references and future publications.

Acknowledgements

The authors are indebted to the many students and collaborators who have contributed to the work on ILC and repetitive control. In particular, their long standing collaboration with staff and students at Southampton University (UK) headed by E. Rogers and P. Lewin is appreciated.

References

Amann N. (1996). *Optimal Algorithms for Iterative Learning Control,* Ph. D. thesis, University of Exeter.

Amann N., Owens D.H. and Rogers E. (1996). Iterative learning control for discrete-time systems with exponential rate of convergence, *IEE Proceedings: Control Theory and Applications* **143**: 217–244.

Amann N., Owens D.H. and Rogers E. (1998). Predictive Optimal Iterative Learning Control, *International Journal of Control* **69**(2), 203–226.

Arimoto S., Kawamura S. and Miyazaki F. (1984). Bettering operations of robots by learning, *Journal of Robotic Systems* **1**: 123–140.

Bien Z. and Xu J. (1998). *Iterative Learning Control: Analysis, Integration, and Application*, Kluwer.

Camacho E.F and Bordons C. (1998). *Model Predictive Control*, Springer.

Chen Y. and Wen C. (1999). *Iterative Learning Control: Robustness and Applications*, Springer.

Chen Y. and Moore K.L. (2000). Comments on United States Patent 3,555,252 – Learning control of actuators in control systems, *Proceedings of the 2000 International Conference on Automation, Robotics, and Control*, Singapore.

Cryer B.W., Nawrocki P.E. and Lund R.A. (1976). A road simulation system for heavy duty vehicles, Technical report 760361, Society of Automotive Engineers.

Daley S., Hätönen J. and Owens D.H. (2004). Hydraulic servo system command shaping using iterative learning control, *Proceedings of the Control Conference 2004,* Bath, UK.

de Roover D. (1997). *Motion Control of a Wafer Stage – A Design Approach for Speeding Up IC Production*, Ph. D. thesis, Delft University of Technology.

Edwards J. B. and Owens D.H. (1982). *Analysis and Control of Multipass Processes*, Research Studies Press.

Furuta K. and Yamakita M. (1987). The design of learning control systems for multivariable systems, *Proceedings of the IEEE International Symposium on Intelligent Control*, Philadelphia, PA, USA, pp. 371–376.

Gorinevsky D. M. (1992). Direct learning of feedforward control for manipulator path tracking, *Proceedings of the IEEE International Symposium on Intelligent Control,* Glasgow, UK.

Gunnarsson S. and Norrlöf M. (2001). On the design of ILC algorithms using optimisation, *Automatica* **37**: 2011–2016.

Harte T.J., Hätönen J.J. and Owens D.H. (2005). Discrete-time inverse model-based iterative learning control: Stability, monotonicity and robustness, *International Journal of Control* **78**(8), 577–586.

Hatzikos V., Hätönen J. and Owens D.H. (2004). Genetic algorithms in norm-optimal and non-linear iterative learning control, *International Journal of Control* **77**(2): 188–197.

Hätönen J. (2004). *Issues of Algebra and Optimality in Iterative Learning Control,* Ph. D. thesis, University of Oulu.

Hätönen J. and Owens D.H. (2004). Convex modifications to an iterative learning control law, *Automatica* **40**: 1213–1220.

Hätönen J., Harte T.J., Owens D.H., Ratcliffe J., Lewin P. and Rogers E. (2003). A new robust iterative learning control law for application on a gantry robot, *Proceedings of the 9th IEEE Conference on Emerging Technologies and Factory Automation,* Lisbon, Portugal.

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

Hätönen J.J., Owens D.H. and Feng K. (2006). Basis functions and parameter optimization in high order iterative learning control, *Automatica* **42**(2): 287–294.

Lee J.J. and Lee J.W. (1993). Design of Iterative Learning Controller with VCR servo system, *IEEE Transactions on Consumer Electronics* **39**: 13–24.

Lee K.S., Bang S.H., Yi S. and Yoon S.C. (1996*a*). Iterative learning control of heat up phase for a batch polymerization reactor, *Journal of Process Control* **6**(4): 255–262.

Lee K.S., Kim W.C. and Lee J.H. (1996*b*). Model-based Iterative learning control with quadratic criterion for linear batch processes, *Journal of Control, Automation and Systems Engineering* **3**: 148–157.

Longman R. W. (2000). Iterative Learning Control and Repetitive Control for engineering practise, *International Journal of Control* **73**(10): 930–954.

Moore K.L. (1993). *Iterative Learning Control for Deterministic Systems*, Springer.

Moore K.L. (1998). Multi-loop control approach to designing Iterative Learning Controllers, *Proceedings of the 37th IEEE Conference on Decision and Control*, Tampa, FL, USA.

Norrlöf M. (2000). Iterative Learning Control: Analysis, Design, and Experiments, Ph. D. thesis, Linköping University.

Norrlöf Mikael (2002). An adaptive iterative learning control algorithm with experiments on an industrial robot, *IEEE Transactions on Robotics and Automation* **18**(2): 245–251.

Owens D.H. and Feng K. (2003). Parameter optimization in Iterative learning control, *International Journal of Control* **76**(11): 1059–1069.

Owens D.H., Tomas-Rodriguez M. and Daley S. (2008). Limit sets and switching strategies in parameter optimal iterative learning control, *International Journal of Control* **81**(4): 626–640.

Ratcliffe J., Harte T.J., Hätönen J., Lewin P.Rogers E. and Owens D.H. (2004). Practical implementation of a model inverse optimal iterative learning controller on a gantry robot, *Proceedings of the IFAC Workshop on Periodic Systems (PSYCO 04),* Yokohama, Japan.

Rogers E. and Owens D.H. (1992). *Stability Analysis for Linear Repetitive Processes*, Springer.

Tao K. M., Kosut R. L. and Aral G. (1994). Learning feedforward control, *Proceedings of the American Control Conference*, Baltimore, MD, USA.

Togai M. and O. Yamano (1985). Analysis and design of an optimal learning control scheme for industrial robots: A discrete system approach, *Proceedings 24th IEEE Conference on Decision and Control,* Ft. Lauderdale, FL, USA.

Uchiyama M. (1978). Formation of high speed motion pattern of mechanical arm by trial, *Transactions of the Society of Instrumentation and Control Engineers* **19**(5): 706–712.

Xu J.X. and Tan Y. (2003). *Linear and Nonlinear Iterative Learning Control*, Springer.

Zilouchian A. (1994). An iterative learning control technique for a dual arm robotic system, *Proceedings of the 1994 IEEE International Conference on Robotics and Automation,* San Diego, CA, USA, pp. 1528–1533.

Received: 25 June 2007
Revised: 30 July 2007