BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES
Vol. 56, No. 3, 2008

Invited paper

Robustness of iterative learning control – algorithms with experimental benchmarking

E. ROGERS*

School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK

Abstract. Iterative learning control is a technique especially developed for application to processes which are required to repeat the same operation over a finite duration. The exact sequence of operation is that the task is completed, the process is reset and then the operation is repeated. Applications are widespread among many industries, e.g. a gantry robot which is required to place items on a conveyor under synchronization as part of a food manufacturing process. In effect, iterative learning control exploits the fact that once a single execution of the task is complete then the input control action and output response produced are available to update the control input for the next trial and thereby sequentially improve performance. Moreover, it may be possible to undertake the required computations during the time between completing one trial and the start of the next. This paper gives an overview of some very significant recent progress in this general area, including results from experimental benchmarking, and also some areas for on-going/future research are outlined.

1. Introduction

Iterative Learning Control (ILC) is concerned with the performance of systems that operate in a repetitive manner where the task is to follow some specified trajectory in a specified finite time interval, also known as a pass or a trial in the literature, with high precision. The novel principle behind ILC is to suitably use information from previous trials, often in combination with appropriate current trial information, to select the current trial input to sequentially improve performance from trial-to-trial. In particular, the aim is to improve performance from trial-to-trial in the sense that the tracking error (the difference between the output on a trial and the specified reference trajectory) is sequentially reduced.

The original work in this area is credited to [1] and since then there have been substantial developments in both systems theoretic and applications terms. For an overview of the algorithm development side see, for example, [2, 3] (the second of these references has the added feature of a categorization of what is a very diverse area). Applications areas include robotics, automated manufacturing plants and food processing. For more details, including some those where there is clear potential for significant added benefit from fully developed ILC, one possible source is the survey article [4].

One fundamental systems theoretic problem in ILC is to determine under what conditions will a scheme to converge to zero tracking error. This is an aspect of the general subject area which has seen much work for both linear and nonlinear plant models, including the target of monotonic error convergence (in the trial-to-trial direction). Zero tracking error is, however, almost impossible to achieve due to random and non-repeating disturbances. Moreover, there is often a trade-off to be made between reduction of the trial-to-trial error and the performance achieved along the trials. For example, it is possible to converge trial-to-trial to a limit error which has unacceptable along the trial dynamics, see, for example, [5]. In essence, ILC is a 2D (information propagation in two independent directions) system.

In general, current research and development in the iterative learning community can, as in other areas, be broadly partitioned into starting from either a linear or nonlinear model of the plant dynamics but here we restrict attention to the former where there are still many open problems. This is especially true at the interface between theory and applications. Given the diverse range of algorithms which have been developed over the years, there is a clear need to develop tools and case studies which allow for valid comparison of competing designs. In this paper, we focus on algorithms which have been experimentally benchmarked using facilities especially designed and constructed for this purpose, including some where a robust design is possible. Finally, some areas of ongoing work/future research are briefly discussed. In the next section we give the required background.

2. Background

Since the original work in the mid 1980's [1], the general area of ILC has been the subject of considerable research in terms of the underlying theory (with experimental verification in some cases). Commonly used ILC algorithms construct the input to the plant or process from the input used on the last trial plus an additive increment which is typically a function of the measured output error on the current and/or a finite number of previous trials, where on any trial the error is the difference between the achieved and desired plant outputs. It is

*e-mail: etar@ecs.soton.ac.uk

simply not possible to give even a high level coverage of all the approaches used, and the resulting algorithms, and hence attention is initially restricted to the norm optimal approach which has recently seen experimental verification. Comparative results for two other popular classes of algorithms, and also against a standard three term (or PID) controller, are given in the sections of this paper with deal with robustness.

Suppose that $y_k(t)$ and $u_k(t)$ denote the output and input respectively of the plant on trial k which is of duration T, i.e. $0 \leq t \leq T < \infty$, and has the same value on each trial. Suppose also that $r_d(t)$ is the desired or reference trajectory. Then $e_k(t) = r_d(t) - y_k(t)$ is the current trial error and the objective of constructing a sequence of input functions such that the performance achieved is gradually improving with each successive trial can be refined to a convergence condition on the input and error, i.e.

$$\lim_{k\to\infty} ||e_k|| = 0, \ \lim_{k\to\infty} ||u_k - u_\infty|| = 0$$

Here $||\cdot||$ is a signal norm in a suitably chosen function space with a norm-based topology and u_∞ is termed the learned control.

The state-space model of the plant to be controlled by an ILC scheme is assumed at this stage to be of the following form

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t), \ 0 \leq t \leq T \\ y_k(t) &= Cx_k(t) \end{aligned} \tag{1}$$

where on trial k, $x_k(t)$ is the $n \times 1$ state vector, $y_k(t)$ is the $m \times 1$ output vector, $u_k(t)$ is the $r \times 1$ vector of control inputs, and the trial length $T < \infty$.

The mathematical definition of ILC used here has the following general form.

Definition 1. Consider a dynamic system with input u and output y. Let $\mathcal{Y}$ and $\mathcal{U}$ be the output and input function spaces respectively and let $r_d \in \mathcal{Y}$ be a desired reference trajectory for the system. An ILC algorithm is successful if, and only if, it constructs a sequence of control inputs $\{u_k\}_{k\geq 0}$ which, when applied to the system (under identical experimental conditions), produces an output sequence $\{y_k\}_{k\geq 0}$ with the following properties of convergent learning

$$\lim_{k\to\infty} y_k = r_d, \ \lim_{k\to\infty} u_k = u_\infty$$

Here convergence is interpreted in terms of the topologies assumed in $\mathcal{Y}$ and $\mathcal{U}$ respectively.

The dynamics of processes described by (1) can be represented in operator form as

$$y = Gu + z_0$$

where $G : \mathcal{U} \to \mathcal{Y}$ is the system input/output operator (assumed to be bounded) and z_0 represents the effects of system initial conditions. If $r_d \in \mathcal{Y}$ is the reference trajectory, or desired output, then the tracking error is defined as

$$e = r_d - y = r_d - Gu - z_0 = (r_d - z_0) - Gu.$$

Hence without loss of generality, it is possible to replace r_d by $r_d - z_0$ and consequently assume that $z_0 = 0$.

It is clear that an ILC algorithm, if convergent, solves the problem $r_d = Gu_\infty$ for u_∞. If G is invertible, then the formal solution is just $u_\infty = G^{-1}r_d$. A basic assumption of the ILC paradigm is that direct inversion of G is not acceptable since, for example, this would require exact knowledge of the plant and involve derivatives of the reference trajectory. This high-frequency gain characteristic would make the approach sensitive to noise and other disturbances. Also inversion of the whole plant G is unnecessary as the solution only requires finding the pre-image of the reference trajectory r_d under G.

The problem can easily be seen to be equivalent to finding the minimizing input u_∞ for the optimization problem

$$\min_u \{\|e\|^2 : \quad e = r_d - y, \quad y = Gu\}$$

where $||\cdot||$ denotes any appropriate norm on $\mathcal{Y}$. The improved approach considered here results in an algorithm with the following two important properties: (a) automatic choice of step size, and (b) potential for improved robustness through the use of causal feedback of current trial data and feedforward of data from previous trials.

More precisely, the algorithm here, on completion of trial k, calculates the control input on trial $k+1$ as the solution of the minimum norm optimization problem

$$\begin{aligned} u_{k+1} = \arg\min_{u_{k+1}} \{&J_{k+1}(u_{k+1}) : \\ &e_{k+1} = r_d - y_{k+1}, \quad y_{k+1} = Gu_{k+1}\} \end{aligned}$$

where the performance index, or optimality criterion, used is defined to be

$$J_{k+1}(u_{k+1}) = \|e_{k+1}\|^2_{\mathcal{Y}} + \|u_{k+1} - u_k\|^2_{\mathcal{U}}.$$

The initial control $u_0 \in \mathcal{U}$ can be arbitrary in theory but, in practice, will be a good first guess at the solution of the problem.

This problem can be interpreted as the determination of the control input on trial $k+1$ with the properties that: (i) the tracking error is reduced in an optimal way; and (ii) this new control input does not deviate too much from the control input used on trial k.

The benefits of this approach are immediate from the simple interlacing result

$$\|e_{k+1}\|^2 \leq J_{k+1}(u_{k+1}) \leq \|e_k\|^2, \qquad \forall k \geq 0 \tag{2}$$

which follows from optimality and the fact that the (non-optimal) choice of $u_{k+1} = u_k$ would lead to the relation $J_{k+1}(u_k) = \|e_k\|^2$. The result states that the algorithm is of the descent class as the norm of the error is monotonically non-increasing in k. Also, equality holds if, and only if, $u_{k+1} = u_k$, i.e. when the algorithm has converged and no more input-updating takes place. There is an implicit choice of step size here which means that, unlike the steepest descent methods, this parameter does not have to be selected by the user.

In this work, we consider the following cost function for processes described by (1) which is clearly a special case of the general form given above.

$$J_{k+1}(u_{k+1}) = \frac{1}{2}\sum_{t=0}^{N}\left\{e_{k+1}^T(t)\,Qe_{k+1}(t) + H^T RH\right\} \qquad (3)$$
$$H \equiv H(u,k) = u_{k+1}(t) - u_k(t)$$

where the weighting matrices Q and R are of compatible dimensions, and symmetric positive semi definite and positive definite respectively. This is the linear model ILC version of the familiar linear quadratic performance criterion from optimal control theory which is, in effect, a combination of the optimal tracking (of $r_d(t)$) and the disturbance accommodation problem (regarding $u_k(t)$ as a known disturbance on trial $k+1$).

Following [6], the solution of the optimal ILC control problem for processes described by (1) with cost function (3) is as follows:

- Matrix gain (Riccati) equation

$$K(t) = A^T K(t+1)A + C^T Q(t+1)C$$
$$-[A^T K(t+1)B \times \left\{B^T K(t+1)B + R(t+1)\right\}^{-1} \qquad (4)$$
$$\times B^T K(t+1)A]$$

where $K(t)$ is a matrix gain which has the terminal condition $K(N) = 0$.

- Predictive component equation

$$\xi_{k+1}(t) = \left\{I + K(t)BR^{-1}(t)B^T\right\}^{-1} \qquad (5)$$
$$\left\{A^T\xi_{k+1}(t+1) + C^T Q(t+1)e_k(t+1)\right\}$$

where $\xi_{k+1}(N) = 0$.

- Input update equation

$$u_{k+1}(t) = u_k(t) - \left[\left\{B^T K(t)B + R(t)\right\}^{-1} B^T K(t)\right. \qquad (6)$$
$$\left.\times A\left\{x_{k+1}(t) - x_k(t)\right\}\right] + R^{-1}(t)B^T\xi_{k+1}(t)$$

3. Measuring ILC performance

In norm-optimal ILC the weighting matrices Q and R can be used to adjust the balance between convergence speed and robustness respectively. A critical task therefore is to investigate just how much the choice of the entries in Q and R affect algorithm performance (in many optimal control applications these are chosen to be diagonal matrices). Therefore it is first necessary to discuss exactly what ILC performance is and how to measure it.

Note: In this work we only consider diagonal Q and R and, in particular, the case when $Q = qI$ and $R = rI$, where $q \geq 0$ and $r > 0$ are real scalars and I is the identity matrix with compatible dimensions in each case.

It is generally recognized that there are three variables which are of particular importance when describing the performance of an ILC algorithm [7]. These are as follows.

- Convergence speed.
- Minimum tracking error.
- Long-term stability.

Although the instantaneous data recorded during each trial is useful for analyzing the learning process and its stability, it is clearly necessary to calculate some general measure of the tracking accuracy for each trial, and observe how these change as the trials progress. This can specifically indicate minimum error, time to reach minimum error (convergence speed) and any sign of instability. Popular measures of tracking accuracy are the error norm or the mean-squared-error (mse).

Figure 1 shows the typical mse plot for an unstable ILC system, where key parameters used to describe performance are also indicated. In particular, e_1 is the initial mse value, i_{me} is the number of trials required to reach minimum error, $e_{\min}$ is the minimum mse value and i_u is the number of trials before the mse begins to increase and effectively the system becomes unstable. A typical mse plot for a stable system is similar, except that the i_u point is never reached and the mse does not increase as the trials progress. Of these parameters, i_{me} and $e_{\min}$ are most commonly used to describe ILC performance.

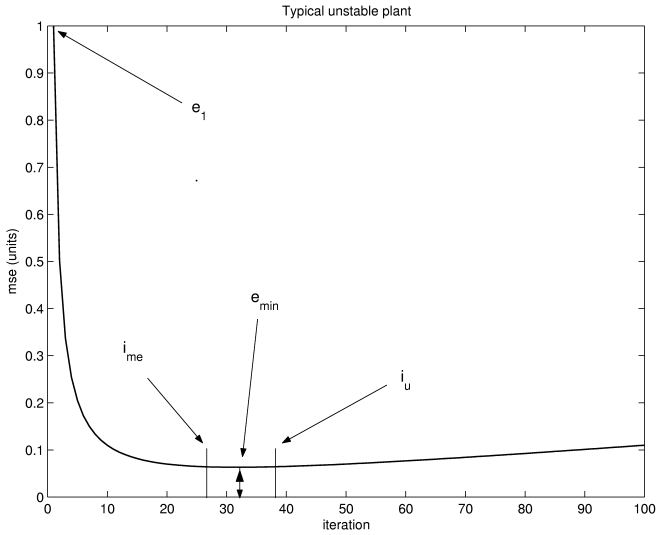

Fig. 1. Typical mse curve for an unstable ILC system

The performance index here simply involves calculating the area under the mse curve for the first N trials, where N is selected appropriately for the system being considered. This results in the performance index for N trials, PI_N. It has been suggested in the literatue that most ILC systems exhibit learning behavior during the first 100 trials. Within this time, the majority of the learning has been achieved and the system has reached, or is near, the minimum tracking error value. Therefore it is appropriate to compare the performance of algorithms during this period. Consequently the experimental results considered here will be compared using PI_{100}.

PI_N is simple to calculate. In particular, divide the area beneath the mse plot into rectangular columns, rather than trapezoids. Then if the width of each trial column is taken as unity, this makes the PI_{100} a simple summation of the first 100 mse values. For the general case PI_N, this is formally defined as:

$$PI_N = \sum_{k=1}^{N} \overline{e^2}_k. \tag{7}$$

These are as follows.

To allow a fair comparison of algorithm performance, several test parameters must be held constant:

- The plant (or plant model in simulation).
- The reference trajectory.
- The value of N.
- The mse value for the first trial (e_1).

Variation in any of these parameters will affect PI_N in a way which does not correspond directly to a change in performance. In which respect, a difficulty many arise with e_1 but note that it can definitely be satisfied if the plant input is set to zero for the duration of the first trial. The plant output should therefore remain constant, and the value of e_1 will be the mse of the reference trajectory. If e_1 held constant, it is logical to remove the PI_N dependency on the unit of mse, by normalizing the mse so that $e_1 = 1$.

It is now possible to define upper and lower bounds on the value of PI_N by considering two extreme cases of tracking performance. Firstly, suppose that the algorithm achieves perfect tracking after only one trial. As specified previously, the mse for the first trial is normalized to 1, but by the second the mse will be 0. Irrespective of the value of N, the mse will remain equal to 0 for all $N > 1$. Therefore the sum of the mses will result in $PI_N(\min) = 1$ which defines the lower bound. Now consider the opposite case when the algorithm learns nothing for any trial. In this case the mse will be equal to 1 for each trial and therefore the upper bound can be defined as $PI_N(\max) = N$. If the algorithm becomes unstable and the calculated PI_N is larger than N, then it is set to N by default. The PI_N value will therefore lie between the bounds $1 \leq PI_N \leq N$. The closer the value of PI_N is to 1, the better the tracking performance.

4. Gantry robot test facility

The gantry robot (Fig. 2) is a commercially available system found in several industrial applications. The robot is located above one end of a plastic chain conveyor, and is tasked with collecting payloads from a dispenser and placing them onto the moving conveyor beneath. This is a fairly involved task, as the robot must synchronize both speed and position with the conveyor to achieve accurate placement of the payload. The gantry robot can be treated as three separate single-input single-output (SISO) systems (one for each axis) which can operate simultaneously to locate the end effector anywhere within a cuboid work envelope. The lowest axis – X moves in the horizontal plane, parallel to the conveyor beneath. The Y-axis is mounted on the X-axis and moves in the horizontal plane, but perpendicular to the conveyor. The Z-axis is the shorter vertical axis mounted on the Y-axis. The X and Y-axes consist of linear brushless dc motors, while the Z-axis is a linear ball-screw stage powered by a rotary brushless dc motor. All motors are energized by performance matched dc amplifiers. Axis position is measured by means of linear or rotary optical incremental encoders as appropriate. In this paper, the conveyor beneath the gantry robot is not considered.

To implement ILC, is it necessary to obtain a model for the plant which is to be controlled. Each axis of the gantry was modelled independently by means of sinusoidal frequency response tests. From this data it was possible to construct Bode plots for each axis and hence approximate transfer-functions determined. These were then refined by means of a least mean square optimization technique, to minimize the difference between the frequency response of the real plant and that of the model. The resulting Bode plots comparing the plant and the model are given in Fig. 3 for the X-axis only with the others in [8]. Using the Bode asymptotic gain plot it is now a routine task to construct an approximate transfer function for the dynamics and hence a minimal state-space model. The transfer-function used in this work is 7th order. (See also [9] which describes a reformulation of the algorithm to give an efficient implementation in terms of the computational load etc.)

Fig. 2. The gantry robot

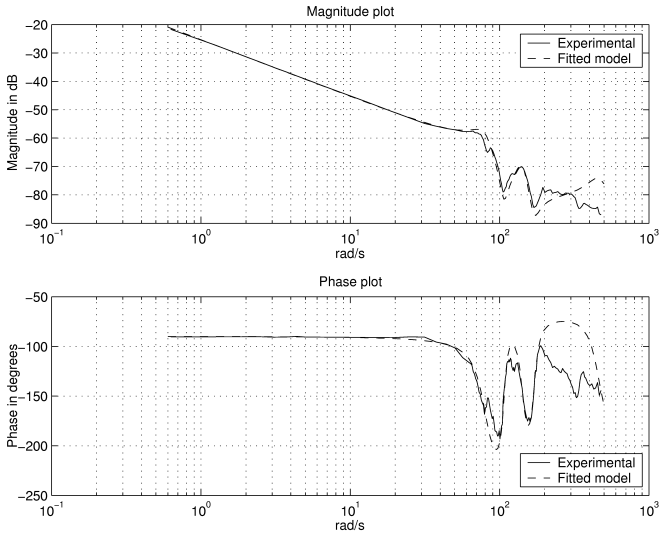

Fig. 3. X-axis Bode plot

5. Test parameters

With all axes operating simultaneously, the reference trajectories for the axes produce a three dimensional synchroniz-

ing 'pick and place' action (Fig. 4). The trajectories produce a work rate of 30 units per minute which is equivalent to a trial duration of 2 seconds. Using a sampling frequency of 1 kHz, this generates 2000 samples per trial.

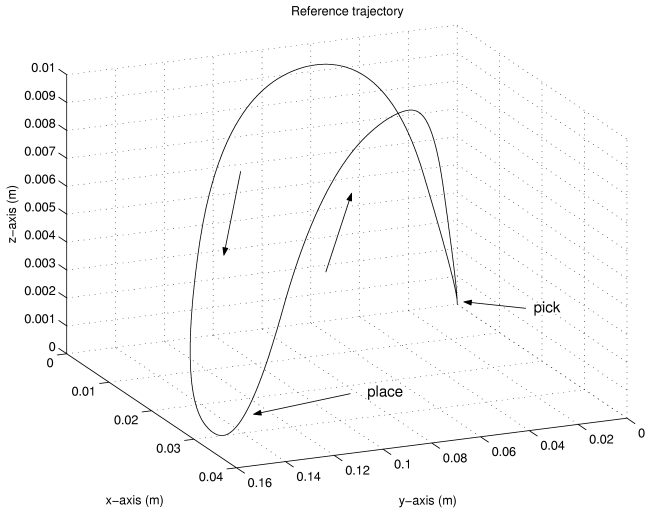

Fig. 4. Three dimensional reference trajectory

All tests were performed in ILC format (this experimental facility can also be configured [8, 9] to operate in a Repetitive Control (RC) mode and therefore also provides a basis for establishing the links between, and the relative performance of, both classes of algorithms) and hence the following hold.

- There is a stoppage time between trial.
- The plant is reset to known initial states before the start of the next trial.
- Calculation of the next ILC plant input occurs between the end of one trial and the start of the next.

A two second stoppage time exists between each trial, during which the next input to the plant is calculated. The stoppage time also allows vibrations induced on the previous trial to die away and prevents vibrations from being propagated between trials. Before each trial begins, the axes are homed to within ± 30 microns of a known starting location to minimize the effects of initial state error.

The plant input signal (voltage) for the first trial is zero. Therefore the algorithm must learn to track the reference in its entirety. There is no assistance from any other form of controller. In the practical implementation, the system states are estimated by means of a tuned full-state observer.

6. NOILC – experimental results

An extensive programme of experiments was undertaken and the results given in this section are from [9]. These confirm the excellent performance in terms of convergence speed, minimum error and long term performance (number of trials which can be completed without (visible) degradation in performance), also termed long term stability in the literature and adopted here from this point onwards.

As representative results from the experiments performed, Fig. 5 shows the mse calculated for each axis during a 5000 trial test designed to investigate the long-term stability. The most important feature here is that there is no indication of an increasing mse which typically indicates that the algorithm is diverging away from the minimum error value and is unstable. The lack of increasing mse strongly suggests that the algorithm is stable. It is important to note, however, that the 5000 trial test does not guarantee infinite trial performance. However, it is a good indicator that the algorithm can achieve long-term performance compared to other algorithms implemented on the same plant which became unstable within 100 or in severe cases just 3 trials.

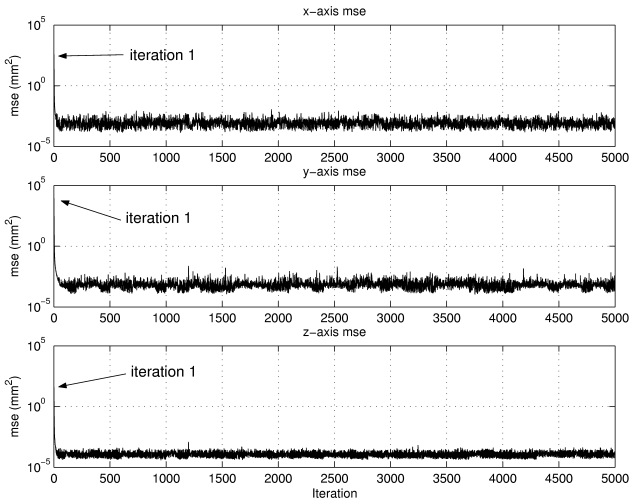

Fig. 5. Implementation – mse 5000 trials

As noted previously, the results here are for diagonal choices of the cost function weighting matrices Q and R with common diagonal entries $q \geq 0$ and $r > 0$. To investigate the effect of varying q and r, a batch of tests was performed using different combinations of these parameters. Table 1 displays the values of q and q which were used to produce a total of 56 combinations. Each combination was implemented for 100 trials and the PI_{100} performance index described in Section 3 was calculated. Given that there are two tuning parameters, it is particularly suitable to plot the algorithm performance on a three dimensional surface chart. Figure 6 displays the performance plot for the X axis. The other two axes performance plots are very similar, particularly the Y axis where the low frequency gain of the linear motor is practically identical to that of the X axis.

Noting that q affects the rate of error reduction and r limits the input change, interpreting the plots becomes a simple task. To the right of the chart is a region of poor tracking performance where the PI_{100} value is near to or equal to 100 indicating that virtually nothing is learnt during the 100 trial test. As could be expected, this corresponds to a small value for q and a large value for r. With these settings, the algorithm is far too conservative. As the ratio of q to r increases, gradually PI_{100} reduces, indicating that the performance is improving. This is represented by the slope to

the right side of the chart. As the q/r ratio continues to increase, PI_{100} is reduced to values very close to 1, indicating that the perfect trajectory is learnt in almost one trial. The balance of error reduction to input change is now approaching optimality. Temporarily increasing q/r has little effect on the performance, until the system becomes unstable and PI_{100} jumps back to 100. This is represented by the channel and then the steep slope to the left of the chart. It is important to note that the ratio of q to r is what determines the algorithm performance rather than the actual values for each parameter. If a larger range of q and r values were used, the chart would still have a channel cutting diagonally across it.

Table 1

q and r values used in experiments

q	r
0.1	0.0001
1	0.001
10	0.01
100	0.1
1000	1
10000	10
100000	100

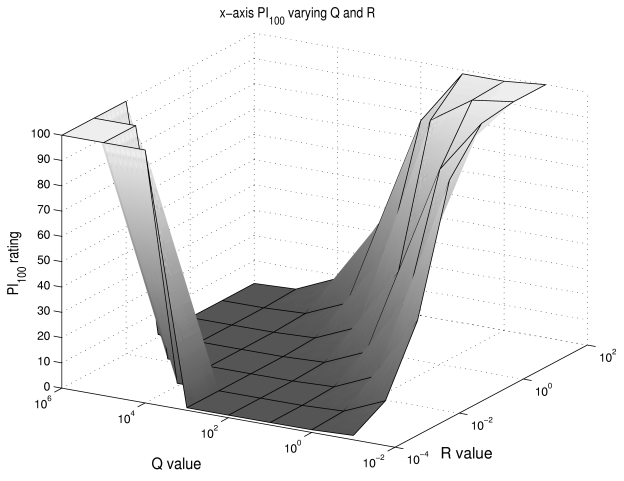

Fig. 6. X-axis PI_{100} for various q and r

Note 1. In this we have assumed that the weighting on each state is the same motivated by the fact that the low frequency responses (Fig. 3 for the X-axis) of the three axes are similar. Non-equal weighting can be examined in the same way.

7. Robust ILC analysis

In this section we address, using results from [10], the relatively open area of robust ILC starting form an assumed plant model in the form (after sampling if required) of the following standard linear, time-invariant single input, single output state-space representation defined over the finite time interval, $0 \le t \le T$ (for ease of notation explicit reference to sampling interval is not given)

$$\begin{aligned} x(t+1) &= \Phi x(t) + \Gamma u(t); \qquad x(0) = x_o \\ y(t) &= Cx(t) \end{aligned} \tag{8}$$

where the state $x(\cdot) \in \mathbb{R}^n$, output $y(\cdot) \in \mathbb{R}$, input $u(\cdot) \in \mathbb{R}$, and Φ, Γ and C are matrices of appropriate dimensions. Without loss of generality, it will be assumed that $C\Gamma > 0$ and that the system (8) is controllable and observable.

Given that the system (8) is defined over a finite time-interval, it can be equivalently described by the matrix equation $\tilde{y}_k = \tilde{G}_e \tilde{u}_k$, with

$$\tilde{G}_e = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ C\Gamma & 0 & 0 & \dots & 0 \\ C\Phi\Gamma & C\Gamma & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C\Phi^{T-1}\Gamma & C\Phi^{T-2}\Gamma & C\Phi^{T-3}\Gamma & \dots & 0 \end{bmatrix} \tag{9}$$

where $\tilde{u}_k = [u_k(0)\ u_k(1)\ \dots\ u_k(T-1)]^T$, $\tilde{y}_k = [y_k(1)\ y_k(2)\ \dots\ y_k(T)]^T$ and the elements $C\Phi^j\Gamma$ of the matrix G_e are the Markov parameters of the plant (8). Suppose also that reference signal $r(t)$ satisfies $r(0) = Cx_o$. Then for analysis it is sufficient to analyze the "lifted" plant description $y_k = G_e u_k$ where $u_k = [u_k(0)\ u_k(1)\ \dots\ u_k(T-1)]^T$, $y_k = [y_k(1)\ y_k(2)\ \dots\ y_k(T)]^T$ and

$$G_e = \begin{bmatrix} C\Gamma & 0 & 0 & \dots & 0 \\ C\Phi\Gamma & C\Gamma & 0 & \dots & 0 \\ C\Phi^2\Gamma & C\Phi\Gamma & C\Gamma & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C\Phi^{T-1}\Gamma & C\Phi^{T-2}\Gamma & C\Phi^{T-3}\Gamma & \dots & C\Gamma \end{bmatrix}. \tag{10}$$

One class of ILC algorithms which has been extensively studied is those based on the steepest descent principle and hence on minimizing a cost function. Here we first consider the following version (see, for example, [11]) where the cost function to minimized for each trial is given by

$$J(u_{k+1}) = \|e_{k+1}\|^2, \ e_{k+1} = r - G_e u_{k+1}. \tag{11}$$

Assume now that the control input on trial $k+1$ is taken as $u_{k+1} = u_k + \epsilon\delta_{k+1}$, where ϵ_{k+1} is a scaling factor and δ_{k+1} determines the direction of the update (vector). Then

$$\begin{aligned} J(u_{k+1}) &= J(u_k + \epsilon_{k+1}\delta_{k+1}) \\ &= \|e_{k+1}\|^2 = \\ &\|e_k\|^2 - 2\epsilon_{k+1}\delta_{k+1}^T G^T e_k + \epsilon_{k+1}^2 \delta_{k+1}^T G_e^T G_e \delta_{k+1} \end{aligned} \tag{12}$$

and hence

$$\|e_{k+1}\|^2 - \|e_k\|^2 = -2\epsilon_{k+1}\delta_{k+1}^T G_e^T e_k + \epsilon_{k+1}^2 \delta_{k+1}^T G_e^T G_e \delta_{k+1}. \tag{13}$$

Consequently to achieve monotonic error convergence, the right-hand side in (13) must be negative. One way to achieve this is to set $\delta_{k+1} = G_e^T e_k$, resulting in the control law

$$u_{k+1} = u_k + \epsilon_{k+1} G_e^T e_k \tag{14}$$

and then

$$\|e_{k+1}\|^2 - \|e_k\|^2 = -2\epsilon_{k+1}\|G_e^T e_k\|^2 + \epsilon_{k+1}^2 \|G_e G_e^T e_k\|^2. \tag{15}$$

The negative term $-\epsilon_{k+1}\|G_e^T e_k\|^2$ is of $o(\epsilon)$ and the positive term $\epsilon_{k+1}^2\|G_e G_e^T e_k\|^2$ is of $o(\epsilon^2)$. Hence, by using a sufficiently small positive ϵ_{k+1}, the right-hand side of (15) can be made negative (note that G_e is invertible by the assumption that $C\Gamma > 0$), resulting in monotonic convergence. In order to automate the selection process for ϵ_{k+1}, it was proposed in [11] that ϵ_{k+1} be determined as the solution of the optimization problem

$$\epsilon_{k+1}^* = \arg \min_{\epsilon_{k+1}\in\mathbb{R}} J(u_k + \epsilon G_e^T e_k). \tag{16}$$

Hence if ϵ_{k+1}^* denotes the optimal value then it is routine to show that

$$\epsilon_{k+1}^* = \frac{\|G_e^T e_k\|^2}{\|G_e G_e^T e_k\|^2}. \tag{17}$$

Then

$$\|e_{k+1}\|^2 - \|e_k\|^2 = -\frac{\|G_e^T e_k\|^4}{\|G_e G_e^T e_k\|^2} \tag{18}$$

where the right-hand side is obviously negative, implying monotonic convergence to zero tracking error.

The problem of what is meant by "robustness" of an ILC control scheme is more involved than in the 1D case. For example, there is robustness to unmodelled dynamics and also to the effects of the process not resetting to the same value before the start of each new trial (this is usually known as robustness to initial conditions). In this section we give new results on robustness to unmodelled dynamics for the a steepest-descent ILC law discussed above and give results from implementation on the gantry robot facility.

Robust control analysis is based on an assumed model for the uncertainty where here a multiplicative approach is used which is modelled by the equation $G_e = G_o U$. Here G_o is the nominal model (i.e. an estimate of the true plant), and U reflects the multiplicative uncertainty (i.e. modelling errors), and also G_o is used instead of G_e in the update law, i.e. $u_{k+1} = u_k + \epsilon_{k+1} G_o^T e_k$. The same cost function as the steepest descent algorithm is considered.

Routine modification of (15) now gives the following result.

Lemma 1. Suppose that $U + U^T$ is a positive-definite matrix. Then if $\|e_k\| \neq 0$ there exists an $\epsilon_{k+1} > 0$ such that $\|e_{k+1}\|^2 - \|e_k\|^2 < 0$.

In the steepest-descent algorithm, ϵ_{k+1} is given by

$$\epsilon_{k+1} = \frac{\|G_o^T e_k\|^2}{\|G_o G_o^T e_k\|^2} \tag{19}$$

and hence there is no clear mechanism to modify this parameter to satisfy Lemma 1. Consequently we now develop a new modified steepest-descent version that will result in monotonic convergence for plants with multiplicative uncertainty U, where $U + U^T$ is assumed to be a positive-definite matrix. The exact form is

$$u_{k+1} = u_k + \epsilon_{k+1} G_e^T e_k \tag{20}$$

where ϵ_{k+1} is selected to be the solution of the optimization problem

$$\min_{\epsilon_{k+1}\in\mathbb{R}} J(\epsilon_{k+1}), \quad J(\epsilon_{k+1}) := \|e_{k+1}\|^2 + w\epsilon_{k+1}^2 \tag{21}$$

and $w \in \mathbb{R}$, $w > 0$.

The cost function $J(\epsilon_{k+1})$ here addresses two design objectives. In particular, the first term aims to keep the tracking error small during each trial and the second aims to keep the magnitude of ϵ_{k+1} small. If successful this will lead to a cautious and robust algorithm in comparison to the basic steepest-descent algorithm. The solution of the optimization problem (21) is (by routine analysis)

$$\epsilon_{k+1} = \frac{\|G_e^T e_k\|^2}{w + \|G_e G_e^T e_k\|^2} \tag{22}$$

and the following result gives its convergence properties.

Lemma 2. If $w \in \mathbb{R}$, $w > 0$ then $\|e_{k+1}\| \leq \|e_k\|$, where equality holds if, and only if, $\epsilon_{k+1} = 0$. Also

$$\lim_{k\to\infty} \|e_k\| = 0 \quad \text{and} \quad \lim_{k\to\infty} \epsilon_k = 0 \tag{23}$$

which establishes monotonic convergence to zero tracking error.

Now focus on the implications of replacing G_e by the assumed uncertainty model, i.e. $G_e = G_o U$. Then using the optimal ϵ_{k+1} we have that monotonic convergence requires $\|e_{k+1}\|^2 - \|e_k\|^2 < 0$ for a non-zero e_k. The next result shows how this can be achieved by taking w to be a sufficiently large positive number.

Lemma 3. Assume that $U + U^T$ is symmetric positive-definite and w is chosen such that

$$w > \frac{1}{2} \frac{\|G_o^T\|^2 \|G_e G_o^T\|^2 \|e_0\|^2}{\sigma_{\min}\left(G_o \dfrac{(U+U^T)}{2} G_o^T\right)} \tag{24}$$

where $\sigma_{\min}\left(G_o\left(\dfrac{U+U^T}{2}\right)G_o^T\right)$ denotes the smallest eigenvalue of the symmetric positive-definite matrix $G_o\left(\dfrac{U+U^T}{2}\right)G_o^T$. In this case, the sequence of tracking errors satisfies $\|e_{k+1}\| < \|e_k\|$ when $e_k \neq 0$.

Note 2. Note that the estimate for w can be very conservative, because the term $e_k^T G_o\left(\dfrac{U+U^T}{2}\right)G_o^T e_k$ is estimated in terms of the smallest eigenvalue of $G_o\left(\dfrac{U+U^T}{2}\right)G_o^T$. Furthermore, if w is selected to be excessively large in magnitude, this can have a undesirable effect on the convergence speed, because a large w will result in a small ϵ_{k+1}, implying that $u_{k+1} \approx u_k$ in such a case. Consequently this proposition should be understood as an existence result, and in practice w can be selected by resorting to a trial and improvement approach.

Note 3. Clearly the sufficient value of w decreases as $\|e_k\|^2$ decreases. This provides a means to reduce w with each successive trial k where such a w could result in an increase in

the convergence speed. A natural choice to exploit this would be $w = w_1\|e_k\|^2$, but, as $\|e_k\|^2$ approaches zero dangerously high inputs could be applied to the plant. A simple remedy is to choose $w = w_0 + w_1\|e_k\|^2$.

The next result shows that in addition to monotonic convergence, $U^T + U > 0$ also implies that $\lim_{k\to\infty} e_k = 0$ if w is selected to be sufficiently large.

Lemma 4. Under the assumptions of Lemma 3 the algorithm converges monotonically to zero tracking error.

Note 4. So far it has been established that for monotonic convergence to zero tracking error it is required that the multiplicative uncertainty U has to be positive in the sense that $U^T + U$ is a positive-definite matrix. However, in most applications the nominal 'matrix' model G_o is obtained by truncating the 'lifted' transfer function $zG_o(z)$, where $G_o(z)$ is the transfer function of the nominal plant model when the time-axis is infinite. Hence it is essential to express this property in terms of these transfer-function models. In which context, note that using the transfer-function descriptions of the plant, the uncertainty model becomes $zG(z) = U(z)G_o(z)$, or $U(z) = G(z)^{-1}G_o(z)$. Also, it can be shown that if $U(z)$ is a positive-real system (or equivalently, its Nyquist diagram lies strictly in the right-half plane, see [12]), the truncated system (matrix) U is positive, i.e. $U^T + U$ is a positive-definite matrix, see [13]. The Nyquist diagram condition is equivalent to the condition that the phase of the uncertainty model $U(z)$ has to lie inside $\pm 90^o$, demonstrating a good degree of robustness.

In summary, the modified steepest-descent algorithm will converge monotonically to zero tracking error if the multiplicative uncertainty U satisfies the positivity condition $U^T + U$ and w is chosen to be sufficiently large. Note that in the standard algorithm $w = 0$. Therefore the introduction of w has resulted in a straightforward mechanism to find a balance between convergence speed and robustness.

8. Further experimental results

The size of the G_o matrices is determined by the sampling frequency and the time period of one trial. For the robust optimal algorithm, the selected sampling frequency is low at only 100 Hz and this has several advantages. A low sample frequency minimizes the sizes of the G_o matrices implying that less memory and computation time are required to generate the next input to the plant. Using a low sample frequency also acts as a simple low-pass filter by aliasing out high frequency noise. This is useful for achieving good long term performance in ILC systems [14] (see also the results section). The gantry robot axes are positioned using a velocity control mode where the linear motor amplifiers operate with their own closed-loop control. The input to the amplifiers is therefore simply a setpoint adjustment and is not directly required to achieve a stable system. Therefore the chosen frequency of 100 Hz is sufficient to accurately control the plant.

As a first set of experiments, the performance of the modified steepest descent algorithm developed in Section 7 has been compared against two other algorithms in the absence of any uncertainty in the plant model, i.e. $G_e = G_o$. The reason for this is to establish that this new algorithm is capable of providing improved performance and the results can be used to assess robustness issues using the modified robust optimal algorithm. The two other algorithms used in this set of experiments are a standard three term (PID) feedback controller (acting on the current trial error alone with K_p, K_i and K_d denoting the proportional, integral and derivative gains respectively) and a simple proportional, or P-type, anticipatory ILC algorithm

$$u_{k+1}(t) = u_k(t) + \alpha e_k(t+1) \tag{25}$$

where α is a scalar learning gain which is chosen by the operator. The anticipatory component (the $(t+1)$ term) here means that the error value of the update is shifted forward by one sample instant. This aims to capture the trend of the error data rather than treating each sampling instant individually.

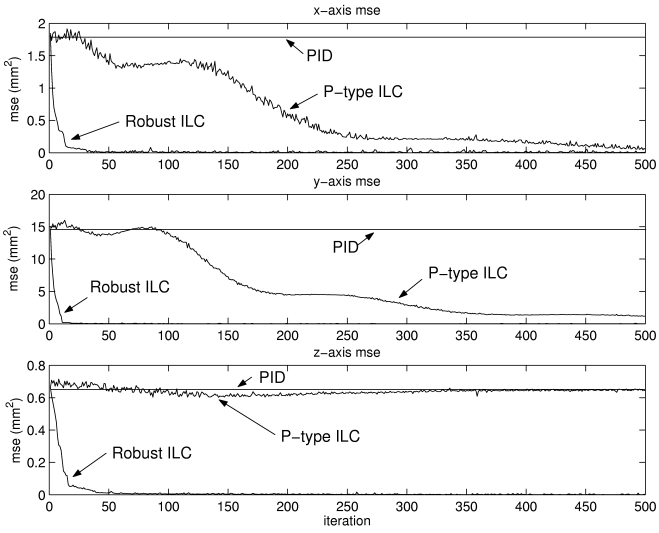

Fig. 7. Comparison of PID, P-type ILC and robust optimal ILC

Figure 7 shows the mse data obtained for the three types of controller over 500 trials, where the performance of the PID controller obviously does not change as it has no ability to learn. To allow fair comparison of the controllers, the first trial of the P-type ILC and robust optimal ILC is controlled by the PID controller. The PID controller is then switched out and the ILC controller operates alone. This automatically sets the mse for the first trial at a uniform value. For the data shown, the value of α was set equal to 0.1 for all axes, and the value of w was set to 0.

From the data it is immediately evident that both the P-type and robust optimal controllers have the ability to improve the tracking performance beyond that of the PID controller. It is also clear that the performance of the robust optimal algorithm is superior to that of the P-type, as the convergence rate and level of error reduction are far greater. The performance of the P-type algorithm also appears to be non-monotonic as there is a tendency for the mse to grow first before reducing. In an attempt to match the performance of the robust optimal algorithm, the learning gain α of the P-type algorithm was set

at 0.01, 0.1 and 1 respectively. At 0.01, the learning rate was very slow or unnoticeable. With α equal to one the learning rate was much faster, but still did not match the robust optimal controller. With this higher learning gain value the mse also began steadily increasing again after approximately 95 trials. This was accompanied by a noticeable vibration of the gantry structure which became increasingly violent. At 200 trials, the system was switched off to prevent damaging the mechanical components.

The increase in mse and the mechanical vibration are clear signs of instability. This is most likely caused by high frequency noise which is amplified at each cycle of the trial loop [14]. It is important to note that the P-type ILC is operated at a sampling frequency of 1 KHz, ten times faster than the robust optimal algorithm. Hence it is possible for much higher frequencies to be added to the control signal at each trial. High frequency noise is not a repeatable disturbance and so the ILC is unable to reduce it.

Having completed these experiments and demonstrated the relative performance of the modified steepest decent algorithm, it is now possible to investigate the properties of the robust optimal algorithm.

To apply the robust optimal algorithm it is necessary to determine G_o and U ($G_e = G_o U$) for the particular application under consideration. Clearly there is a very wide range of admissible G_o and U and here we seek to demonstrate the capabilities of this uncertainty structure by selecting these parameters to closely resemble what choices may be made by a practicing engineer. In particular, we consider in turn the cases when (i) there is an error only in the zero frequency gain of the transfer-function built from measured frequency responses, (ii) a model is produced based only on the low frequency characteristics, and (iii) the axis dynamics can be represented by a pure integrator with a tunable gain.

In the case of (i) we can investigate the consequences of this by constructing G_o starting from the measured Bode plots in the continuous domain and then investigating what happens when the positive real scalar U is varied. For (ii), a standard form of continuous-time model for a linear motor drive is $k/s(s+a)$ where k and a are positive scalars. In physical terms, this amounts to assuming that there is no interaction between the individual axes of the robot and all higher frequency dynamics are ignored. The tuned models actually used for the three axes are

$$G_{ox}(s) := \frac{11}{s(s+200)} \tag{26}$$

$$G_{oy}(s) := \frac{17}{s(s+300)} \tag{27}$$

$$G_{oz}(s) := \frac{10}{s(s+400)}. \tag{28}$$

Finally, in (iii) no account of the individual axis frequency responses is taken when obtaining G_o.

8.1. Model uncertainty – Case (i). Results for Case (i) with $w = 0$ are shown in Fig. 8 where U is selected in the range between 0.75 and 1.25. Also it follows immediately from (22) that if $U < 1$ then this will result in an increase in the learning gain and consequently performance will be severely degraded because the algorithm will over compensate for the error at each trial.

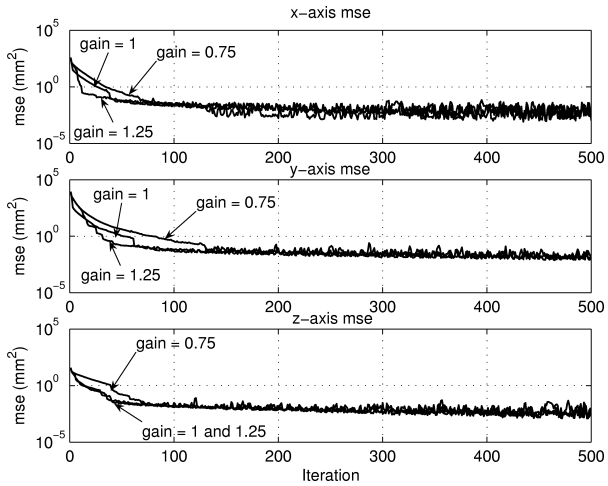

Fig. 8. mse for different scalar gains for Case (i)

As described in Note 3 one option to restore a high convergence rate, while maintaining stability, is to allow the adaptation of the magnitude of w as a function of the current level of tracking error, i.e. use

$$w_{k+1} = w_0 + w_1 ||\mathbf{e}||^2 \tag{29}$$

where w_0 and w_1 are two new tuning parameters, which must be appropriately selected to match the operation of the control system. Parameter w_0 can be used to guarantee stability by setting a baseline magnitude for w_{k+1}, while $w_1||e_k||^2$ adapts w_{k+1} to match the change in tracking error.

Considering the X-axis displacement profiles for trials 490–500 with $U = 0.5$ and $w_{k+1} = 2 \times 10^{-7}||e_k||^2$ leads to the conclusion that the adaptive variant of the gain tuning parameter has clearly improved the convergence rate and minimum tracking error. The relative performance is measured through the use of PI_{100}. This reveals that the Z-axis performance remains poor in comparison to the other axes. However, this can be corrected by setting $w_{k+1} = 0$. The required Z-axis travel is sufficiently small and hence the initial overshoot caused by the increased learning gain does not exceed axis travel limits. By trial 100, the output displacement accurately follows the reference trajectory well.

8.2. Model uncertainty — Cases (ii) and (iii). For Case (ii), the representative log mse results demonstrate that the robustness structure assumed in this work can be used to design a controller based on a reduced order model whose comparative performance is very close to that achieved with a full order model. As expected, the difference is most noticeable for the X-axis where the performance of the low-order model is noticeably worse than that of the high order model. The X-axis has the most significant high frequency dynamics and so is most affected by the simplification process.

The stability theory (see Note 4) allows G_o to be constructed from a continuous-time model of the form β/s, where β is the tunable gain, as in Case (iii). (This represents the effects of choosing any other continuous time transfer function whose phase lies between 0 and $-180°$.) Using this transfer function for controller design, a range of values of β from 0.01 to 1.0 have been considered in detail to investigate the effect of the model gain on stability and convergence. This shows that with a large value for β the rate of learning is slower. As β is reduced, the learning steadily becomes faster until an optimum β is reached. Increasing β any further then rapidly reduces the learning rate and very quickly the system performance degrades. The best values of β for each axis were found to be 0.05 for the X and Y-axes and 0.03 for the Z-axis. In all three cases the zero frequency gain of the resulting simplified model is a close match to that of the higher order model

When β is too large, the controller will assume that a small change in its effort will have a larger effect at the output. This effectively reduces the convergence rate. If β is too small, then the resulting controller assumes that large changes in controller effort are required to effect the output. Consequently there will be overcompensation for a given tracking error and performance will be severely degraded. This suggests a very simple tuning rule for the controller in this case.

1. Set β to a large value (if any form of plant model is available ensure that β generates a gain which is greater than the gain of the model at low frequency).
2. Operate the system and ensure that stability is achieved.
3. Reduce β.
4. Continue steps 2 and 3 until optimal convergence is achieved or until the system begins to severely degrade.
5. If this occurs, increase β slightly.
6. β is set.

To investigate the $\pm 90°$ stability boundary, the plant model $1/s^2$ has been used to design and implement the control law on each axis. This model generates a phase shift of $-180°$ and at low frequency each axis of the plant has a phase shift of $-90°$. The system is therefore just on the theoretical stability boundary. The gain of $1/s^2$ generates a conservative learning system for each axis. The trajectory described by the X-axis from trial 1 where the trajectory is zero to trial 201, in intervals of 20 trials and the thicker line represents the desired reference trajectory. By trial 121, the axis initially travels in the opposite direction to what is required. This effect becomes steadily worse and by trial 201, the rapid change of direction at the start of the trial is severe. The test was in fact terminated at 203 trials to prevent damaging the gantry robot. In addition, there is a growing oscillatory component of the displacement waveform which occurs at time 0.1 seconds. This initially appears at trial 41 and is clearly visible at trial 201m Small oscillations such as this can usually be seen in other ILC systems as their performance begins to degrade. These observations suggest that use of a $1/s^2$ model will produce a poor controller as it introduces an additional phase shift of $-90°$ at low frequency when compared to the plant (see also Note 4).

The results above have only considered the case when $w = 0$ but the theory predicts that a non-zero w could increase robustness of the controller. The approximate model β/s with β selected too small is an example of such a situation and we now investigate this case in more detail.

9. Conclusions

This paper has reviewed some of the major developments in the general subject area of iterative learning control. The emphasis has been on showing how advanced control algorithms in this area can be designed to give high performance which is then experimentally verified by experiments on a gantry robot system. Moreover, the experimental testing of ILC algorithms for robustness against unmodelled plant dynamics is also possible.

It has, of course, only been possible to consider a small fraction of the total subject area and for the case of linear dynamics where there is still much work to be done. Also the complete area of nonlinear ILC has to see a move forward from just trial-to-trial error convergence proofs if the subject area as a whole is to meet its true potential.

Acknowledgements. The results in this paper are from a research programme iterative learning control with experimental benchmarking between The Universities of Southampton and Sheffield in the UK and The University of Zielona Gora, Poland. This work is directed in Southampton by the author and Dr Paul L. Lewin, in Sheffield by Professor David H. Owens, and in Zielona Gora by Professor Krzysztof Galkowski. The work of the UK partners has been funded by the Engineering and Physical Sciences Research Council.

REFERENCES

[1] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operations of robots by learning", *J. Robotic Systems* 1, 123–140 (1984).
[2] K.L. Moore, *Iterative Learning Control for Deterministic Systems, Advances in Industrial Control Series*, Springer Verlag, London, 1993.
[3] H.-S. Ahn, Ch. YanQuan, and K.L. Moore, "Iterative leanring control: brief survey and categorization", *IEEE Transactions on Systems, Man and Cyberneitcs* Part C 37(6), 1099-1121 (2007).
[4] A. Bristow, M. Tharayil, and A.A. Alleyne, "A survey of iterative learning control", *IEEE Control Systems Magazine* 26(3), 96–114 (2006).
[5] D.H. Owens, N. Amann, E. Rogers, and M. French, "Analysis of iterative learning control schemes – a 2D systems/repetitive processes approach", *Multidimensional Systems and Signal Processing* 11(1/2), 125–1777 (2000).
[6] N. Amann, D.H. Owens, and E. Rogers, "Predictive optimal iterative learning control", *Int. J. Control* 69(2), 203–226 (1998).
[7] J-X. Xu and Y. Tan, "Robust optimal design and convergence properties analysis of iterative learning control approaches", *Automatica* 1867–1880 (2002).

[8] J.D. Ratcliffe, *Iterative Learning Control Implemented on a Multi-axis System*, PhD Thesis, University of Southampton, UK, 2005.

[9] J.D. Ratcliffe, P.L. Lewin, E. Rogers, J.J. Hatonen, and D.H. Owens, "Norm-optimal iterative learning control applied to a gantry robots for automation applications", *IEEE Transactions on Robotics* 22(6), 1303–1307 (2006).

[10] J.D. Ratcliffe, J.J. Hatonen, P.L. Lewin, E. Rogers, and D.H. Owens, "Robustness analysis of an adjoint optimal iterative learning controller with experimental verification", *Int. J. Robust and Nonlinear Control*, 18(10), 1089–113 (2008).

[11] K. Furuta and M. Yamakita, "The design of learning control systems for multivariable systems", *Proc. IEEE Int. Symposium on Intelligent Control*, 371–376 (1987).

[12] C.A. Desoer and M. Vidyasagar, *Feedback Systems: Input Output Properties*, Academic Press, London, 1975.

[13] J.J. Hatonen and D.H. Owens, "New connections between positivity and parameter-optimal iterative learning control", *Proc. IEEE International Symposium on Intelligent Control*, 69–74 (2003).

[14] R.W. Longman, "Iterative learning control and repetitive control for engineering practice", *Int. J. Control* 73(10), 930–954 (2000).