UNIVERSITY OF SOUTHAMPTON

Iterative Learning Control: Algorithm Development and Experimental Benchmarking

by

Zhonglun Cai

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

April 2009

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Zhonglun Cai

This thesis concerns the general area of experimental benchmarking of Iterative Learning Control (ILC) algorithms using two experimental facilities. ILC is an approach which is suitable for applications where the same task is executed repeatedly over the necessarily finite time duration, known as the trial length. The process is reset prior to the commencement of each execution. The basic idea of ILC is to use information from previously executed trials to update the control input to be applied during the next one. The first experimental facility is a non-minimum phase electro-mechanical system and the other is a gantry robot whose basic task is to pick and place objects on a moving conveyor under synchronization and in a fixed finite time duration that replicates many tasks encountered in the process industries. Novel contributions are made in both the development of new algorithms and, especially, in the analysis of experimental results both of a single algorithm alone and also in the comparison of the relative performance of different algorithms. In the case of non-minimum phase systems, a new algorithm, named Reference Shift ILC (RSILC) is developed that is of a two loop structure. One learning loop addresses the system lag and another tackles the possibility of a large initial plant input commonly encountered when using basic iterative learning control algorithms. After basic algorithm development and simulation studies, experimental results are given to conclude that performance improvement over previously reported algorithms is reasonable. The gantry robot has been previously used to experimentally benchmark a range of simple structure ILC algorithms, such as those based on the ILC versions of the classical proportional plus derivative error actuated controllers, and some state-space based optimal ILC algorithms. Here these results are extended by the first ever detailed experimental study of the performance of stochastic ILC algorithms together with some modifications necessary to their configuration in order to increase performance. The majority of the currently reported ILC algorithms mainly focus on reducing the trial-to-trial error but it is known that this may come at the cost of poor or unacceptable performance along the trial dynamics. Control theory for discrete linear repetitive processes is used to design ILC control laws that enable the control of both trial-to-trial error convergence and along the trial dynamics. These algorithms can be computed using Linear Matrix Inequalities (LMIs) and again the results of experimental implementation on the gantry robot are given. These results are the first ever in this key area and represent a benchmark against which alternatives can be compared. In the concluding chapter, a critical overview of the results presented is given together with areas for both short and medium term further research.

Contents

List of Figures

List of Tables

Nomenclature

s	Laplace transform variable
z	z-transform variable
k	Trial number
u	Input signal sequence
e	Error signal sequence
y	Output signal sequence
t	Continuous time
p	Discrete time step
$\mathbb{Z}$	Set of integers

Acknowledgements

Firstly, thanks to my parents, my fiancee Yuanyuan Song and all my family for their financial support and encouragement of my study abroad. Secondly, great thanks to my supervisors Professor Eric Rogers and Dr Paul Lewin for their help on not only my research but also my living in UK. Moreover, special thanks to Dr Chris Freeman who did the proof reading of my thesis and gave me great support during my research. Also, many thanks to Dr James Ratcliffe and Lukasz Hladowski (University of Zielona Gora, Poland) for their continuous help in the research discussion and friendship.

Chapter 1

Introduction

Iterative learning control (ILC) is concerned with reference tracking control problems, where the required trajectory is repeated ad-infinitum over a finite duration termed the trial length. This applies to many industrial applications, such as injection molding, robotics, automated manufacturing plants, and food processing to name a few.

Each implementation of the trajectory is commonly known as a trial. 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 in order to sequentially improve performance from trial-to-trial. It is important to note how ILC differs from other control systems design methods that use learning-type ideas or strategies, such as adaptive control, neural networks, and repetitive control (RC). In particular, adaptive control designs modify the controller, which is a system, whereas ILC modifies the control input signals. Additionally, adaptive controllers typically do not take advantage of the information contained in repetitive command signals.

Neural network based control algorithms also modify controller parameters rather than a control signal and, in general, require the modification of large networks of nonlinear neurons. This, in turn means that extensive training data is required that may not be available and also fast convergence may be difficult to achieve. Also neural network based algorithms take longer to calculate which is not suitable for real-time processes. In structural terms, ILC is closely linked to repetitive control (RC) where the main difference (see below for more discussion) is that RC is meant for use in a continuous operation. ILC is intended for discontinuous operation where the system is reset at the end of each trial to begin the next one where, in general, it may not be possible to assume that the same initial conditions apply for each trial. Moreover, the time taken to reset can be used to compute the control input for the next trial. Also there is frequently a trade-off to be made between reduction in the trial-to-trial error and the performance achieved along the trial.

An operation found in many industries to which ILC can be applied is a gantry robot whose task is to collect objects from one location, place it on a conveyor under synchronisation, and then repeat this operation over and over again. Hence data collected during one execution can be used in the time taken to reset to update the control signal for the next execution. A laboratory version of this operation will form the starting point for many of the experimental results reported in this thesis.

An application to which RC can be applied to good effect is the control of a hard disk drives read/write head, in which each trial is a full rotation of the disk, and the next trial immediately follows the current trial. The difference between RC and ILC is the setting of the initial conditions for each trial. In ILC, the initial conditions are often assumed to be the same at the start of each trial although in practice this may not be a valid assumption or one that is difficult to achieve. In RC, the initial conditions are set to the final conditions of the previous trial. The difference in initial-condition resetting has led to different analysis techniques and results but in some cases a form of duality can be established.

Since the original work, over 20 years ago, the ILC subject area has been the focus of much research effort. A considerable volume of this has been on trial-to-trial error convergence analysis, initially for linear plant dynamics, or approximated as such, with extension to the nonlinear case. Developments in sensor and actuator technology etc now mean that experimental verification is possible and this is essential if, for example, the aim is to apply ILC widely in industrial and related applications. There has already been work in this area but much remains to be done.

In this thesis, the major contributions are new results in experimental benchmarking of linear model ILC algorithms, coupled with algorithm development and enhancement where necessary. Moreover, a particular feature is the emphasis of benchmarking the performance of competing algorithms that is also essential for widespread use of ILC but has been the subject of relatively little work until now. Two experimental facilities are used in this thesis. These are a non-minimum phase electro-mechanical system and a gantry robot. Both of these facilities have previously been used to experimentally benchmark a range of simple structure ILC algorithms, such as those based on the ILC versions of the classical proportional plus derivative error actuated controllers, and some state-space based optimal control based algorithms.

The thesis begins with an overview of ILC literatures in Chapter 2 which summarises the development of ILC algorithms in different categories. In Chapter 3, a novel algorithm named the Reference Shift ILC algorithm (RSILC) is developed and experimentally tested on the non-minimum phase plant which improves performance compared with previous algorithms. In Chapter 4, a series of stochastic learning algorithms are implemented on the gantry robot and a comparison of performance is provided. Some modifications required to obtain the results are also given. Chapter 5 introduces the development of ILC techniques based on 2-dimensional (2D) system theory (in particular, the theory of discrete Linear Repetitive Processes) which provided the stability analysis of both the trial-to-trial performance and the performance achieved along the trials. In Chapter 6, performance comparisons and some practical issues related to this research are detailed. Finally in Chapter 7, conclusions are given together with directions for both short and longer term future research.

Chapter 2

Literature Review

This chapter gives a general review of the development of iterative learning control (ILC) algorithms and related techniques. It will be seen that most research of ILC has concentrated on developing and analysing new algorithms. Application based ILC research in the past was heavily outweighed by theoretical work. However, aided by developments in sensor and actuator technology and computing hardware and software, it is now relatively easy to undertake experimental benchmarking with a view to aiding onward industrial application of ILC schemes. This thesis reports substantial new contributions in this general area and as a preliminary step, a survey of the relevant literature is now given.

2.1 Overview of Iterative Learning Control

The idea of ILC can be traced back at least to an article published in Japanese by Uchyama (1978) which is sometimes quoted as the origin of the subject. However, the articles by Arimoto et al. (1984a,b) provided a formal definition of ILC and are the most frequently cited. The theory behind ILC is motivated by human learning. Humans learn by practicing or repeating a task until it is perfected. Many examples can be found in sports training: basketball shootouts, penalty kicks or free kicks, golf swings, etc. Players train thousands of times to gain good performance using their muscle memory. Arimoto et al. (1984a) investigated if similar methods could be applied to robotics, to produce some form of learning autonomy without the need for human intervention. Sensors would replace human eyes and controllers and actuators would take the place of muscle memory. So ILC is a control method which is designed for improving tracking control performance when a system is performing a repeating task. The system is run over and over again on a finite time interval which is called a trial (also known as a pass or an iteration in some literature). The main idea is to use the tracking error of the last trial to update the control input signal for the current trial of ILC and the goal is to obtain better tracking performance from trial to trial. The block diagram (Figure 2.1) shows the structure of the basic ILC loop and the learning progress of the related signals.

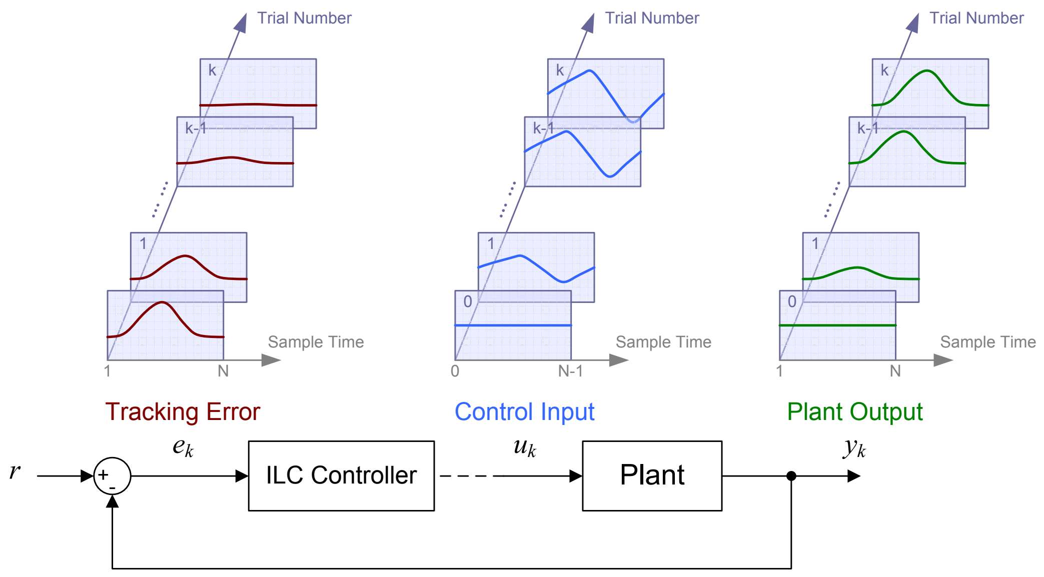


FIGURE 2.1: Block diagram for an ILC process

There are several key features of the principle of ILC which distinguish it from other control techniques (e.g. repetitive control).

- The reference trajectory is pre-defined over a finite time interval $0 \leq t \leq T$ called a trial, where T denotes the trial length. All the control related efforts (e.g. input/output/error signals) are accordingly defined over the same finite interval.
- The initial conditions of the system are reset before each trial commences. This means there is a gap time in which computation of the control effort can be achieved. For example, input updating and filtering.

During the last twenty years, a great number of control ILC algorithms have been developed. ILC has become a framework encompassing many varieties of control approach. Each one has its own merits in terms of performance such as convergence, robustness, learning speed, or suitability for special plants. A survey paper which provides an overview of ILC research in recent years is by Bristow et al. (2006), and Ahn et al. (2007) has categorized almost all the ILC algorithms proposed between 1998 and 2004 from different aspects, such as mathematical formulation, type of application, type of systems, etc.

The main reason why ILC has attracted considerable research effort is its simplicity and potential effectiveness. Generally, the only information controllers can learn from is the tracking error. There are plenty of ways to use this error which gives rise to the large number of ILC algorithms. The remainder of this chapter gives a review of papers concerned with ILC algorithms. Roughly, the algorithms can be divided into two categories. Firstly there are algorithms which assume no prior knowledge of the plant model, and the controllers generally apply simple computation on the tracking error vector. The other category comprises model-based algorithms which firstly need a mathematical representation of the plant (e.g. transfer function, zero-pole, state space matrices, etc.) before the algorithms can be applied. Also, most of these algorithms need a significant amount of matrix computation to be conducted in the gap between each learning trial. Discussion related to these two categories of algorithm is given later in this chapter.

2.2 Basic ILC Algorithms

Basic algorithms are simple and easy to be implemented as only manipulation of the error vector is generally needed in the update. In most of cases, the computation can be carried out in the gap (i.e. the resetting time) between each trial, therefore it does not require a controller with high computing performance. This is a great attraction for both researchers and manufacturers, as these algorithms might provide good or acceptable performance for a very low cost, for example, D-type ILC Arimoto et al. (1984a), P-type Arimoto et al. (1985).

2.2.1 D-type ILC Algorithm

Arimoto et al. (1984a) formulated a derivative type (D-type) ILC control algorithm in the form of (2.1), which can be also represented in the block diagram shown in Figure 2.2.

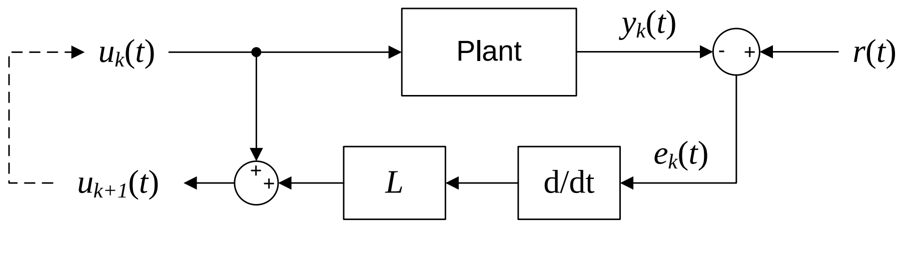


FIGURE 2.2: Block diagram for the D-type ILC algorithm

$$u_{k+1}(t) = u_k(t) + L\dot{e}_k(t) \tag{2.1}$$

$$e_k(t) = r(t) - y_k(t) \tag{2.2}$$

where $u_k(t)$ is the input signal applied on the kth trial, L is the learning gain, $\dot{e}_k(t)$ is the error derivative, $r(t)$ is the reference trajectory and $y_k(t)$ is the plant output. Here $t \in [0, T]$, where T is a known finite time (trial length). The plant appearing in that paper, which is a DC motor, has an integral effort and use of the derivative of the tracking error is able to give satisfactory learning progress. Arimoto et al. (1984a) also gave the condition of convergence for the D-type ILC as

$$||1 - CBL|| < 1 \tag{2.3}$$

where B and C are from the state space model of the plant, L is the learning gain from ILC update law in (2.1), $|| \cdot ||$ is the norm for l_2 space.

2.2.2 P-type ILC Algorithm

In Arimoto et al. (1985), the D-type algorithm was transferred into a simpler form which uses only the error signal with a proportional gain (so-called P-type, see Equation (2.4)) in order to potentially avoid amplifying small noise signal through differentiation and destabilizing the control system. Also, without the need to calculate the derivatives, it enables great simplicity and ease of implementation. The law is given by:

$$u_{k+1}(t) = u_k(t) + Le_k(t) \tag{2.4}$$

where L is the proportional learning gain.

2.2.3 Other Basic ILC Algorithms

A P-I-D combined type ILC algorithm was introduced by Kim and Kim (1996) as a general form. When the gains of the appropriate parts (i.e. the proportional and integral part or the derivative and integral part) are set to zero, this becomes P-type or D-type ILC respectively. A higher order ILC algorithm uses more than one trial of tracking error to update the control input and enables fast convergence and improves the stability of the control loop (Bien and Huh, 1989; Ahn et al., 1993).

Barton et al. (2000) reported that instead of using the instantaneous error, use of the error occurring a number of samples ahead is able to successfully deal with a system with phase lag. Freeman et al. (2005b) later introduced this technique as the phase-lead algorithm because it is originally based on frequency domain analysis. The same idea was considered by Wang and Longman (1996). The phase-lead algorithm with a carefully selected filter can be used to successfully control the non-minimum phase plant (Freeman et al., 2005b).

An ILC update law with a forgetting factor was discussed in Wang and Longman (1996). Using the forgetting factor reduces the noise transmission and increases the noise immunity of the algorithm. This effect becomes obvious when considering the Nyquist plot of a system. In the cases where the poles of the system are very close to the unit circle, the forgetting factor has the ability to draw the poles away from the unit circle and reduce the risk of instability (Lewin, 1999).

Below (Table 2.1) is a summary table of the reviewed basic ILC algorithms.

Algorithm	Equations
D-type ILC	$u_{k+1}(t) = u_k(t) + L\dot{e}_k(t)$
P-type ILC	$u_{k+1}(t) = u_k(t) + Le_k(t)$
PID-type ILC	$u_{k+1}(t) = u_k(t) + K_p e_k(t) + K_i \int e_k(t)\mathrm{d}t + K_d \dot{e}_k(t)$
Higher order ILC	$u_{k+1}(t) = u_k(t) + \sum_{n=0}^{N} \beta_n(t) e_l(t)$
Phase lead ILC	$u_{k+1}(t) = u_k(t) + Le_k(t+\tau), \tau \in \mathbb{Z}$
Forgetting factors	$u_{k+1}(t) = \beta[u_k(t) + Le_k(t)]$

TABLE 2.1: Basic ILC algorithms in equations

2.2.4 ILC with the Assistance of Feedback Controller

In some cases ILC can be ‘added on’ to a conventional control scheme. In particular, a standard feedback control scheme is first designed to either stabilise an unstable plant or modify the transient response of an already stable system. This is sometimes known in the ILC community as a hybrid arrangement. Consider Figure 2.3 which is a schematic of ILC performance compared against a standard control scheme here taken for simplicity as a three term PID loop. The line corresponding to the PID controller alone is horizontal as it does not have any trial to trial learning capability. The blue line represents ILC alone and the final line represents the hybrid case and is drawn to illustrate that in many cases this leads to even better performance.

Generally, the hybrid approach has two alternative configurations which are serial arrangement and parallel arrangement. It has been reported by Ratcliffe et al. (2005a) that the serial arrangement is easier to apply but potentially will lead to increased noise. The parallel arrangement operates independently of the learning controller so that it is able to satisfactorily deal with sudden changes in the plant dynamics. Tayebia and Islamb (2006) gave a series of experimental results using a PD controller cooperating with an ILC controller to cope with unknown parameters and disturbances.

It has been reported that in a closed loop control, using the ILC tracking error may result in conflicting feedback and feedforward actions (Dijkstra and Bosgra, 2002a). This can be avoid by introducing frequency weighting to the measured errors but Dijkstra and Bosgra (2002a) state that the problem can be also solved without any extra filter by taking the desired behaviour of ILC into account and choosing the input signal accordingly.

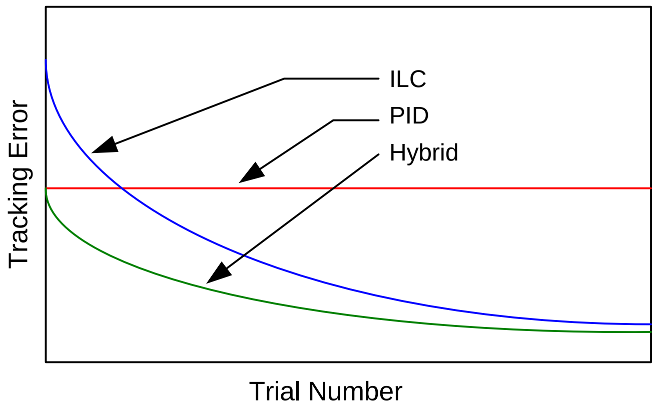


FIGURE 2.3: Performance comparison of ILC, PID and hybrid approaches.

2.3 Model-based Algorithms

Model-based ILC algorithms are developed based on a pre-assumed plant model. Modelling of the plant must therefore be conducted before algorithm development and implementation. An accurate model of the system can be used with the real plant in parallel to provide information for the controller which might not be available from other methods (Roover and Bosgra, 1997). In most cases, the model is useful for providing information about system states which can not be measured easily. As well as feeding the control input into the real plant, the difference between the real plant output and model-computed output can also be supplied to the model for increased accuracy.

Phan and Frueh (1996, 1999) introduced a novel way of implementing a model-based controller by learning the plant model during each trial. As the reference trajectory is fixed for all trials, it is only necessary to learn the dynamics of the plant for the particular trajectory as other dynamics will not be excited. After each trial is performed, a set of data for the control input and the corresponding output can be obtained. Using this data, a set of basis functions can be trained to emulate the behaviour of the plant and produce a model. The model is then used for standard model-based control. Effectively, this method can be integrated as a real-time plant identification technique. Phan and Frueh (1999) implemented this model-based controller on an experimental apparatus, consisting of a number of parallel steel rods held together by a thin spring-steel wire. Actuation force was supplied to one rod, while the tip of another rod was required to follow a set trajectory. The model based controller successfully reduced the tracking error by over one order of magnitude.

Generally, for a discrete LTI SISO system given by:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{2.5}$$

over a finite time interval $0 \leq t \leq N$ $(N < \infty)$ these plant dynamics can be then written in matrix form as

$$G = \begin{bmatrix} CB & 0 & 0 & \cdots & 0 \\ CAB & CB & 0 & \cdots & 0 \\ CA^2B & CAB & CB & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{N-1}B & CA^{N-2}B & \cdots & \cdots & CB \end{bmatrix} \tag{2.6}$$

where A,B,C represent the Markov parameters in the system (2.5).

2.3.1 Inverse Model ILC

There are many examples of model based algorithms developed in recent years which explicitly use this representation. These include inverse model ILC, adjoint ILC, and optimal ILC. Theoretically, the inverse model method can been seen as an ideal solution to the linear control problem. But it requires an ideal model representation in mathematical form. If the control input $u_k(t)$ and system output $y_k(t)$ are represented as a linear transformation and if ILC is successful, then an ideal control input can be found eventually as the trial number k goes to infinity. The system output $y_\infty(t)$ will be exactly the same as the reference signal $r(t)$. By reversing the left hand side and right hand side in Equation (2.8), the ideal control input $u_\infty(t)$ can be obtained.

$$y_k(t) = G \cdot u_k(t) \tag{2.7}$$

$$r(t) = G \cdot u_\infty(t) \tag{2.8}$$

$$u_\infty(t) = G^{-1} \cdot r(t) \tag{2.9}$$

Such an inverse can be similarly achieved by exchanging the denominator polynomial terms and numerator polynomial terms in a transfer function of the plant model or calculating an inverse matrix of a model matrix given by (2.6). But practically, due to the inaccuracies of the model, this technique is not suitable for implementation on real plants as the inverse, if it exists, is likely to be extremely sensitive to modelling error and noise. Therefore a combination of ILC and an inverse model technique has been considered during a recent investigation (Markusson et al., 2001).

However, the inverse model method is not feasible for a wide range of control systems, for example, a non-minimum phase plant. This is because the inversion of a non-minimum phase plant will lead to an unstable system as there will be at least one pole in the right-half complex plane. In Sogo et al. (2000), a stable inverse method has been introduced, which solved the problem of finding a stable inverse for non-minimum phase plants.

Other model based ILC algorithms (e.g. norm optimal ILC, ILC in stochastic systems) are mainly rooted in distinct areas of control theory, e.g. optimal control, control of stochastic systems, etc. This class of algorithms is reviewed next.

2.3.2 Optimal ILC Approaches

Optimisation is the process of looking for the best solution for varieties of problems when a system has more than one specific ‘point’ that needs to be considered. Applying optimisation to control problems will typically balance several competing objectives for a controller, for example, system stability, complexity, robustness, convergence, etc. Generally, a cost function will be minimised and by varying the cost function weighting terms, it is possible to adjust the balance between these performance criteria. In ILC schemes, the cost function usually includes a description of the tracking error, so that the optimal controller attempts to reduce it to a minimum (Chen and Fang, 2004; Gunnarsson and Norrlöf, 1999). Many strategies implement a gradient descent approach for the minimisation process (Togai and Yamano, 1985). A linear quadratic optimal learning control algorithm has been designed by Frueh and Phan (2000) and this approach is developed to allow the optimisation to avoid requiring detailed knowledge of the system. Hatzikos et al. (2004) provide a useful summary of optimal ILC development, before going on to investigate the use of Genetic Algorithms (GA) to develop an optimal controller which can be applied to nonlinear plants.

In Amann et al. (1996b), a norm-optimal approach was proposed to determine the control input at the $(k+1)^{th}$ trial by minimizing the following cost function:

$$J_{k+1}(u_{k+1}) = \|e_{k+1}\|^2_{\mathcal{Y}} + \|u_{k+1} - u_k\|^2_{\mathcal{U}} \tag{2.10}$$

Where $\mathcal{U}$ and $\mathcal{Y}$ are l_2-spaces, the norms $\|\cdot\|$ are the appropriate norms for the input output spaces $\mathcal{U}$ and $\mathcal{Y}$. The cost function involves both the tracking error and also the change in the input sequence. Minimising the latter aims to prevent excessive control effort from being applied to the plant and helps to generate smooth inputs to actuators (Amann et al., 1996a; Lee et al., 2000). It also makes the learning somewhat conservative. The cost function can be written in a more familiar algebraic form involving sums. The related control input update law can be given as:

$$J_{k+1} = \sum_{t=1}^{N} e^T_{k+1}(t)Q(t)e_{k+1}(t) + \sum_{t=0}^{N-1}[u_{k+1}(t) - u_k(t)]^T R(t)[u_{k+1}(t) - u_k(t)] \tag{2.11}$$

Where $Q(t)$ and $R(t)$ are weighting matrices used to adjust the balance between optimally reducing the tracking error and limiting the change in the input sequence from trial to trial. The optimised input update algorithm has the following form:

$$u_{k+1} = u_k + R^{-1}G^T Q e_{k+1} \tag{2.12}$$

Here e_{k+1}, the current trial error, is used in the input update rather than the past error e_k. The weights Q and R are chosen as:

$$Q = qI$$
$$R = rI$$

where q and r are scalars and I is the identity matrix of appropriate dimension.

Using the current trial error and system transpose makes the system noncausal, resulting in an algorithm which is impossible to implement in practice. However, Amann et al. (1996b) reported that the problem can be solved by means of an additional optimal feedback control formulation of the algorithm. Later the algorithm has been implemented on a gantry robot (Ratcliffe et al., 2006b) and the performance has been found to outperform all other controllers used on that plant. Some of the results generated by the norm-optimal algorithm on the gantry robot have been used to compare with the ILC algorithm applied in this thesis. A simplified version of the norm-optimal ILC algorithm named the Fast norm-optimal ILC (F-NOILC) has been implemented by Ratcliffe et al. (2005b). Also these approaches have been considered and implemented to the control of an accelerator based free electron laser with improved performance reported (Kichhoff et al., 2008).

2.3.3 Robust ILC

It is usually impossible to identify a system model which is an exact representation of the physical plant itself. However, within certain tolerance bounds, the performance might be guaranteed theoretically. But it is not evident that an ILC controller will be able to cope with disturbances, variations in the plant and modelling errors. H_∞ (i.e. "H-infinity") methods are used in control theory to synthesize controllers achieving robust performance or stabilization. Liang and Looze (1993) derive and define robustness conditions to ILC systems specifically. Roover (1996) extended these analysis techniques for ILC in the H_∞ mathematical framework into the actual synthesis of an ILC controller and showed how this process can be generalised to the synthesis of an H_∞ (sub)optimal controller. Another important results is that the uncertainty of a system can be explicitly incorporated into the design procedure via weighting functions, turning it into a robust performance synthesis problem. A practical design method for the past-error feedforward scheme has been given for the open-loop case (Roover and Bosgra, 2000).

Sensitivity is an important measure in robustness analysis, as it describes how one component or variable of a system is affected by a change in another component or variable. High sensitivity implies that a small change of one variable has a large effect on others. Therefore low sensitivity is generally desired when designing a robust controller. Tayebi and Zaremba (2003) propose a generic approach to designing robust learning control systems. By using a hybrid combination of a standard feedback controller in parallel with a learning controller, stability is theoretically guaranteed, as long as the feedback controller meets a specified robustness performance condition. Therefore a simple calculation leads to a performance weighting function, which is applied to the input of the learning controller and guarantees overall stability.

A special form of ILC (denoted by Hankel ILC) with separate actuation and observation time windows was designed and shown to be able to deal with model uncertainties (Wijdeven and Bosgra, 2008). Donkers et al. (2008) also studied robustness against model uncertainty of norm optimal ILC controllers. French (2008) provided the conditions of robust stability by using gap metrics.

Almost all of the currently reported work on robust control is for discrete time systems whose dynamics are then written in the form of (2.5) that then forms the basis for onward analysis. This means that the along the trial dynamics cannot be explicitly included. Some work on an alternative approach using 2D systems theory is considered in this thesis and see also the work of, for example, Ratcliffe et al. (2008).

2.3.4 ILC in a Stochastic Setting

A number of ILC algorithms have been developed in a stochastic setting over recent years. The results currently available are concerned with algorithm derivation and the establishment of various fundamental systems theoretic properties. For example, in Saab (2001a,b, 2003), a discrete-time stochastic learning algorithm has been introduced for both P-type and D-type ILC. The input updates have the same form of (4.6) and (4.7), although the algorithm is based on the discrete-time system and the learning gain K_k is computed as

$$\begin{aligned} K_k &= \mathbf{P}_{u,k}(CB)^T \cdot [(CB)\mathbf{P}_{u,k}(CB)^T + (C - CA) \\ &\quad \mathbf{P}_{x,k}(C - CA)^T + C\mathbf{Q}_t C^T + \mathbf{R}_t]^{-1} \qquad (2.13) \\ \mathbf{P}_{u,k+1} &= (I - K_k CB)\mathbf{P}_{u,k} \qquad (2.14) \\ \mathbf{P}_{x,k+1} &= A\mathbf{P}_{x,k}A^T + B\mathbf{P}_{u,k}B^T + \mathbf{Q}_t \qquad (2.15) \end{aligned}$$

where $\mathbf{Q}_t = E(\omega(t,k)\omega(t,k)^T)$, $\mathbf{P}_{x,0} = E_k(\delta x(0,k)\delta x(0,k)^T)$ are positive semi-definite matrices, $\mathbf{R}_t = E_k(v(0,k)v(0,k)^T)$ is a positive definite matrix and $\mathbf{P}_{u,0} = E(\delta u(t,0)$ $\delta u(t,0)^T)$ is a symmetric positive definite matrix (E and E_k are the expectation operators with respect to the time domain and the trial domain respectively, $\delta x(0,k), \delta u(t,0)$ are the initial state error and the input error respectively).

A sub-optimal algorithm was proposed following the stochastic learning algorithm which simplified the computation of the learning gain matrix by removing the $\mathbf{P}_{x,k}$ term and hence the learning control gain matrix K_k is set to

$$\begin{aligned} K_k &= \mathbf{P}_{u,k}(CB)^T \cdot [(CB)\mathbf{P}_{u,k}(CB)^T + C\mathbf{Q}_t C^T + R_t]^{-1} \qquad (2.16) \\ \mathbf{P}_{u,k+1} &= (I - K_k CB)\mathbf{P}_{u,k} \qquad (2.17) \end{aligned}$$

where $\mathbf{P}_{u,0}$ and $\mathbf{Q}_t$ are the same matrices as given above. Until present, there is no reported implementation which applies these approaches to any kind of physical platform for verification or comparison. Chapter 4 will report the first ever experimental results on stochastic ILC, including some performance focused modifications of the basic algorithms.

2.3.5 ILC based on 2D Systems Theory

Multidimensional, or nD, linear systems propagate information in $n > 1$ independent directions and examples can be found in many different aspects of, for example, circuits and signal processing. One of the most applications relevant cases is systems recursive over the positive quadrant of the 2D plane for which Roesser (1975) introduced the following state-space model

$$\begin{aligned} \begin{bmatrix} x^h(i+1,j) \\ x^v(i,j+1) \end{bmatrix} &= A \begin{bmatrix} x^h(i,j) \\ x^v(i,j) \end{bmatrix} + Bu(i,j) \\ y(i,j) &= C \begin{bmatrix} x^h(i,j) \\ x^v(i,j) \end{bmatrix} \qquad (2.18) \end{aligned}$$

Here x^h and x^v are the horizontal and vertical state components and i and j are non-negative integer-valued horizontal and vertical coordinates.

An alternative state-space model first introduced by Fornasini and Marchesini (1978) is

$$\begin{aligned} x(i+1,j+1) &= A_1 x(i,j+1) + A_2 x(i+1,j) \\ &\quad + A_0 x(i,j) + Bu(i,j) \\ y(i,j) &= Cx(i,j) \qquad (2.19) \end{aligned}$$

Here the state updating takes place from (i,j) to $(i+1,j+1)$ distinguishing it from the shifting operator used in the standard discrete-time state space model, i.e. the state vector is not split into horizontal and vertical sub-vectors. Note also that these models are not totally independent and transformations exist that enable one form to be converted to the other and vice versa. Note also that a transfer-function description of 2D (with a natural extension to 3D and so on) linear systems is, in general, a function of two independent variables and hence the need to work with functions of two (or more) indeterminates and hence immediate complications arise in trying to extend standard linear systems theory. For example, primeness for functions of two or more indeterminates is no longer a simple concept. A detailed treatment of this area can be found in, for example, Rogers et al. (2007) and the relevant cited references.

ILC can be treated as a 2D linear system where one direction of information propagation in the 2D models above represents the dynamics along the trial and the other the trial-to-trial dynamics. This fact is known in the literature see, for example, Geng et al. (1990) and Kurek and Zaremba (1993) where the latter paper shows how to formulate the basic problem of trial-to-trial error convergence of an ILC scheme in terms of the asymptotic stability of its 2D Roesser (1975) model interpretation.

In the 2D Roesser (1975) and Fornasini and Marchesini (1978) state space models information propagation in both directions takes place over an infinite duration whereas in ILC information propagation along the trial only occurs over the finite and fixed trial length. Repetitive processes are a distinct class of 2D systems where information propagation in one of the two directions only occurs over a finite duration and hence they should be a more natural setting for the analysis of ILC schemes. In particular, this setting could provide a very powerful approach to the analysis and design of ILC control laws for both trial-to-trial error convergence with control over the along the trial dynamics. For example, a control law with just these properties could be highly appropriate for a gantry robot whose task is to collect an object from a location, place it on a moving conveyor, and then return for the next one and so on. If, for example, the object has an open top and is filled with liquid, and/or is fragile in nature, then unwanted vibrations during the transfer period could have very detrimental effects.

In Hladowski et al. (2008b) and Hladowski et al. (2008a), some new ILC algorithms based on 2D systems theory particularly linear repetitive processes, using either the state-feedback setting given by (2.20) or the output-feedback setting of (2.22) have been developed

$$u_{k+1}(t) = u_k(t) + K_1\eta_{k+1}(t+1) + K_2 e_k(t+1) \tag{2.20}$$
$$\eta_{k+1}(t+1) = x_{k+1}(t) - x_k(t) \tag{2.21}$$

$$u_{k+1}(t) = u_k(t) + K_1\Delta y_k(t+1) + K_2\Delta y_k(t) + K_3 e_k(t+1) \tag{2.22}$$
$$\Delta y_k(t+1) = y_k(t) - y_{k-1}(t) \tag{2.23}$$

Here η_{k+1} denotes the difference between the current system state and the previous system state. Similarly Δy_k is the difference between the current output and the previous output. K_1, K_2 are the learning parameters which will be computed and adjusted using 2D system theory.

The development of these algorithms and their experimental verification on the gantry robot is the subject of Chapter 5.

2.4 Experimental Verification of ILC

In the late 1980s, there appeared a number of papers extending the work of Arimoto et al. (1984a) to various applications. For example, Kawamura et al. (1985) presented the application of ILC algorithm on a three-degree freedom robot manipulator and reported that the robot can be trained to polish a surface of a curved plate. Bondi et al. (1988) considered ILC on robotic manipulators with feedback design, however only simulation results were presented. Also, Oh et al. (1988) applied an ILC algorithm on a two-link robot manipulator but only numerical examples were provided. Bien and Huh (1989) developed a higher-order ILC algorithm and provided a convergence proof. They provided examples via simulation and stated that the algorithm ensured good performance even in the presence of disturbance.

In the mid 1990s, some researchers attempted to apply ILC practically in an industrial setting. For example, Luca and Panzieri (1994) proposed a simple iterative learning algorithm for generating gravity compensation in a flexible two-link robot arm and placed emphasis on the effectiveness of the approach. Moon et al. (1996) applied ILC to the track-following control of an optical disk drive. Roover (1996) reported how the synthesis of an ILC controller can be generalised to the synthesis of an H_∞ suboptimal controller and verified the method on a wafer stage experimental facility. In Lee and Bien (1996) ILC was considered with application to the heat-up phase of a batch reactor and it was found that the method yields a pronounced improvement in heat-up phase operation as consistent heat-up profiles with a minimised settling time were obtained. Kim and Kim (1996) stated that a PID-type ILC controller can improve machining accuracy when CNC machine tool performs repetitive machining tasks. Moreover, there are a number of papers which mention ILC application to robotic manipulators but compared with the number of papers for ILC in theory, experimental based literature is far less well represented. After 2000, more and more ILC papers have considered experimental applications. However most of the papers with experimental verification have considered very few algorithms and there are few references that can be found with comparison of algorithm performance and experimental benchmarking.

ILC is designed for performing repeating tasks, typically by robots in a variety of industries. Therefore experimental verification prior to the implementation on industrial applications is essential as there are a number of practical problems in this area and some of them are of critical importance. Also, the process of algorithm development for ILC is currently becoming faster and faster. A number of ILC algorithms are however remaining at the stage of mathematical formulation. Most of them have associated convergence proofs, however the assumptions behind the theory are designed to support the proof. Whether the algorithm is suitable or not for application to real plants depends on the results of experimental tests. Therefore, experimental benchmarking and comparison is extremely important as it sheds light on a different aspect than purely mathematical proof. Here, in this thesis, papers mentioning non-minimum phase systems and robotic systems are further reviewed in the remaining sections of this chapter to underpin the relevance of the experimental testing on the non-minimum phase electro-mechanical system and the gantry robot reported in this thesis.

2.4.1 Non-minimum Phase Systems

A non-minimum phase linear system has at least one of its zeros located in the right half s-plane. In the linear case at high frequencies, minimum phase zeros induce a +90 degree phase shift while non-minimum phase zeros produce a -90 degree phase shift. Another key feature of non-minimum phase systems is that the characteristic step response of the plant will initially peak in the opposite direction to the final steady state output. This makes non-minimum phase systems very difficult to control because the controller must anticipate that the plant will initially react in the opposite direction to what is desired. Examples of non-minimum phase systems in engineering practice include large ships turning at high speed (Dutton et al., 1998), and the vertical displacement of helicopters.

A common approach to using ILC for non-minimum phase systems is to find the inverse model of the plant (Ghosh and Paden, 2001). This is not a simple technique, because inverting the transfer function of the plant by causal filtering will produce a system which is unstable (Markusson et al., 2001). The solution is to use a technique known as stable inversion (Sogo et al., 2000).

This technique separates the plant into a minimum phase element plus a non-minimum phase element. Each of these elements can then be treated separately. The causal element will have a stable inverse which can be found by causal filtering, while the anti-causal element will have a stable inverse which can only be found by anti-causal filtering. Ghosh and Paden (1999, 2002) also propose an alternative method for finding an approximate pseudo-inverse model for non-minimum phase plants.

There are a number of papers considering control of non-minimum phase plants (e.g. non-minimum phase systems with feedback control, and using optimal control). However, there exist very few examples in the literature concerned with ILC applied to non-minimum phase systems, and the majority of them are simulation based studies. Freeman (2004) developed a non-minimum phase mechanical experimental facility consisting of a number of mechanical components such as springs, inertias and dampers. These components are arranged in a special setup to achieve the non-minimum phase characteristic. In Freeman et al. (2005b), a number of ILC and RC control schemes were tested and compared using the facility and it was found that a simple phase-lead algorithm was able to control the plant successfully when used with carefully selected parameters and filters.

Since then very few practical studies have been carried out using non-minimum phase plants. Experimental benchmarking of ILC algorithms using non-minimum phase plant is therefore a fertile area of study. Chapter 3 concerns the derivation of a novel ILC algorithm using non-minimum phase system which is called the reference shift algorithm. Experimental results are provided which show that the resulting performance of the proposed algorithm is superior to that which has been achieved previously (Cai et al., 2007, 2008).

2.4.2 Robotic Systems

In the early stages of the ILC concept, Arimoto et al. (1984a) developed an ILC method specifically to improve the trajectory tracking performance of robots. Most manufacturing robots are designed to replace human beings in the completion of millions of cycle of repetitive tasks without a break to a relatively high precision. It is therefore unsurprising that application to robotics has become one of the biggest focus areas for ILC. Many types of robot have been considered within the context of ILC including flexible manipulators, arc welding robots and gantry robots.

2.4.2.1 Simulation Study

Immediately after the ILC concept was first proposed, a number of simulation studies of ILC on robots were carried out. For example, Arimoto et al. (1984b), Arimoto et al. (1985), Bondi et al. (1988), Kawamura et al. (1985), Kawamura et al. (1988), Mita and Kato (1985), Togai and Yamano (1985), Wang and Cheah (1998). Most of these simulations used mathematical robot models in a software simulation environment (usually a technical mathematical software, e.g. Mathworks Matlab, MathCAD, SciLab, etc) to produce system outputs and compute tracking errors. This is a very simple and rapid way of verifying the algorithms and getting the associated performance. However, all simulation environments are approximations of the real system. It is impossible to consider all information from the real world, such as mechanical faults, manufacturing inconsistences, sudden noise or disturbances, measurement limitations and so on. Therefore, simulations can only represent the system performance in ideal conditions and to ensure the algorithm feasibility and performance in practice, real experiments should be undertaken. However simulation work can not be omitted as it will significantly reduce the risk of damaging facilities and the amount of parameter tuning time that is required.

2.4.2.2 Experimental Study

As simulation alone is not enough to thoroughly evaluate algorithm performance, practical research work is becoming more and more important. Longman (2000) and Elci et al. (1994) described the practical implementation of basic ILC on a seven axis industrial joint robot. The robot must follow a rapid trajectory movement, which maximises the interaction of the axes and induces non-linear dynamics such as Coriolis and centrifugal effects. The seven axes are each controlled separately as if the robot consisted of seven SISO systems. The non-linear effects and interaction between axes are simply treated as repeatable disturbances. Even with a simple learning controller, the tracking error was reported to be reduced by up to a factor of 1000 in approximately 12 trials. This demonstrates that ILC is well suited to controlling individual subsystems of a larger overall system, with little knowledge of how the subsystems interact with each other (Hwang et al., 1993). A series of practical stages has been completed by Ratcliffe et al. (2004), Ratcliffe et al. (2005a), Ratcliffe et al. (2006b), Ratcliffe et al. (2006a) to build and test a number of ILC controllers on a multi-axis gantry robot which provides considerable benchmarking data on ILC algorithm performance. The major work in this thesis saw the completion of the implementation of a number of newly developed algorithms on the gantry robot as well as further comparison. Full details are given in Chapter 4 and Chapter 5.

2.4.3 Rotary Systems

Rotational motion is generally influenced by position-dependent or time-periodic external disturbances. Therefore, control of rotary systems is a good candidate for ILC application. The major component in this area is the control of motors. Many related types of application are well suited for ILC algorithm implementation, a good example is that of conveyor chain systems. As mentioned previously, one of the typical features of ILC is that the system condition will be reset after each trial is completed. For robotic systems, this represents a challenge as the process of resetting needs to be very accurate in order to reduce the difference between initial conditions for each trial which might potentially degrade the overall performance. Currently, robotic systems generally set up a resetting tolerance so that once the robot moves into a region close to the reset point, the system is treated as having been reset. In the case of rotary systems, resetting is often far simpler and merely requires the system to stop, at which point the rotary position can reset

Barton et al. (2000) implemented a learning controller on a chain conveyor system in order to improve the tracking performance of this synchronising process. A hybrid controller was used which consisted of a learning controller coupled with a 3-term feedback controller. The hybrid controller used was arranged in a series formation in which the learning controller output was supplied to the feedback controller, and adjusted the reference trajectory in order to achieve accurate tracking. Due to the continuous nature of the task, the experimental application was, in fact, more suited to repetitive control.

2.4.4 Process Control Systems

Process control is another major application in the general control area. It has been widely applied to industrial tasks such as mixing materials, chemical reaction process and temperature control. As is common in manufacturing applications, most of the tasks are run repeatedly with the same settings or under the same payloads for thousands of trials. Therefore, ILC should yield considerable performance advantage in controlling such processes or tuning parameters in order to improve the productivity and quality of the products.

Havlicsek and Alleyne (1999), Xu et al. (1999), Gao et al. (2002), Mezghani et al. (2001), Mezghani et al. (2002) have applied ILC to chemical processes such as injection molding and other chemical reactors. Xu et al. (2002) reported a significantly improvement to trajectory tracking in water heating processes through use of iterative learning. Also ILC has been applied to batch process control applications as well (see, for example, Xu et al. (2001) and Xiong and Zhang (2003)). Xu et al. (2001) stated that the process is difficult to control due to the presence of modelling errors and time delays, but it was reported that near perfect tracking can be obtained under certain conditions.

2.4.5 Biological Applications

During recent years, ILC has become more and more popular in the biochemical industry and there are many successfully applications being reported. Choi et al. (1996) discussed the practical implementation of ILC to the fed-batch cultivation of Acinetobacter calcoaceticus RAG-1. This microorganism is used to produce emulsan, a chemical with good emulsifying properties for crude oil. Ethanol is an important factor for the microorganism as it is a carbon source aiding cell growth, cell maintenance and emulsan production. However, too much ethanol has a negative effect on the microorganism. Therefore the ethanol level must be carefully regulated. Choi et al. (1996) implemented a hybrid ILC controller and PI-type feedback controller to improve the control of ethanol supply to the microorganism over a number of batches. The learning controller was of basic P-type structure and uses the inverse plant model as the learning gain function. Results from practical implementation demonstrated that using the learning controller improved the yield from one batch by approximately 20% after only three trials.

A more recent implementation investigated by Mezghani et al. (2002) used an optimal ILC update on a chemical batch reactor which can be used to manufacture pharmaceutical products. The cost function used minimised the error while simultaneously limiting the change in the output over the next trial, thus restricting sudden changes in the control effort. Sliding window filtering was also performed to smooth the output and reduce the effect of extreme control effort values.

Freeman et al. (2008a) and Freeman et al. (2008b) developed a robotic workstation and applied ILC for the purpose of stroke rehabilitation using Functional Electrical Simulation (FES). The task presented to each patient was to track an elliptical trajectory while their arm was constrained to a horizontal plane. ILC was used to apply stimulation to the triceps muscle in order to assist their remaining voluntary effort in performing the task. Results from eighteen subjects were presented and showed that a high level of performance could be achieved using the proposed method.

2.4.6 Other Applications

Moon et al. (1996) developed an ILC controller capable of learning the periodic disturbances and reducing the tracking error in order to improve the control of optical disk drives. The controller was a hybrid design composed of a feedback controller and a learning controller operating in parallel. As it was not desirable for the learning controller to learn the non-periodic disturbances, a filter was added to the learning algorithm to prevent high frequency components of the signal being used in the learning process. Results showed a reasonable reduction in the periodic tracking error when the learning controller is implemented. Adaptive feedforward and repetitive control algorithms are now standard in most DVD and HDD drives (Chew and Tomizuka, 1990; Onuki and Ishioka, 2001; Zhou et al., 2004).

Dijkstra and Bosgra (2002b) and Dijkstra and Bosgra (2002a) applied ILC algorithms on a high precision wafer stage which is another typical device where ILC can be applied. Wassink et al. (2007) reported an implementation of a modified ILC algorithm on enhancing inkjet printhead performance. In Tousain and van Casteren (2007), a new ILC approach to the lamp current control problem was designed and implemented on the lamp driver. It was reported that ILC could achieve the requirements on the light output of Ultra High Pressure (UHP) lamps set by projection applications.

2.5 Problems with ILC

Although ILC can lead to a high level of performance in trajectory tracking applications, it is somewhat surprising that industry has not yet adopted this control method for all general applications involving repeating tasks. There are, however, some important reasons why that is so.

2.5.1 Initial Conditions

In the theoretical analysis or simulation of ILC, usually it is assumed that the initial conditions have been reset after each trial. But actually in practice it is not easy to

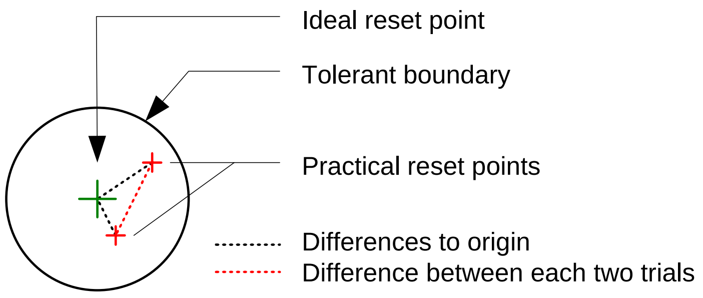


FIGURE 2.4: Practical resetting procedure in ILC

achieve this. As illustrated in Figure 2.4, in practice, it is usual to set up a region about the ideal core in which resetting is assumed to have been completed (commonly a circle centred on the ideal position). When the actual measured position is within the boundary, the system will be considered as completion of resetting. However, the resetting differences in resetting position between trials will have a detrimental influence and disturb the learning progress.

Heinzinger et al. (1989) considered a class of ILC law and proposed that the initial errors would not affect the convergence and stability of a system when the initial errors are repeatable. Also much work has been carried out to investigate these problems, including developing algorithms which are robust to initial state error (Lee and Bien, 1996; Chen et al., 1996, 1999).

2.5.2 Long Term Performance

A large number of papers, for example Huang and Longman (1996); Lewin (1999); Barton et al. (2000); Longman (2000), have maintained that a very significant problem with ILC implementation is long term stability or performance. Also many other papers such as Ratcliffe et al. (2005a) mention that this is one of the most important reasons why ILC has not been widely in industry. It has been reported both in simulation and practice that when a learning algorithm is implemented over a large number of trials, the tracking error will begin to grow up after it is initially reduced, and result in the eventual instability of the system. One frequently reported explanation for this from the frequency domain analysis (Huang and Longman, 1996) reported said that when ILC begins to operate, low frequency error becomes negligible, the effect of the high frequencies begins to become evident and the overall tracking error noticeably grows. To overcome this, many filtering techniques have later been introduced and designed in order to cut off unnecessary frequencies in the learning process (Plotnik and Longman, 1999; Longman, 2000). Moreover, Ratcliffe et al. (2005a) reported that an aliasing method was able to operate in conjunction with the ILC implementation used and gave a superior performance compared with the other filters used.

Usually, the normalised ratio of current error and previous error in either the frequency domain or time domain (Z-transformed or Laplace-transformed) ($||\frac{E_k}{E_{k-1}}||$) will be computed in order to evaluate the error evolution process. The error evolution can be derived using the following simple linear input/output mapping

$$Y(z) = G(z)U(z)$$

where $G(z)$ is the plant, $U(z)$ and $Y(z)$ are the z-transformed input and output vectors respectively. Now it is easy to derive the following equations (using P-type ILC described in (2.4) as an example, R denotes the reference trajectory and the plant is assumed to be invertible)

$$\begin{aligned}
\mathbf{Y}_k &= \mathbf{G}\mathbf{U}_k \\
\mathbf{E}_k &= \mathbf{R} - \mathbf{Y}_k \\
\mathbf{U}_k &= \mathbf{G}^{-1}(\mathbf{R} - \mathbf{E}_k) \\
\mathbf{U}_{k+1} &= \mathbf{G}^{-1}(\mathbf{R} - \mathbf{E}_{k+1})
\end{aligned}$$

According to the iterative learning law

$$\begin{aligned}
\mathbf{U}_{k+1} &= \mathbf{U}_k + \mathbf{L}\mathbf{E}_k \\
\mathbf{G}^{-1}(\mathbf{R} - \mathbf{E}_{k+1}) &= \mathbf{G}^{-1}(\mathbf{R} - \mathbf{E}_k) + \mathbf{L}\mathbf{E}_k \\
\mathbf{E}_{k+1} &= \mathbf{E}_k - \mathbf{G}\mathbf{L}\mathbf{E}_k \\
\frac{\mathbf{E}_{k+1}}{\mathbf{E}_k} &= 1 - \mathbf{G}\mathbf{L}
\end{aligned}$$

Here the learning matrix $\mathbf{L}$ could be an arbitrary linear operator L. Then the error evolution ratio can be also written as

$$\frac{\mathbf{E}_{k+1}}{\mathbf{E}_k} = 1 - L\mathbf{G}$$

Therefore, if the condition below is satisfied for all frequencies ($z = e^{j\omega T_s}$), the monotonic convergence of tracking error will be guaranteed.

$$||1 - L\mathbf{G}(z)|| < 1 \tag{2.24}$$

Where the norm $||\cdot||$ is for l_2 space. For the majority of cases in practice, this condition can be satisfied only within a certain frequency range (see for example Figure 2.5, results are taken from Cai et al. (2008)). Therefore the operator L must include a low pass filter with a cut off frequency

Filtering is an important part of ILC implementation in this thesis. Different filtering methods have been introduced and tested in later practical work (see from Chapter 3 to Chapter 5 for details). The choices of filters and filter settings are discussed further

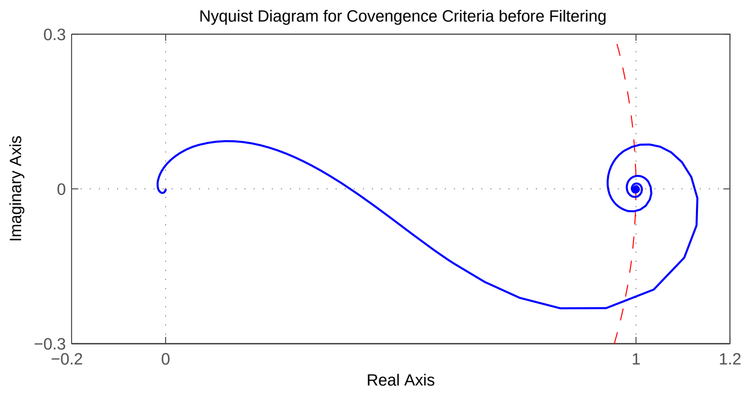


FIGURE 2.5: Nyquist plot of $||1 - L\mathbf{G}(z)|| < 1$ for an experimental facility.

in these chapters.

2.6 Summary

A review of ILC literature has been presented, and it has been concluded that the majority of ILC algorithms are based on a common fundamental principle, in which the input from the previous trial is used and a modification of the error sequence is added to develop the input for the next trial. In basic algorithms, the error modification is generally straightforward and can take the form of a single scalar gain. For model-based algorithms, the model is frequently used to derive an operator which is applied to the error vector, in order to generate more rapid error reduction. Techniques from a variety of other control methodologies such as optimal, adaptive and robust control have been applied to the ILC framework with the aim of improving the convergence rate and robustness to unknown and un-modelled disturbances.

ILC has been successfully applied to a number of industrial applications including control of robotic manipulators, rotary systems, chemical batch processes, bio-applications and many more. The use of ILC in these cases has greatly improved the performance of the plant being controlled. However, in general more experimental results are required.

Chapter 3

Reference Shift Algorithm

3.1 Introduction

As previously described in Chapter 2, limited progress has been made on ILC for non-minimum phase plants due to the significant challenges associated with the infeasibility of using the inverse of the plant model in the algorithm (Markusson et al., 2001). More recently, an electro-mechanical non-minimum phase experimental facility has been developed (Freeman, 2004) and used to compare the performance of some ILC algorithms in the presence of non-minimum phase characteristics. In this chapter, a new algorithm for ILC applied to non-minimum phase systems is developed and the experimental results obtained with it demonstrate that it can give significant performance benefits.

3.2 Non-minimum Phase Test Facility

3.2.1 Mechanical structure

The non-minimum phase test facility (see Figure 3.1) consists of a rotary mechanical system of inertias, dampers, torsional springs, a timing belt, pulleys and gears. The non-minimum phase characteristic is achieved by using the arrangement shown in Figure 3.2, where θ_i and θ_o are the input and output positions, J_r and J_g are inertias, B_r is a damper, K_r is a spring and G_r represents the gearing. This is originally designed through the conversion of a non-minimum phase electrical circuit (Freeman, 2004). A further spring-mass-damper system is connected to the input in order to increase the relative degree and complexity of the system.

A standard squirrel cage induction motor situated at the end of the test facility drives the load and is powered by an inverter which receives the control signal from the control computer. A 1000 pulse/rev encoder records the output shaft position and sends it back

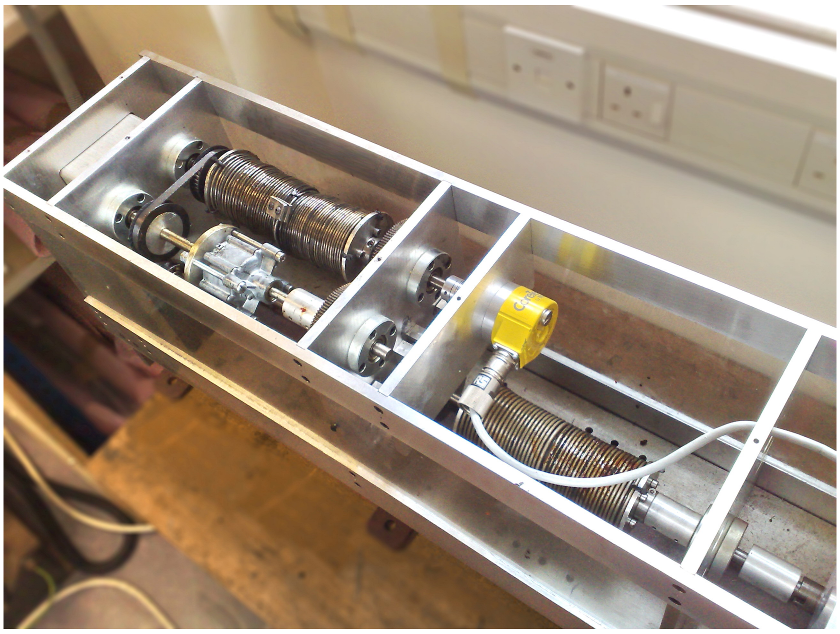


FIGURE 3.1: The non-minimum phase test facility

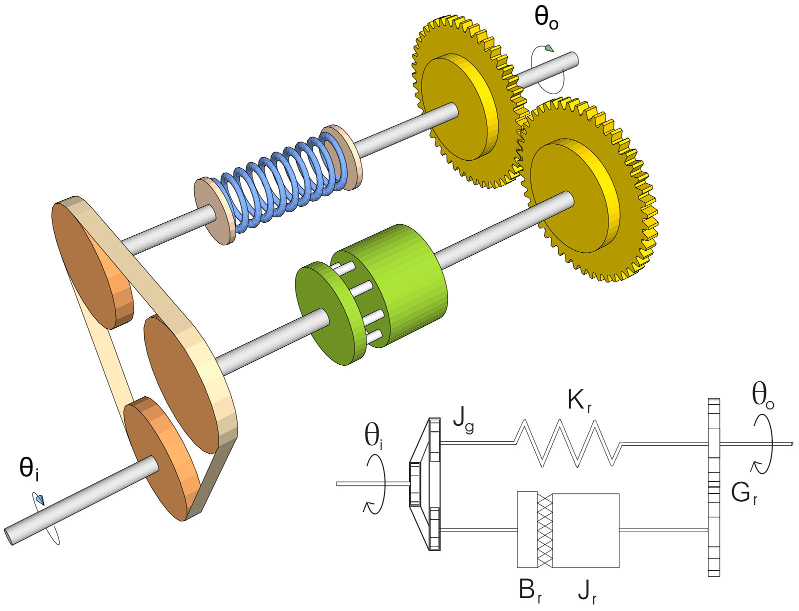
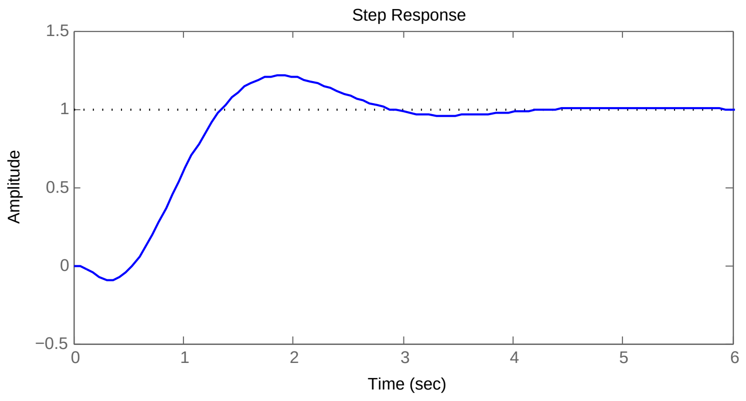


FIGURE 3.2: The non-minimum phase section.

to the data expansion card in real time. The whole mechanical system is controlled using a dSpace card which is a PCI based data input/output expansion card which is linked with popular mathematical and simulation software tool (Mathwork's Matlab/Simulink) which makes algorithm implementation more efficient and straightforward.



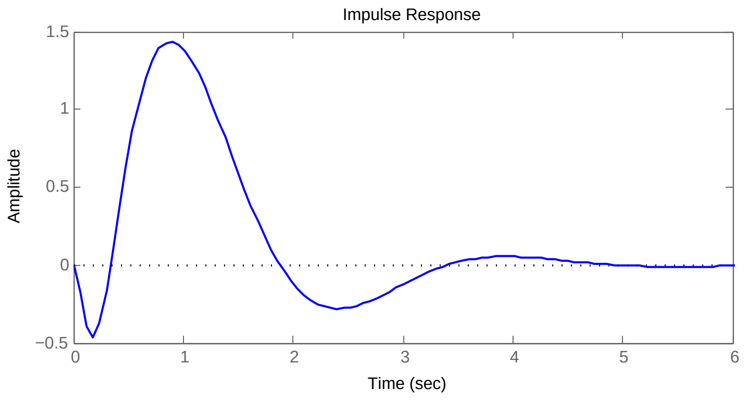


Figure 3.3: Step and impulse response of the non-minimum phase plant

3.2.2 System Modelling

The system model can be derived from differential equations modelling its individual elements separately which are then combined. Such a model would be based on very ideal assumptions and most of the coefficients may not be accurate enough to produce a reliable model. Therefore it is not able to provide a suitable mathematical model for practical research such as is required for the application of model-based algorithms and the performing of frequency domain analysis. The modelling of the non-minimum phase test facility therefore used frequency response tests and in this way the nominal continuous time plant transfer function has been identified by Freeman (2004) as

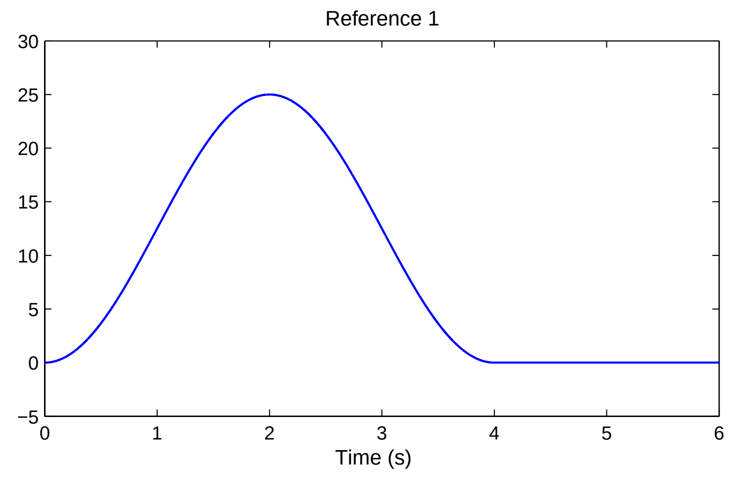
$$G_0(s) = \frac{1.202(4-s)}{s(s+9)(s^2+12s+56.2)} \tag{3.1}$$

It can be seen that there is one zero in the right hand side of the s-plane. The corresponding step response and impulse response for the close loop system are shown in Figure 3.3.

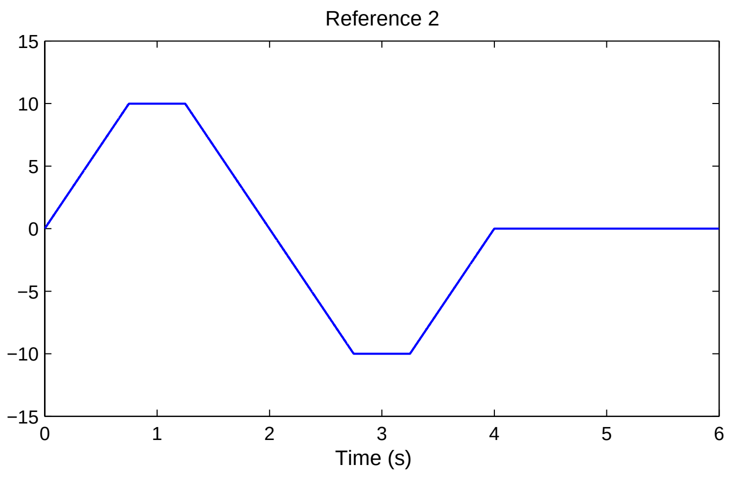
3.2.3 System Settings

A PID feedback loop around the plant is used in all the tests for the purpose of pre-stabilising the system since this has been found to produce superior results. The PID gains used are $K_p = 138$, $K_i = 5$ and $K_d = 3$. The resulting closed-loop system constitutes the system to be controlled. The sampling period, $T_s = 0.001$ (sampling frequency: 1000Hz), has been used in all experimental tests.

The test facility used a series of reference trajectories, of which two types of trajectory were most commonly applied; one is a sinusoid signal, the other is a trapezoidal sequence (see Figure 3.4(a) and 3.4(b) for examples of these). In order to make all testing results comparable, the results provided in later chapters are all produced using the same reference trajectories.



(a) Sinusoid reference.



(b) Trapezoidal reference.

FIGURE 3.4: Reference trajectories for the non-minimum phase plant

Since the test facility is a rotary system, there is no problem associated with resetting the system to initial conditions. There is a gap set up during each trial to ensure that the facility complete stops running before the next trial commences. On occasion, it is preferable to put a period of zero input before or after the reference signal which is designed to record more information from the output as there usually will be a significant output lag in such rotary systems. This also avoids a sudden 'jump' appearing in the signal which potentially disturbs the whole system performance.

3.2.4 Expression of Results

In order to carry out the research for experimentally benchmark the ILC algorithms, the comparison of results is an essential method. During all the experiments carried out on the experimental facilities, the following types of data are recorded for performance analysis and comparison.

- Input/Output/Error data vectors and system states.
  In ILC algorithms, input vector u_k, output vector y_k and error vector e_k ($k = 1, 2, 3...N$, where N denotes the total trial number in the test) are the most important source data for the analytical and comparative use. Sometimes, the system states x_k are also stored if the algorithm needs information from the states. A typical representation of the input/output/error data is using a 3-dimensional surface chart to clearly show the iterative learning progression (see for example Figure 3.5).

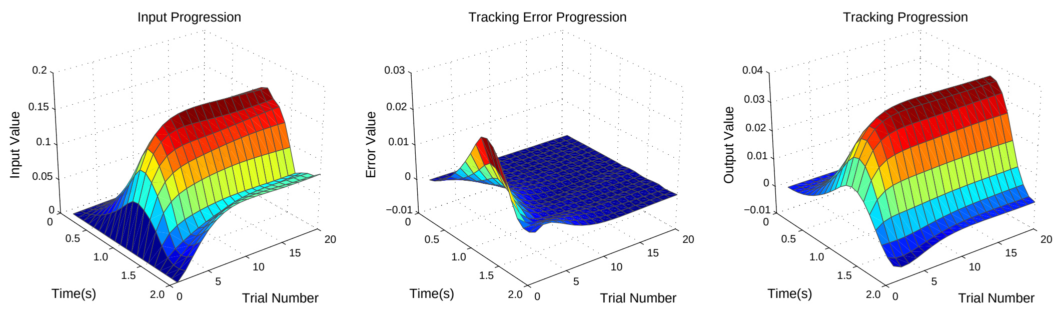


FIGURE 3.5: 3-D surface chart for showing ILC testing results

- Computed mean squared error (MSE) or normalised error (NE) for each trial. Mean squared error is computed using

$$\mathrm{mse}(e_k) = \frac{\sum_{t=1}^{T} [e_k(t)]^2}{T}$$

  where T is the length of the trial. A typical MSE plot uses a log scale in the vertical axis as ILC normally will reduce the tracking error to a very small value within a few trials. Using a log scale will enable a clearer vision of the difference between small values and will also maintain the overall trend of the learning process (see

Figure 3.6 as an example). This is a good method of showing the difference amount different algorithms.

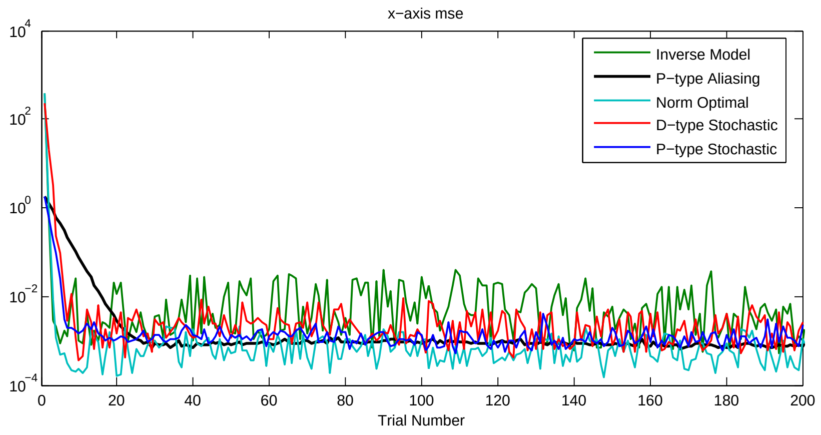


Figure 3.6: Mean squared error plots for showing ILC testing results

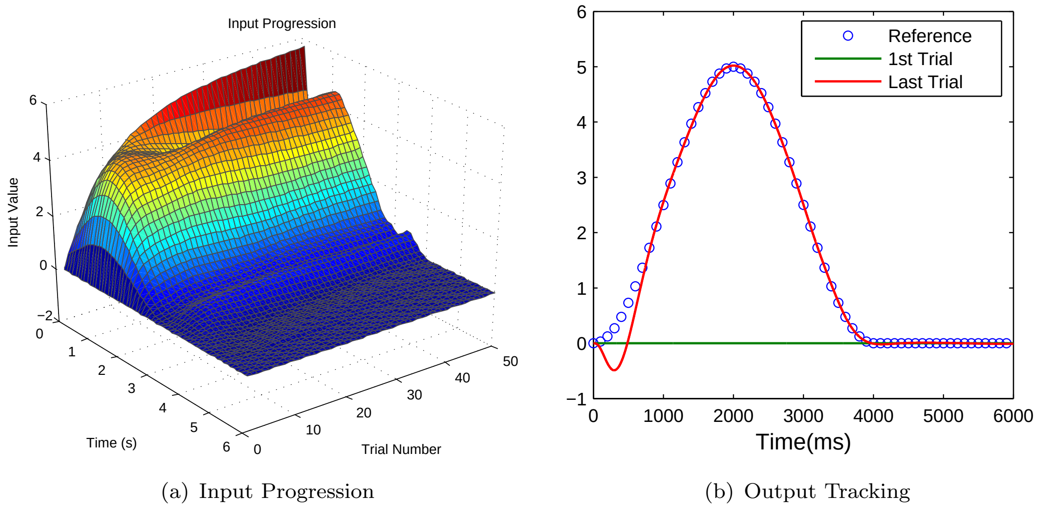
- For each particular algorithm, the specific parameters used are stored as well. For some advanced algorithms, these parameters will automatically adapt from trial to trial. It is useful to record the evolution of parameters for algorithm adjustment and improvement.

3.3 Previous Work using the Non-minimum Phase Facility

A large body of work has been previously conducted using the plant, including system modelling, evaluation of ILC algorithms and repetitive control schemes (Freeman et al., 2005b,a). There are a number of ILC algorithms which have been implemented on the plant. For example, basic P-type, D-type, delay type and later phase lead, norm optimal, predictive optimal. As mentioned in Chapter 2, the phase lead algorithm has been reported as a simple and effective algorithm for controlling the non-minimum phase plant. This algorithm can be expressed as:

$$u_{k+1}(t) = u_k(t) + Le_k(t+\tau) \tag{3.2}$$

Where $\tau \in \mathbb{Z}^+$ is the value for the phase lead. But according to Freeman et al. (2005b), the remaining problems faced when implementing this method are the choice of τ to use and the requirement that $e_k(t+\tau)$ not be truncated significantly as this leads to a large input appearing at the start of each trial when the controller is forced to immediately track a non-zero output. An example from a simple simulation will be used to illustrate this. Figure 3.7(a) shows the input progression of the phase lead algorithm in one simulation, which has got a large magnitude at the start of input signal. Also, the start of the tracking output is not followed well due to the effect of the non-minimum phase characteristic making the system initially go in the opposite direction for a short period (see Figure 3.7(b)). This problem can be solved by putting a period of zero at the beginning of a reference (Freeman et al., 2005b).



(a) Input Progression (b) Output Tracking

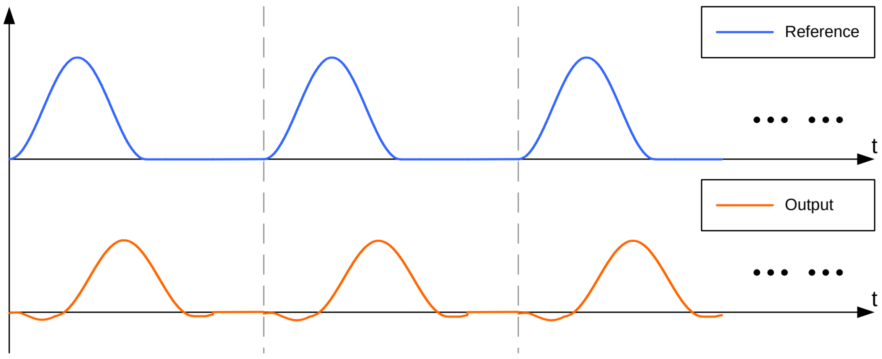
Figure 3.7: Simulation of the phase-lead algorithm

3.4 Reference Shift Algorithm

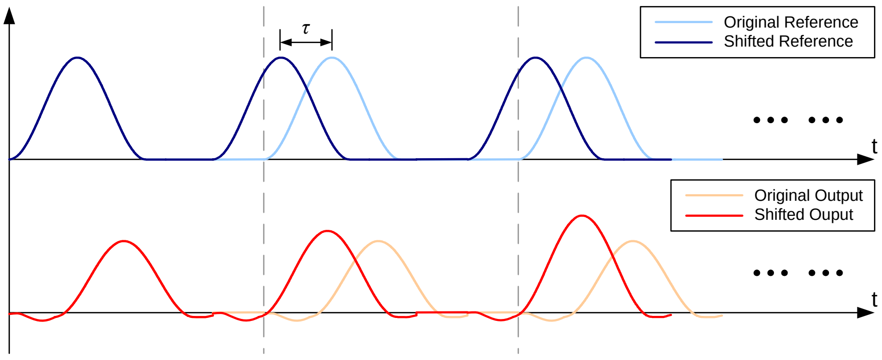
A new method called the reference shift algorithm has been developed and tested experimentally in order to improve some of the problems encountered by the phase-lead algorithm (Cai et al., 2007, 2008). The main idea of the reference shift algorithm is to let the algorithm itself look for the best phase shift value by aligning the measured output with the reference signal, and to then shift the reference by the computed value to get the next output. Figure 3.8(a) shows an example of the output shape for a non-minimum phase plant and it contains a time delay in the response. Figure 3.8(b) shows when reference is shifted by τ and then the system response can be aligned with the original desired reference.

3.4.1 Initial Approach

Due to the problems described before, the desired controller must incorporate both a method of reference shifting and a minimization scheme in order to be able to reduce



(a) System response without reference shift



(b) System response with reference shift

FIGURE 3.8: An illustration of the reference shift approach.

the tracking error in as few trials as possible. Consider first the update given by

$$u_k(t) = r'_k(t) + f_k(t) \tag{3.3}$$
$$r'_k(t) = r(t+\tau_k) \tag{3.4}$$
$$f_{k+1}(t) = f_k(t) + Le_k(t+\tau_k) \tag{3.5}$$
$$e_k(t) = r(t) - y_k(t) \tag{3.6}$$
$$\tau_k = \arg\min_{\tau\in[0,T]}\{\| r(t+\tau) - y_k(t) \|^2\} \tag{3.7}$$

where τ is the shift time of the reference signal, f_k is the feed-forward signal, the plant output is $y_k = r - e_k$ and T is the trial length. Figure 3.9 shows the controller structure of the proposed reference shift algorithm. It can be seen that the reference and the feed-forward signal are fed into the feedback loop together. From (3.5) and (3.7), it is clear that the reference signal will be shifted trial by trial in order to find a τ which is able to minimise the difference between control output and shifted reference.

From the block diagram it can also be seen that the reference shift algorithm interacts with the learning controller by updating the reference supplied to the plant input as well as that used in producing the error that is used in the learning update. Note also that a similar approach has been considered for the case of repetitive control in (Steinbuch, 2002). Moreover, an alterative approach has also been designed as well (see Figure

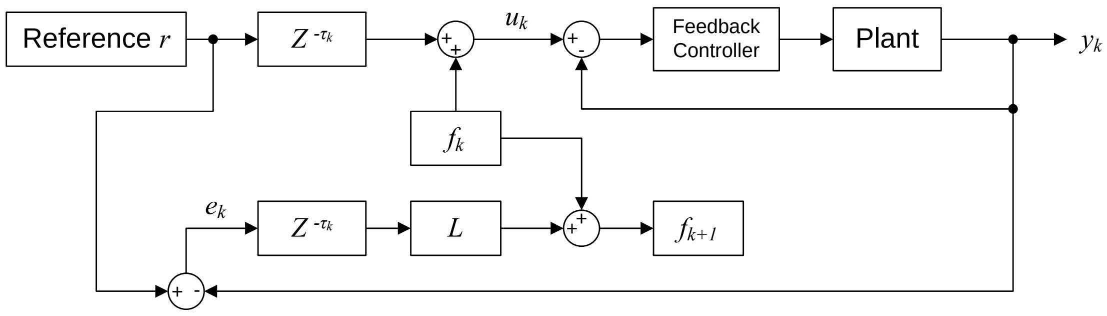


FIGURE 3.9: Controller structure of the reference shift algorithm.

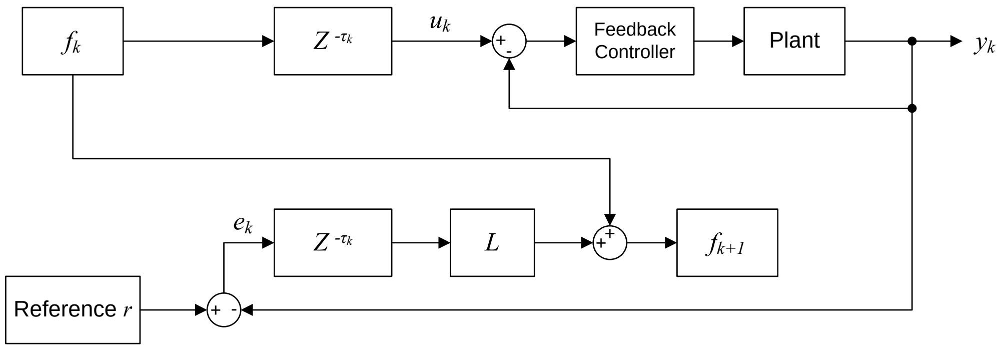


FIGURE 3.10: Alternative structure of the reference shift algorithm

3.10). The approach uses only the shifted control input instead of shifted reference. The related control input is computed as follow:

$$u_k = f'_k \tag{3.8}$$

$$f'_k(t) = f(t+\tau_k) \tag{3.9}$$

The updating of f_{k+1} is exactly the same as the original approach of from (3.5) to (3.7).

3.4.2 Initial Simulations

The reference shift algorithm has been simulated using Matlab/Simulink. The control plant in the simulation is the model of the non-minimum phase test facility described above. Equation (3.11) shows detailed input/output/error plots for the reference shift algorithm as well as the error evolution plots and the tracking performance for the first and last trial in the simulation. Figure 3.12 shows the error plots of both structures of proposed reference shift algorithm and it can be seen that the performances are good when used with low pass filters applied. Figure 3.13 shows the recorded shift procedure at the first trial. The system response has a output lag of around 1 second and when the reference was shifted, the system response can be align to the original desired reference.

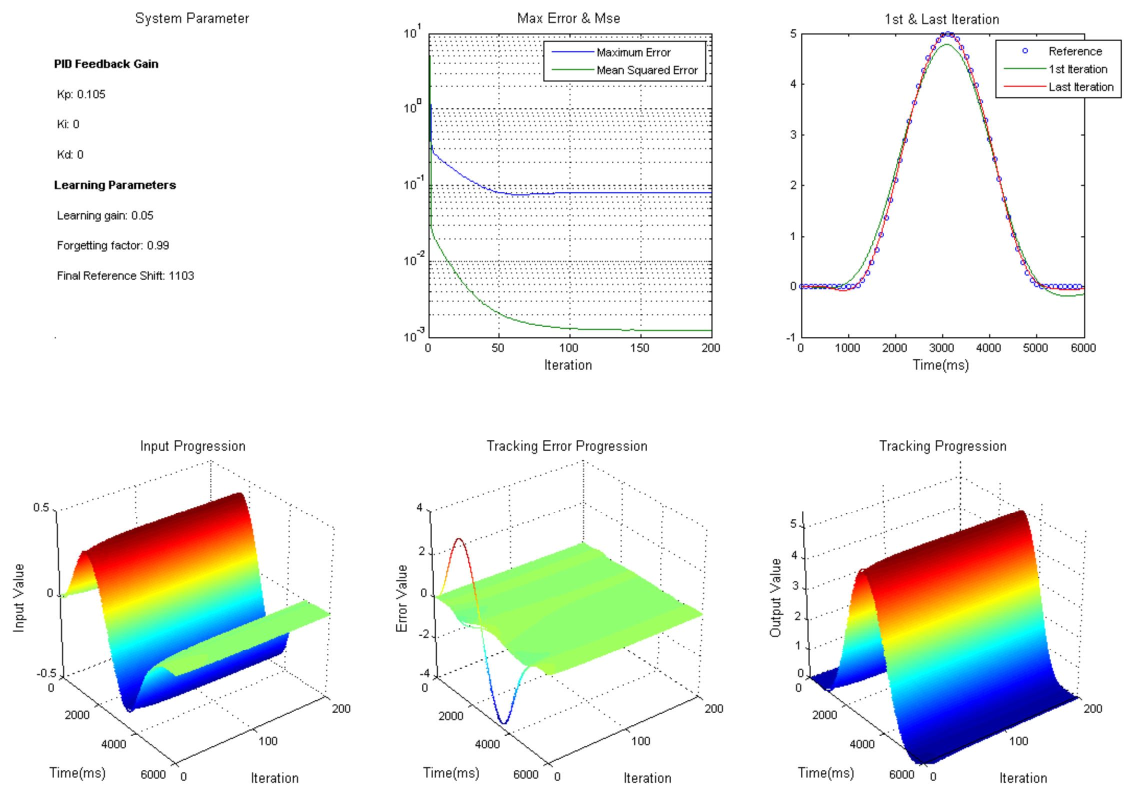


FIGURE 3.11: Initial simulation results of reference shift algorithm

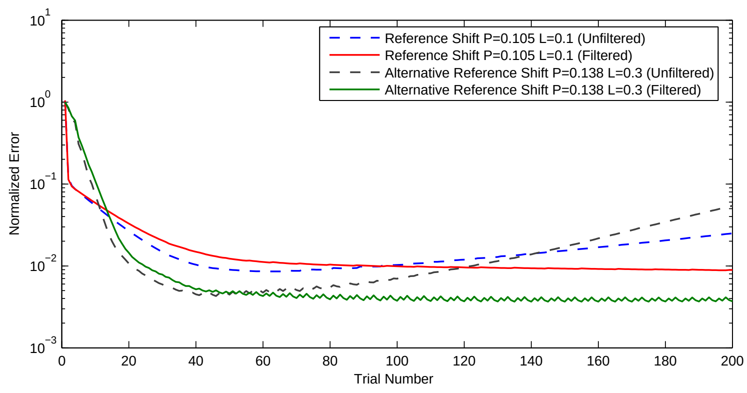


FIGURE 3.12: Initial simulation results of reference shift algorithm

In order to test the long term stability of the proposed algorithm, one simulation of 1000 trials was carried out and the error plots are shown in Figure 3.14. According to the error plots, there is no kind of trend indicating that the system would eventually go unstable. Therefore, the long term stability for the reference shift algorithm appears to be acceptable.

More simulations have been carried out and random disturbances were introduced into the control loop to simulate real experimental environments. Figure 3.15 shows the results of the algorithm dealing with disturbances, where no low pass filter is used. It can be seen that convergence is reasonably good. However, the long term stability can

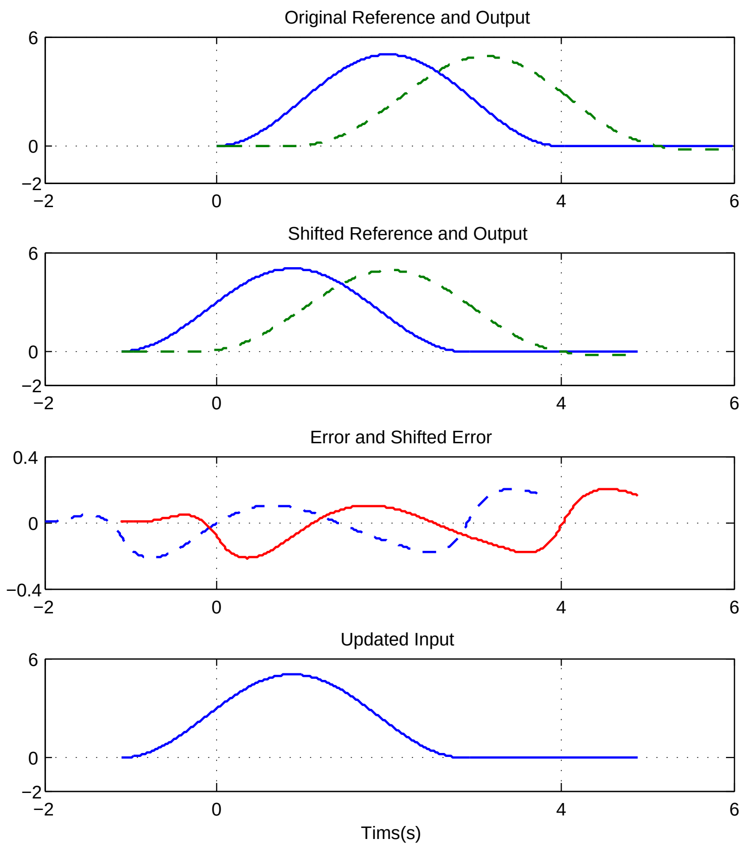


FIGURE 3.13: Initial simulation results of reference shift algorithm

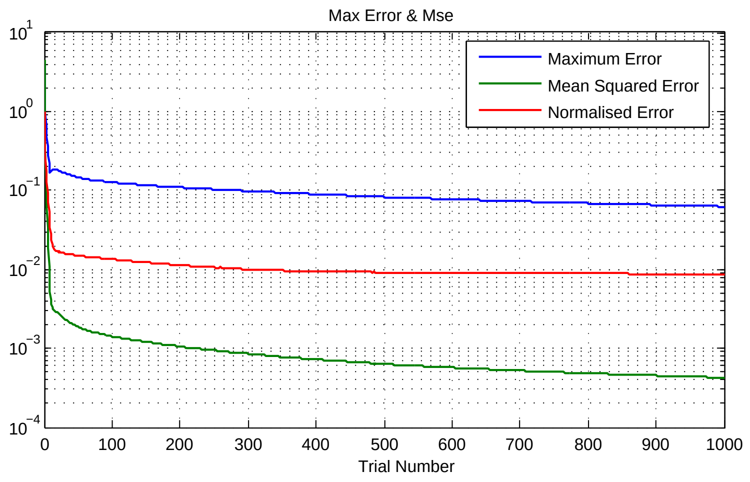
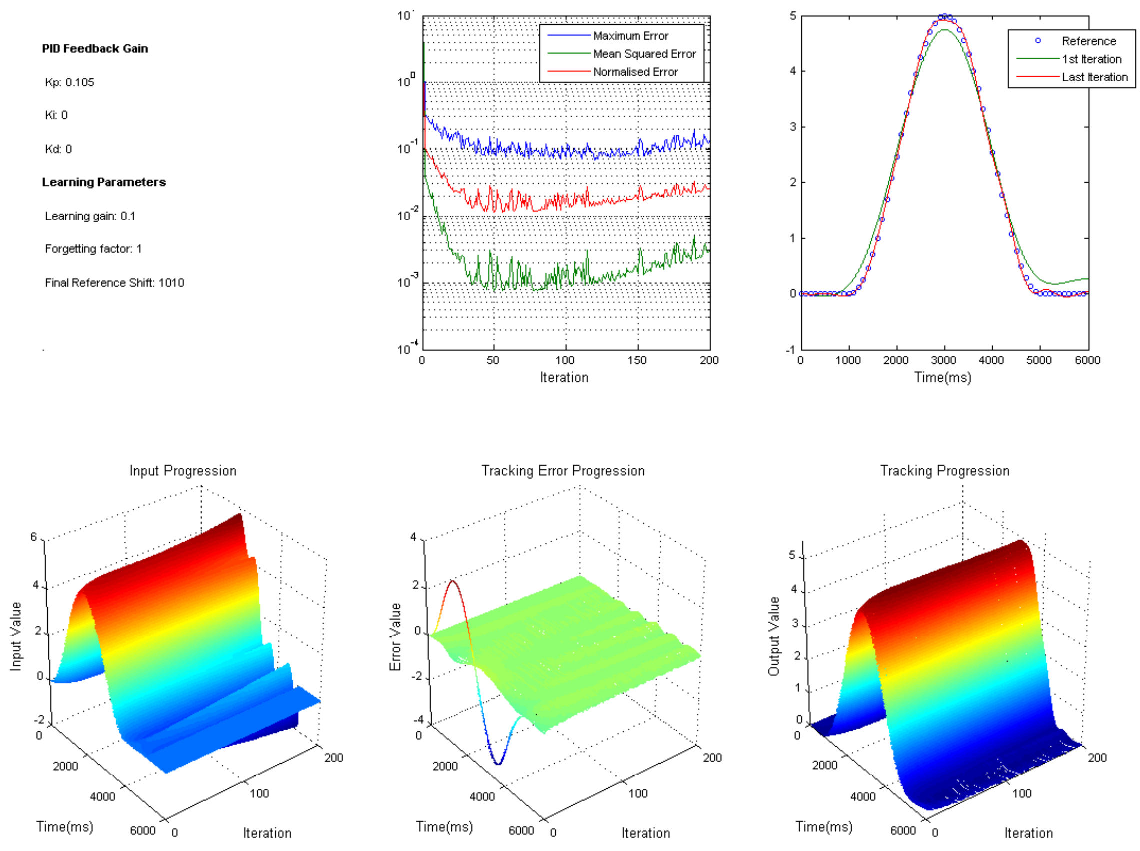


FIGURE 3.14: Initial simulation results of reference shift algorithm (long term stability test)

not be assured when there is no robust filter to remove high frequency noise. This is shown in the input mesh plot, that higher frequencies can build up and make the input become not unacceptable. Having isolated these problems, some analysis on the algorithm can be undertaken. The analysis focuses on the error evolution process and the system convergence rate in the frequency domain.



PID Feedback Gain

Kp: 0.105

Ki: 0

Kd: 0

Learning Parameters

Learning gain: 0.1

Forgetting factor: 1

Final Reference Shift: 1010

FIGURE 3.15: Initial experimental results of the reference shift algorithm

3.4.3 Convergence Analysis

The relationship between e_{k+1} and e_k must first be determined in order to examine the convergence properties. From the learning update law in (3.5), let e'_k be used to represent the error after shifting, that is $e'_k(t) = e_k(t+\tau)$. Also let r'_k be the shifted reference for the k^{th} trial, and P represent the open-loop transfer function of the feedback loop, then according to the block diagram of Figure 3.9:

$$\begin{aligned} P(r'_k + f_k) &= r - e_k \\ Pf_k &= r - e_k - Pr'_k \\ Pf_{k+1} &= r - e_{k+1} - Pr'_{k+1} \end{aligned} \tag{3.10}$$

then from the learning update law in (3.5), f can be eliminated:

$$\begin{aligned} r - e_{k+1} - Pr'_{k+1} &= r - e_k - Pr'_k + PLe'_k \\ e_{k+1} &= e_k - PLe'_k + P(r'_k - r'_{k+1}) \end{aligned} \tag{3.11}$$

Intuitively $P(r'_k - r'_{k+1})$ will be very small and potentially will go to zero as there will exist no difference between subsequent values of the calculated shift as the error reduces to zero. However it cannot be completely discounted and a revised algorithm is therefore necessary — once this has been introduced the convergence analysis will be resumed.

3.4.4 The New Algorithm

The previous algorithm described suffered from problems which arose due to the constraint of only applying an input of length T seconds. During trial k the input was defined over $t \in [-\tau_{k-1}, T-\tau_{k-1}]$ seconds and incorporated $r(t+\tau_{k-1})$ and $f_k(t)$. If τ_k, when it is calculated, is not equal to τ_{k-1}, part of $f_k(t)$ must be discarded to form the new update $f_{k+1}(t)$ which is defined over $t \in [-\tau_k, T-\tau_k]$. The discarding of data is then likely to produce an unsatisfactory transient response as the applied input is no longer smooth, and also increases the possibility that a design that satisfies the theoretical convergence conditions does not result in an acceptable implementation. In addition, no input is supplied during $t \in (T-\tau_k, T]$ which may lead to an unsatisfactory response in this interval.

The new algorithm resolves these problems by expanding the interval in which the input is applied so that no data is discarded in building each updated input. Figure 3.16 demonstrates the structure of the updated RSILC algorithm which, together with the reference shift law, can be described by

$$\begin{aligned}
u_{k+1}(t) &= r'_{k+1}(t) + f_{k+1}(t) & (3.12)\\
f_{k+1}(t) &= f_k(t-\tau''_{k+1}) + Le_k(t-\tau''_{k+1}+\tau_k) & (3.13)\\
r'_{k+1}(t) &= r(t-\tau'_{k+1}) & (3.14)\\
\tau'_{k+1} &= \max(\tau_k, \tau'_k) & (3.15)\\
\tau''_{k+1} &= \max(0, \tau_k - \tau'_k) & (3.16)\\
\tau_k &= \arg\min_{\tau}\{\| r'_k(t-\tau) + f_k(t-\tau) - y_k(t) \|_2^2\} & (3.17)
\end{aligned}$$

with $f_1(t) = 0$ and $\tau'_1 = 0$. Although appearing more complicated, the structure of

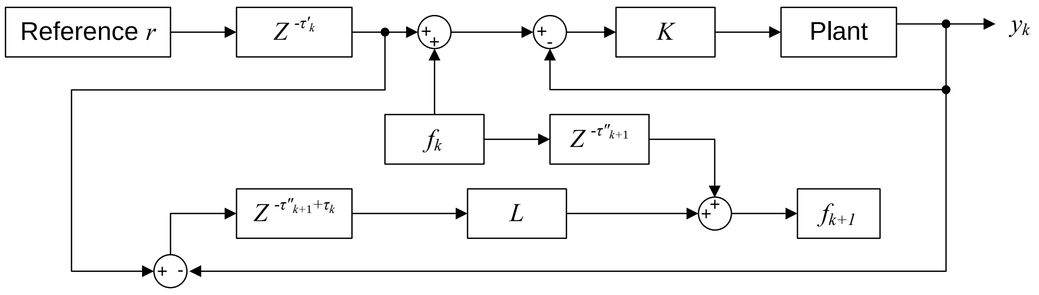


FIGURE 3.16: Controller structure of the revised reference shift algorithm.

this algorithm closely follows that of the previous one. It can again be seen that on the k^{th} trial the feed-forward signal, f_k, is summed with the shifted reference signal,

r'_k, to produce the control input, u_k, which is applied to the plant. The cost function (3.17) is identical to (3.7) since $r'_k(t-\tau_k) + f_k(t-\tau_k) = u_k(t-\tau)$ and so inherits the same motivation. The most important difference is that, in order to express the time over which each trial extends in a meaningful way, each signal is now defined over $t \in [0, T+\tau'_k]$. It can be readily seen that the value of τ'_{k+1} is the largest value of shift that has been calculated at the commencement of the $k+1^{th}$ trial, that is $\tau'_{k+1} = \max\{\tau_k, \tau_{k-1}, \ldots, \tau_0\} \geq 0$. Another important difference is that the reference, $r'_{k+1}(t)$, that is tracked by the output of the plant on the $k+1^{th}$ trial, is now a copy of the original reference shifted backward in time, with a period of being set to zero concatenated to the start. It is therefore given by

$$r'_{k+1}(t) \quad = \quad \begin{cases} 0, & t \in [0,\ \tau'_{k+1}) \\ r(s), & s = t - \tau'_{k+1}, \quad t \in [\tau'_{k+1},\ T + \tau'_{k+1}] \end{cases} \tag{3.18}$$

The error is only defined over the course of the original reference so that no truncation occurs when it is used in (3.13). It is thus given by

$$e_{k+1}(t) \quad = \quad \begin{cases} 0, & t \in [0,\ \tau'_{k+1}) \\ r'_{k+1}(t) - y_{k+1}(t), & t \in [\tau'_{k+1},\ \tau'_{k+1} + T] \end{cases} \tag{3.19}$$

It is then possible to simplify the algorithm by firstly observing that substituting $r'_k(\cdot) + f_k(\cdot)$ for $f_k(\cdot)$ on the right-hand side of (3.13) produces the additional term $r'_k(t - \tau''_{k+1}) = r_k(t - \tau'_k - \tau''_{k+1})$. Since

$$\begin{aligned} \tau'_k + \tau''_{k+1} \quad &= \quad \begin{cases} \tau'_k \text{ if } \tau_k < \tau'_k \\ \tau_k \text{ otherwise} \end{cases} \\ &= \quad \tau'_{k+1} \end{aligned} \tag{3.20}$$

this becomes $r'_{k+1}(t)$, which is also what is produced by substituting $r'_{k+1}(\cdot) + f_{k+1}(\cdot)$ for $f_{k+1}(\cdot)$ on the left side of (3.13). This means that (3.12) can effectively be absorbed by (3.13) to produce the single update

$$u_{k+1}(t) = u_k(t - \tau''_{k+1}) + Le_k(t - \tau''_{k+1} + \tau_k) \tag{3.21}$$

providing the initial condition is changed from $f_1(t) = 0$ to $f_1(t) = r(t)$. The algorithm has not been expressed in this form initially in order to emphasise the action of the reference shift, and to correspond with the form of the algorithm examined in Section 3.4.1.

In order to clearly illustrate the remaining action of the algorithm, Figure 3.17 (a) shows an example of the input, $u_k(t)$, and reference, $r'_k(t)$, that are used during the k^{th} trial.

The original reference, $r(t)$, has been shifted backward by τ'_k, and no data has been discarded in forming $u_k(t)$ which is applied over $t \in [0, T + \tau'_k]$. Once this trial is

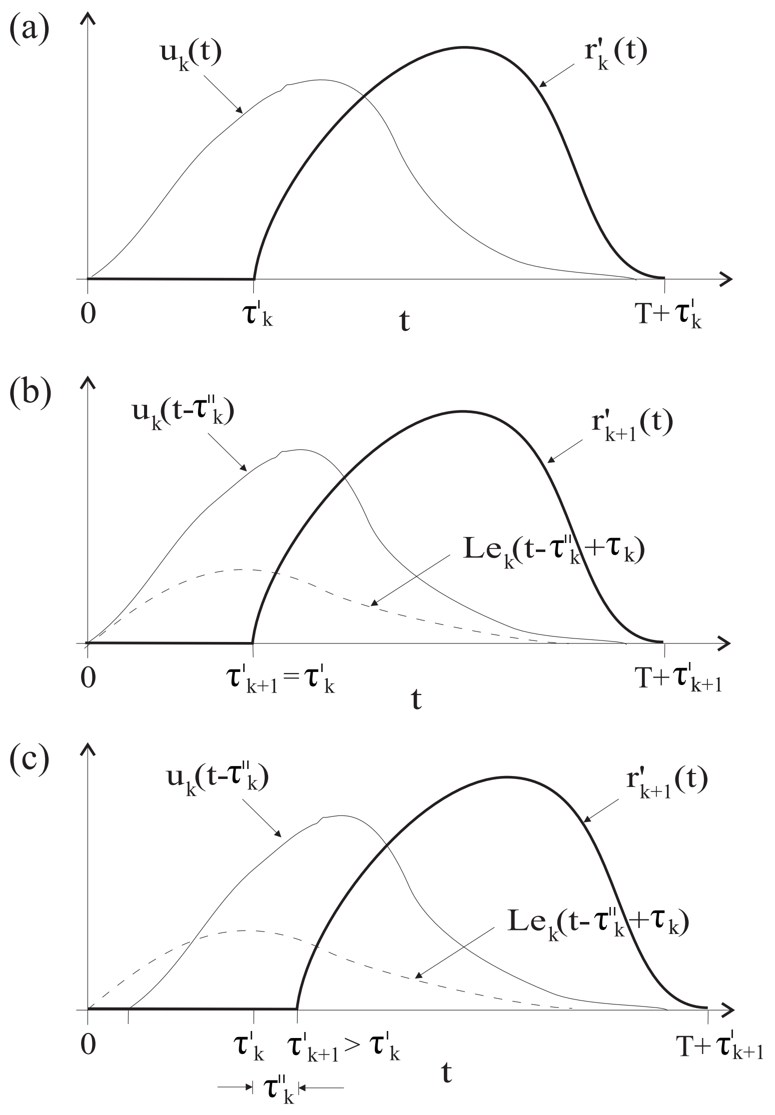


FIGURE 3.17: Update mechanism showing (a) signals during k^{th} trial, (b) the case where $\tau_k \leq \tau'_k$, and (c) the case where $\tau_k > \tau'_k$.

completed, τ_k is calculated using (3.17). Figure 3.17 (b) shows the components of the new input, $u_{k+1}(t)$, for the case that $\tau_k \leq \tau'_k$. Here the maximum shift, τ'_{k+1}, is set equal to the value used in the previous trial, τ'_k. The difference in these values, which, using (3.20), can be shown to be $\tau''_{k+1} = \tau'_{k+1} - \tau'_k$, is zero. This means that neither u_k nor e_k need to be shifted to keep their relative position with respect to the reference, and the latter is therefore merely shifted by τ_k and added to u_k to form the new input, u_{k+1}. Figure 3.17 (c) shows the alternative case where $\tau_k > \tau'_k$. Once shifted by τ_k, the error will start before any of the other signals, and so they all must be shifted backward in time so that it instead starts at $t = 0$. The value of the reference shift is accordingly changed from τ'_k to τ_k, and both components used to build the new input, u_{k+1}, are shifted by the difference, τ''_{k+1}, in order to maintain their relative temporal position with respect to the reference.

The substitution $s = t - \tau_k$ can be used and then restricted in order to ensure that the original reference is tracked over the intended range, that is $s \in [0, T]$.

Now the relationship between e_{k+1} and e_k must be obtained in order to examine the algorithm's convergence properties. Manipulation of (3.21) results in

$$e_{k+1}(t) = e_k(t - \tau''_{k+1}) - LPe_k(t - \tau''_{k+1} + \tau_k) \tag{3.22}$$

in which P is the plant being controlled. Applying the z-transform gives

$$E_{k+1}(z) \;=\; \left(1 - LP(z)z^{\tau_k/T_s}\right) z^{-\tau''_{k+1}/T_s} E_k(z) \tag{3.23}$$

where the z-dependence of P has been stated explicitly. Hence we have monotonic convergence (see also Steinbuch and van de Molengraft (2000); Longman (2000)) provided

$$\begin{aligned} \left|\left(1 - LP(e^{j\omega T_s})e^{j\omega\tau_k}\right) e^{-j\omega\tau''_{k+1}}\right| &= \\ \left|1 - LP(e^{j\omega T_s})e^{j\omega\tau_k}\right| &< 1 \end{aligned} \tag{3.24}$$

and the convergence properties for any given frequency of interest can be studied.

If it is assumed that $r'_k(t) + f_k(t) - y_k(t)$ is periodic, a reference dependent assumption which is usually justified in practice, we can use Parseval's relation to write the discrete form of the cost (3.17) as

$$\begin{aligned} &\| \, r'_k(t - \tau_k) + f_k(t - \tau_k) - y_k(t) \, \|_2^2 = \\ &\textstyle\sum_{n=1,2\ldots N} \left|r'_k(n - \tau_k/T_s) + f_k(n - \tau_k/T_s) - y_k(n)\right|^2 = \\ &N \textstyle\sum_{k=1,2\ldots N} |a_k|^2 \end{aligned} \tag{3.25}$$

where a_k are the Fourier Series coefficients of

$$e^{-j\omega\tau_k}\left(R'_k(e^{j\omega T_s}) + F_k(e^{j\omega T_s})\right) - Y_k(e^{j\omega T_s}) \tag{3.26}$$

and $N = T/T_s + 1$. Since

$$\begin{aligned} e^{-j\omega\tau_k}\left(R'_k(e^{j\omega T_s}) + F_k(e^{j\omega T_s})\right) - Y_k(e^{j\omega T_s}) &= \\ e^{-j\omega\tau_k}U_k(e^{j\omega T_s})\left(1 - e^{j\omega\tau_k}P(e^{j\omega T_s})\right)& \end{aligned} \tag{3.27}$$

the cost given by (3.17) is equivalent to

$$\begin{aligned} \min_{\tau_k} \textstyle\sum_{\omega=\frac{2\pi}{N},\frac{4\pi}{N}\ldots 2\pi} \big|e^{-j\omega\tau_k}U_k(e^{j\omega T_s})\cdot& \\ \left(1 - e^{j\omega\tau_k}P(e^{j\omega T_s})\right)\big|^2& \end{aligned} \tag{3.28}$$

Therefore the minimisation objective here can be written directly in terms of the monotonic convergence criterion with $L = 1$ as

$$\min_{\tau_k} \sum_{\omega=\frac{2\pi}{N},\frac{4\pi}{N}...2\pi} \left|U_k(e^{j\omega T_s})\right|^2 \left|1 - LP(e^{j\omega T_s})e^{j\omega\tau_k}\right|^2 \tag{3.29}$$

and $\left|U_k(e^{j\omega T_s})\right|$ can be regarded as a weighting function for the frequency-wise minimisation of the monotonic convergence criterion. The appearance of u_k in the weighting function may not seem a natural choice if one were sought in the design stage prior to the implementation of the algorithm. In terms of practicality, however, it is a most natural selection as there is no value to be gained in satisfying the monotonic convergence criterion for frequencies that are not present in the plant output, and the importance of satisfying the criterion is strongly related to the magnitude of each frequency present in the plant output, and this too is dictated by u_k.

The criterion restricts $LP(e^{j\omega T_s})e^{j\omega\tau_k}$ to lie in the unit circle centred on +1. Since the phase of a time-delay approaches zero at low frequencies, this effectively restricts the phase of $P(\cdot)$ to be less than 90° in magnitude at low frequencies.

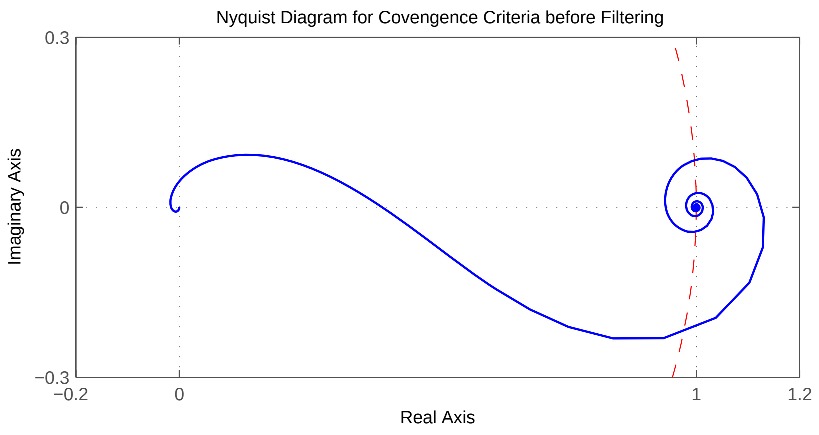
This frequency-domain condition is well known in the ILC literature and forms the basis for the stability analysis of many control algorithms (see for example Padieu and Su (1990)). Moreover Longman (2000) has shown that in practice it can produce good learning transients throughout each trial. Since the present algorithm has been especially formulated with the reduction of these transients in mind, satisfying the criterion should also ensure the property of monotonic convergence exists over as large a section of each trial as possible. Experience has shown that in many applications of ILC, convergence and the transient performance along the trials can be conflicting requirements.

3.4.5 Filtering

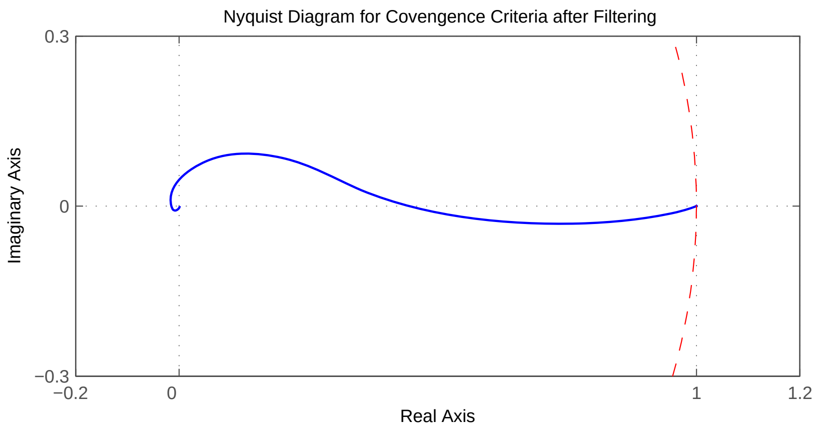
Figure 3.18(a) shows a typical Nyquist plot of $[1 - LP(z)z^{\tau_k/T_s}]$ for the unfiltered algorithm in which τ_k is set at 1.1. Only a small section of the plot is outside the unit circle centred at the origin and therefore does not satisfy the criteria for monotonic convergence. A filter must therefore be designed to ensure monotonic convergence by removing higher frequency components.

The filter can be applied to both the error signal and the input signal, the only difference being whether the demand itself is filtered. If the filter is applied to the error signal, the monotonic convergence criteria given by (3.24) becomes

$$\left|1 - L(e^{j\omega T_s})P(e^{j\omega T_s})e^{j\omega T_s - \tau_k/T_s}\right| < 1 \tag{3.30}$$



(a) Before filtering.



(b) After filtering.

FIGURE 3.18: Nyquist plot for the convergence criteria before and after filter applied.

where L is now a function of z. Initially, a low-pass FIR filter was designed to reduce the plot at those frequencies which do not satisfy the monotonic convergence criteria. When using FIR filters, a sensible approach is to first design a causal linear phase filter with the required magnitude properties and an odd number of coefficients which are symmetrical about a central term, $a_M z^{-M}$. Then this filter can be multiplied by z^M to create a non-causal filter which has the zero-phase characteristic.

Alternatively an IIR filter can be used in a procedure which requires fewer coefficients and consequently permits the speed of the computational process to be increased. The error signal is filtered using an IIR filter with a gain which is half the desired value at each frequency. The resulting output is then time-reversed producing another signal which is then fed into the same filter again. By reversing the output again, the required filtered signal is achieved.

After filtering with the non-casual filter, the resulting Nyquist plot of $1 - LP(z)z^{\tau_k/T_s}$ is shown in Figure 3.18(b). The condition for monotonic convergence is satisfied using this fixed value of τ_k. Nyquist diagrams corresponding to different τ_k values using the non-causal filter are shown in Figure 3.19 and it can be seen that varying τ_k excessively means that monotonic convergence will not be achieved.

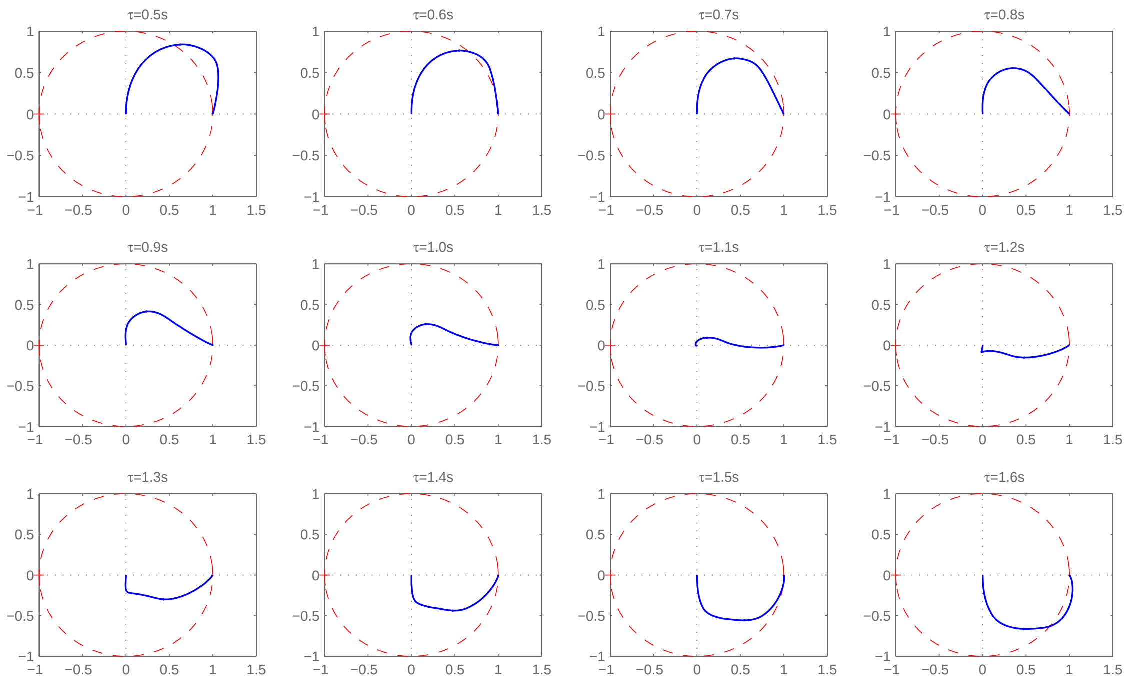


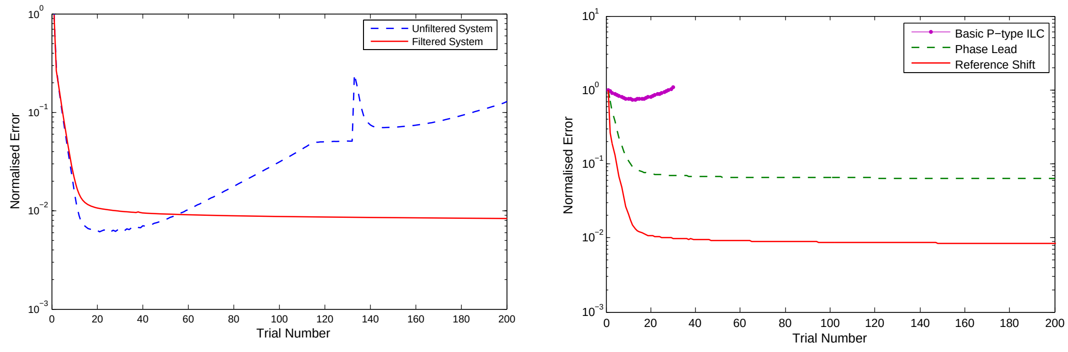
FIGURE 3.19: Nyquist plot for the convergence criteria after filtering of different τ_k.

3.4.6 Simulations

A series of simulations have been run in order to test the performance of the algorithm and also to compare its performance with that of other algorithms using the same plant model and reference trajectory. The two reference trajectories used in all the tests have been shown in Figure 3.4(a) and 3.4(b).

Figure 3.20(a) presents simulation results of the tracking performance for both the unfiltered and filtered systems. The plot shows the normalized error recorded over 200 trials. It can be seen that, after initial convergence, the errors using the unfiltered algorithm start to increase after about 50 trials and the system becomes unstable. Similar simulations using smaller values of gain have been performed but this merely postpones the effect of this error increase. However, it can also be observed that the use of a zero phase IIR filter ensures good performance.

The purpose of the reference shift algorithm is to achieve faster convergence and smaller final error than the other learning laws that have been applied. To examine its success, simulations have been performed in order to test its performance against several previously implemented algorithms. To achieve this, a model of the plant transfer function has been generated using a method in which a linear frequency response is fitted to experimental data, it is, however, a significant advantage that the reference shift algorithm does not require any such plant model. Simulations have been performed in order to compare the performance of the different algorithms, which include P-type, phase-lead as well as the reference shift algorithm. The results are shown in Figure 3.20(b) and clearly show the performance advantage gained using the current algorithm. The proportional learning gain L in the P-type algorithm was set to -0.01 which generated the simulation results. Other learning gains were tried as well but they could only produce worse results with faster divergent speed. The parameter of phase lead algorithm τ = 1500ms was stated as the optimal value in Freeman (2004). Here the optimal value of τ and the learning gain $L = 0.1$ are used but there is not a period of zeros points in the initial part of the reference which made the performance of phase lead algorithm worse than it appeared in the previously written literatures.



(a) Filtering comparison. (b) Comparison of different algorithms.

FIGURE 3.20: Numerical simulation for the reference shift algorithm.

3.4.7 Experimental Results

The reference shift algorithm has been implemented on the non-minimum phase experimental test facility in order to assess the convergence properties of the algorithm in practice and compare its performance with that of the former algorithms. Mean squared errors and individual trail performance are recorded for the evaluation of the algorithm. Figure 3.21 shows the error results and the tracking performance of a 200 trial experiment using reference shift ILC for tracking the sinusoid reference. It can be seen that the error reduces rapidly and remains at a very low value for the remainder of the test. Furthermore, there is no indication that the error will go on to increase if the test were to be continued. The plot of the error results for tracking the second reference is shown in Figure 3.23. Due to the abrupt transition in the second reference, instead of the smooth wave in the first reference, the mean squared error and normalised error are greater than that shown in Figure 3.21.

The plant output for the 200th trial of the test for both references is shown in Figures 3.22 and 3.24, in which the time variable ‘s’ has been used so that the output, whilst

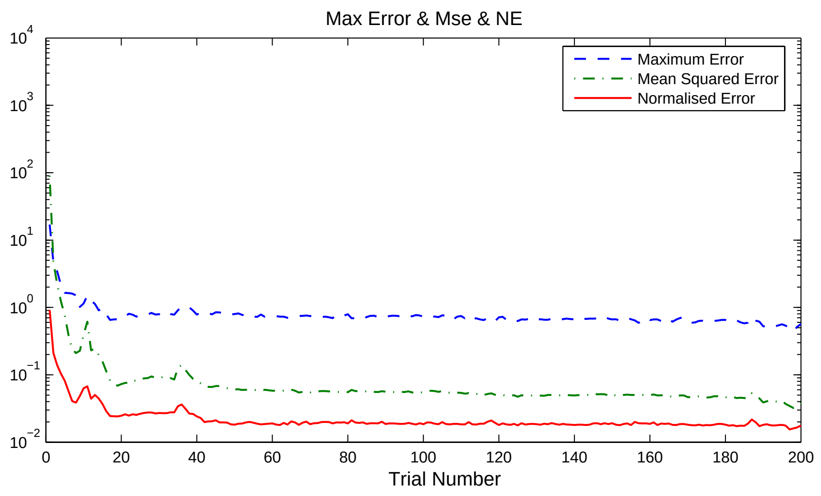


FIGURE 3.21: Error results plot of reference shift algorithm for the sinusoid reference ($K_p = 0.138, K_i = 0.005, K_d = 0.003, L = 0.3$).

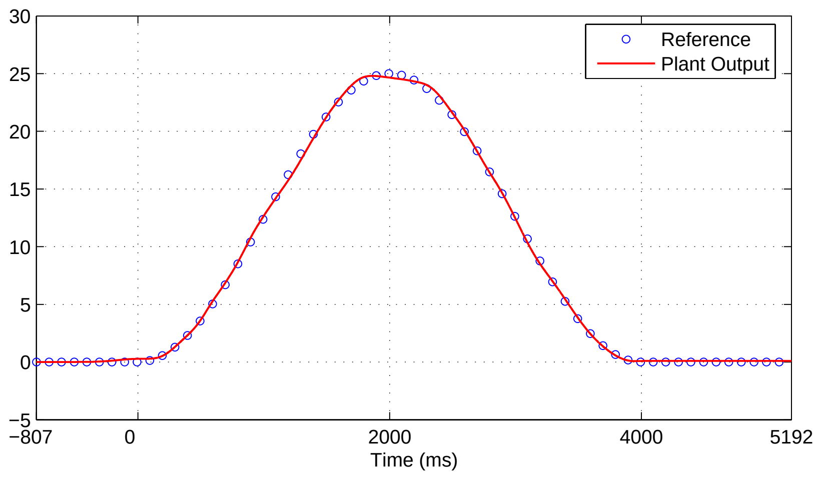


FIGURE 3.22: The tracking trace of the 200th trial for the sinusoid reference in one experiment.

tracking the shifted reference in the plot, also follows the original reference over its range when it is given as a function of the time variable 't', as discussed in the previous section. The values of the time shift in this case are 0.807s and 0.977s. It should be noted that the y axis of the error plots are in log scale to make differences clearly visible.

In order to test the long term stability for the reference shift algorithm, experiments of 400 trials for both references have been performed and the error results and tracking progression are shown in Figures 3.25 and 3.26. These plots indicate that the long term performance of the reference shift algorithm is satisfactory as the error shows no signs of increasing. In Figure 3.26, due to some unknown plant disturbances, the error rapidly grows but quickly reduces again after only a few trials, which also shows the re-learning ability of the algorithm.

Figure 3.27(a) and 3.27(b) show the evolution of τ_k over 400 trials for both references,

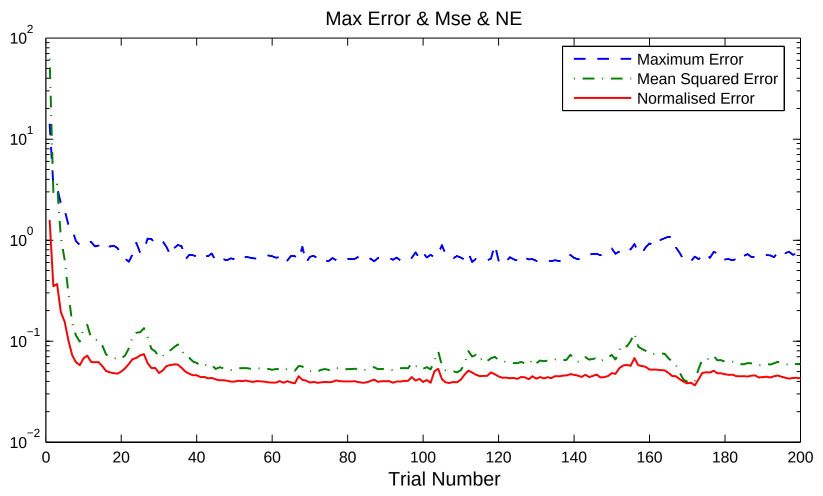


FIGURE 3.23: Error results plot of reference shift algorithm for the trapezoidal reference ($K_p = 0.138, K_i = 0.005, K_d = 0.003, L = 0.3$).

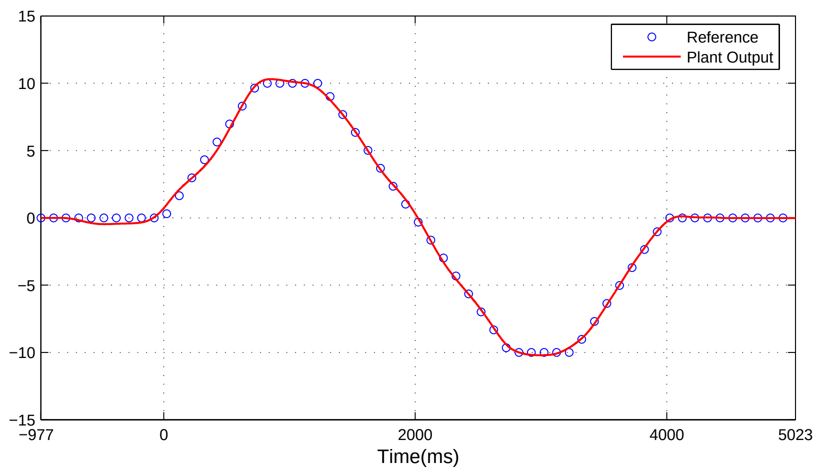


FIGURE 3.24: The tracking trace of the 200th trial for the trapezoidal reference.

illustrating that it rapidly converges to a value which does not alter significantly over the course of the experiments.

3.4.8 Performance Comparison

The experimental results can be directly compared with the results in previous algorithm implementations as all experimental conditions are the same, and the algorithm was operating using the same reference trajectories as previous tests. The results are compared with those of the phase lead ILC algorithm and the norm-optimal ILC (NOILC) algorithm. The phase lead algorithm has been reported as a simple and effective approach when applied to the non-minimum phase plant. The NOILC algorithm has received considerable attention in the ILC literature due to its mature theoretic basis (Amann et al., 1996b) but is considered here as a possible alternative to RSILC. In NOILC (stated

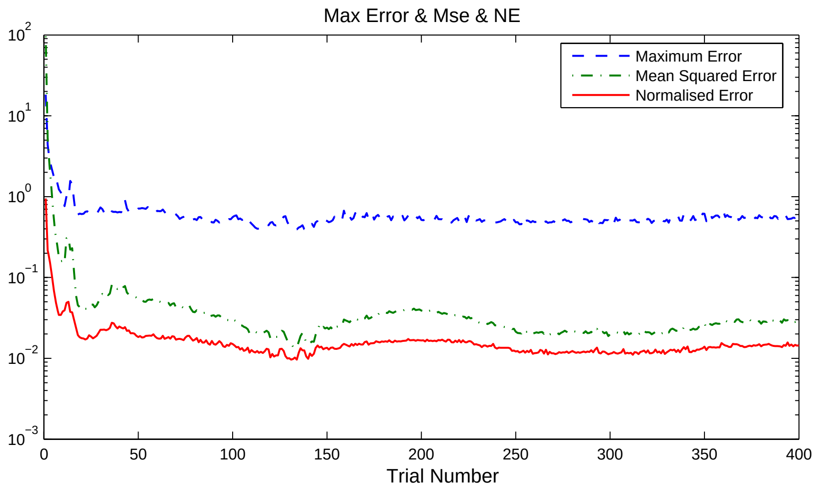


FIGURE 3.25: Long term performance test of reference shift algorithm for the sinusoid reference ($K_p = 0.138, K_i = 0.005, K_d = 0.003, L = 0.3$).

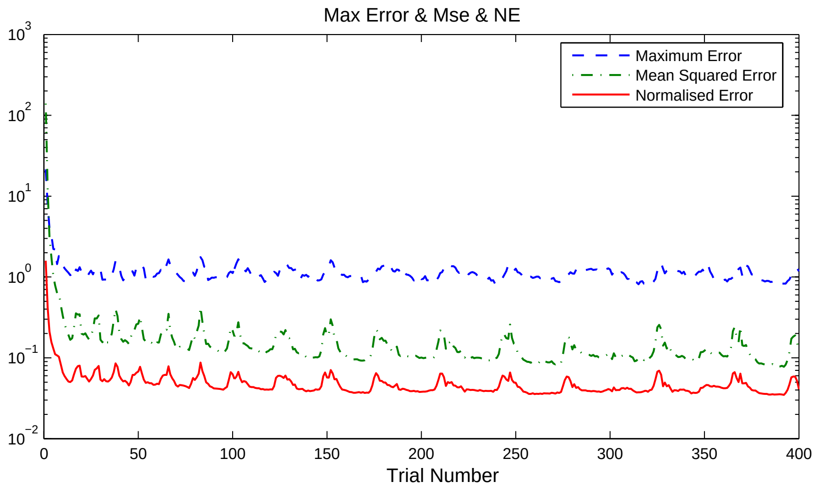


FIGURE 3.26: Long term performance test of reference shift algorithm for the trapezoidal reference ($K_p = 0.138, K_i = 0.005, K_d = 0.003, L = 0.3$).

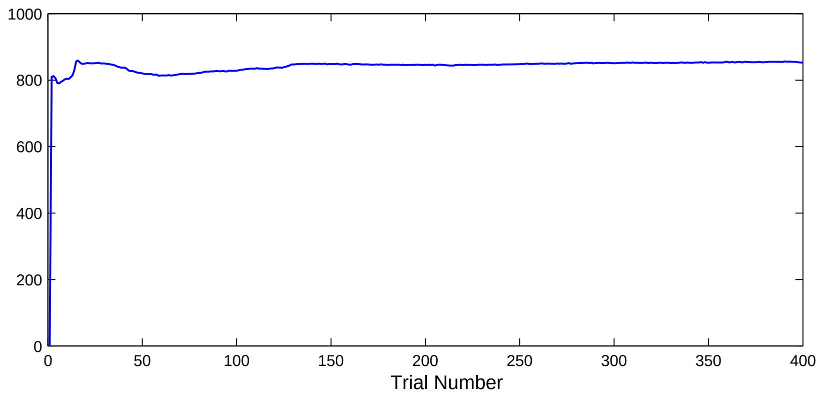
here in its abstract Hilbert space setting), the input on the $(k+1)^{th}$ trial is chosen to minimise

$$J_{k+1}(u_{k+1}) = \|e_{k+1}\|^2_{\mathcal{Y}} + \|u_{k+1} - u_k\|^2_{\mathcal{U}} \tag{3.31}$$

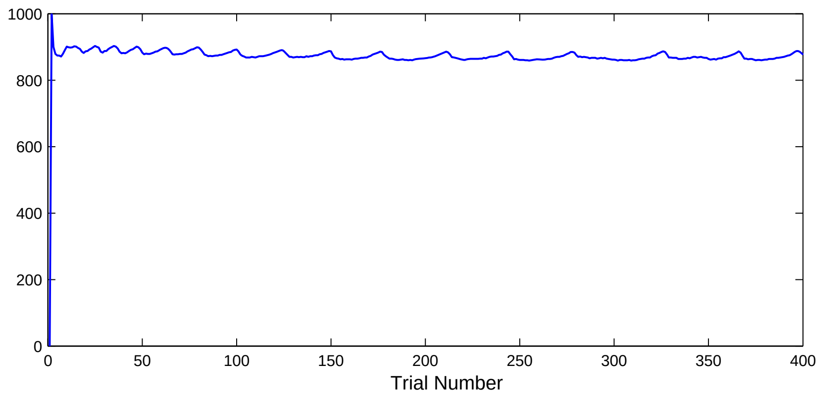
where $\mathcal{U}$ and $\mathcal{Y}$ denote the input and output function spaces respectively. The non-causal solution is

$$u_{k+1} = u_k + P^* e_{k+1} \tag{3.32}$$

in which P^* is the plant adjoint, is transformed into a causal implementation incorporating both feedback and feedforward actions. (The use of $u_{k+1} - u_k$, i.e. the error between the control input signals on two successive trials provides some control over the possibility that the need to reduce a large error will result in a large control signal demand).



(a) Evolution of τ_k for the sinusoid reference.



(b) Evolution of τ_k for the trapezoidal reference.

FIGURE 3.27: Evolution of τ_k for both reference in experiments.

To apply NOILC, the route taken was to build a minimum state-space model from the plant transfer function of (3.1) and specialize (3.31) by a linear quadratic cost in e_{k+1} and $u_{k+1} - u_k$ with weighting matrices Q and R respectively and a terminal end of trial quadratic constraint with weighting matrix F. For this plant, Q and R are both scalar and were set to $Q = 1$ and $R = 10$, respectively.

The algorithm is especially appropriate to the present system as it is known to perform well with non-minimum phase plants (Amann and Owens, 1994). Figure 3.28 shows error results comparing the reference shift algorithm to the norm optimal algorithm and to the phase-lead algorithm. The parameters used in the phase lead algorithm $\tau = 1500$ms and $L = 0.1$ were taken from Freeman (2004) which was stated to be the optimal values. Firstly, compared with the phase-lead algorithm, the reference shift algorithm out-performs it in not only convergence speed but also in the ability to converge to a constant 'final' error. Here the phase-lead algorithm is actually performing on a modified reference signal which contains a number of samples with zero value in the initial part. This eliminated the effect of poor learning performance in the initial part as previously described in the beginning of this chapter. In terms of the performance when compared with the norm optimal approach, the reference shift algorithm produces errors which are bounded below a small value and it achieves similar convergence with less fluctuation in tracking error. The error typically reduces to a very low value within 10 trials and remains close to that value.

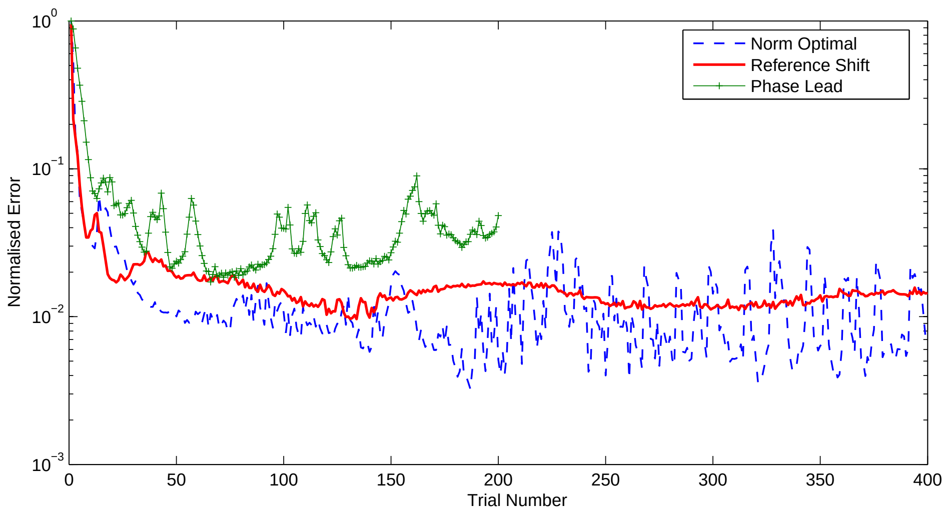


FIGURE 3.28: Performance comparison of different algorithms.

3.5 Summary

In this chapter, a non-minimum phase plant test facility has been introduced and used in ILC algorithm tests. Furthermore, a new method has been proposed to improve tracking performance which incorporates shifting the reference signal from trial to trial and the use of ILC to update the control input. It has been found that the two learning loops perform an optimal plant identification in terms of a specific weighting between learning and stability. The corresponding plant inverse is then used in the ILC law. Frequency domain filters have been used to satisfy the monotonic convergence criteria in order to prevent the system from going unstable. Experimental results have shown that the reference shift algorithm achieves faster convergence and lower final error than those algorithms previously used to control the non-minimum phase test facility. However some aspects of the reference shift algorithm require further attention. For example, the calculations necessary between each trial are complex and may limit its use when applied to rapid industrial applications. Also, as illustrated in Figure 3.19, the convergence performance is a function of the time shift which prompts the use of an adaptive filter to improve the convergence speed and maintain stability.

Chapter 4

Implementation of Stochastic ILC Algorithms

4.1 Introduction

The problem of selecting suitable learning gain matrices in ILC becomes more difficult in the presence of random disturbances such as measurement noise, re-initialisation errors, and state disturbances. A number of ILC algorithms have been developed in a stochastic setting in recent years to deal with the problems of random disturbances. The results currently available are in the form of algorithm derivation and the establishment of various fundamental systems theoretic properties Saab (2001a, 2003). In terms of eventual use in applications, the crucial next stage involves comparison of their performance. This chapter therefore involves the implementation of the stochastic learning algorithms on a gantry robot system and analysis of the experimental results obtained.

4.2 The multi-axis gantry robot

A major objective of this thesis is to experimentally verify ILC algorithms and compare the performance using two benchmarking test facilities. One is the non-minimum phase system of the last chapter. The second is a multi-axis gantry robot which will be used for the other ILC algorithm tests and comparisons contained in this thesis. This system more closely corresponds with systems found in industry, so that the experimental results from the rig will be able to provide more indicative results for practical application.

4.2.1 Mechanical Structure

The multi-axis system is a three-axis gantry robot with a conveyor (see Figure 4.1). The robot is a commercially available unit manufactured by Aerotech Inc USA and consists of three individual, linear motion axes. The gantry robot has been designed as a representation of a typical industrial tracking control problem. It is mounted above a chain conveyor system, and its task is to collect a payload from a dispenser then place the object onto the moving conveyor. Therefore the robot must accurately synchronize both speed and position with the conveyor before releasing the payload.

Many industrial processes incorporate similar actions to those featured in the multi-axis system, for example, food canning, bottle filling or automotive assembly. All of these applications require accurate tracking control for each execution over a finite duration with a minimum level of error in order to maximize production rates and minimize loss of product due to faulty manufacture.

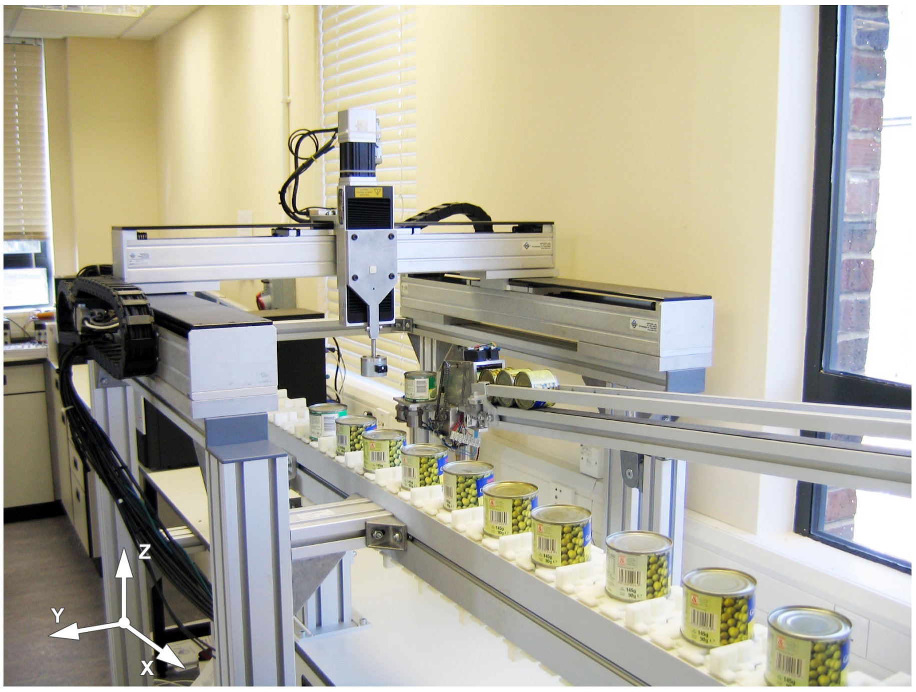


FIGURE 4.1: The multi-axes gantry robot.

As shown in Figure 4.1, the robot consists of three separate axes which are mounted perpendicular to each other. The lowest horizontal axis, X, moves parallel to the conveyor beneath. It is built from two subsystems, a brushless linear dc motor and an un-powered linear bearing slide. The second horizontal axis, Y, has one end mounted on each component of the X-axis. The Y-axis is a single brushless linear dc motor. The vertical Z-axis, consists of a linear ball-screw stage driven by a rotary brushless dc motor. All motors are powered by performance matched dc amplifiers. Position feedback is obtained by means of optical incremental encoders.

The whole gantry system is mounted on a four-leg-framework which is constructed from FlexLink components (detailed figure is shown in Figure A.1 of Appendix A. Two short legs are fixed on the work bench and the other two legs are on the ground. There are two horizontal beams connected on the longer legs in order to ensure the framework is well grounded to avoid and reduce the resonances in the high frequency tests. One is fixed under the work bench and the other is connected to the conveyor which is also fixed to the work bench.

4.2.2 Modelling of the Axes

The development of mathematical models which adequately describe the dynamic behaviour of the plant which is to be controlled is an essential part of the practical implementation component in control engineering research. Moreover, the mathematical models of the plant are extremely helpful in the designing, testing and tuning of parameters of the algorithm in a simulation environment. This will significantly reduce the risk of damaging the physical plant by applying potentially harmful input signals.

In general, given a linear plant model, the design of a controller uses either the transfer function or the state-space plant representation. However, thanks to mathematical software, the transformation between these forms (e.g. continuous-time to discrete-time, transfer function to state-space representation) is easy and fast. All three axes had been modelled by Ratcliffe (2005). However, due to the entire facility being moved and resembled, the modelling work had to be redone as the base structure and surrounding environment changed.

In order to obtain the mathematical model for each axis of the gantry robot, a series of frequency response tests have been completed and the Bode plots for all axes have been generated to be used in a model fitting procedure. Frequency domain modelling involves supplying a known input to the plant and measuring the output. A mathematical function can then relate the input and output sequences. Usually sine-waves are used as the input sequences. If a sine-wave of known frequency and amplitude is supplied to an open-loop, linear system, the measured output will also be a sine-wave, but with a different amplitude and shifted phase. Typically, the ratio of the input to output amplitude is recorded as the gain in decibels (dB) while the phase shift is recorded in degrees. If a sufficiently large range of frequencies are tested, a Bode plot representing the frequency domain response of the plant can be produced. Key features of the Bode plot can indicate plant dynamics such as poles, zeros, time delays and resonances. Using the Bode plotting rules in reverse it is possible to construct an approximate transfer function of the plant by hand.

The X-axis of the gantry robot is the most complex one of the three axes as it is a dual-side linear motor. Sinewaves of 130 different frequencies from 0.1 Hz (0.628

rad/s) to 80 Hz (502.655 rad/s) were supplied to the plant and the output data were recorded. In order to get more accurate results, two groups of 130 different frequencies were used and the testing of each frequency group was carried out three times. Each of the three response data sets were then averaged to produced the plots shown in Figure 4.2. The result proved that there was significant difference between the new model and the previously obtained model.

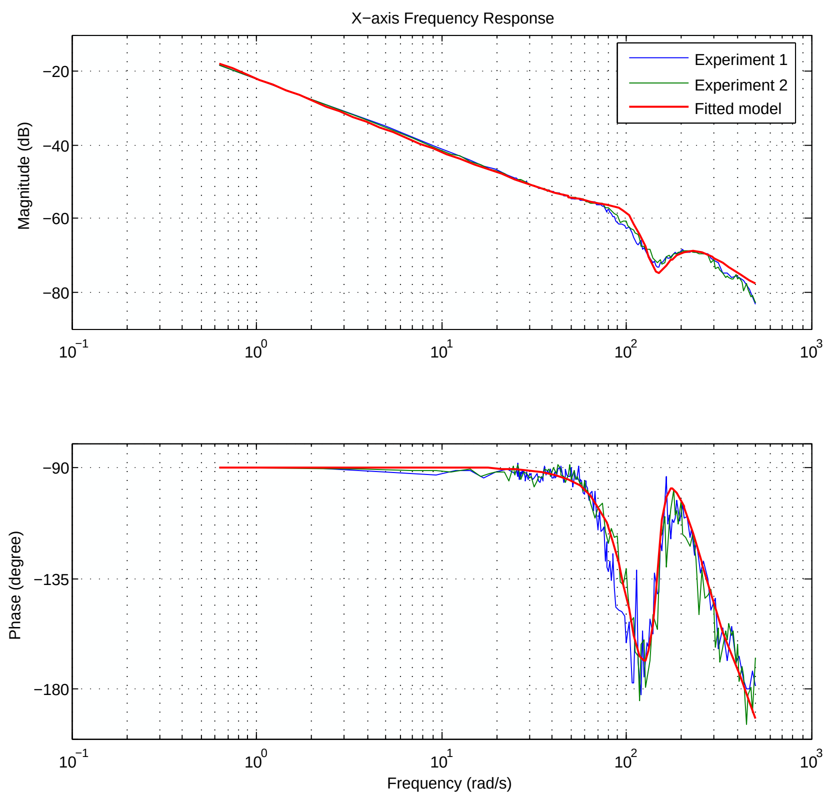


FIGURE 4.2: Frequency response testing results for the X-axis of the gantry robot.

Due to the structure and environment, the responses of the plant sometimes do not closely resemble a sine wave. Often, a drifted sine wave is instead produced. By computing the averaging value or difference, a correct frequency response value can be recovered (see Appendix A for detailed response plots). By using the Matlab control toolbox, an approximate plant model of an appropriate order (e.g. 7th order for the X-axis) was generated and then finely tuned by hand to fit the experimental data accurately. The Bode plot of the fitted model is shown in red in Figure 4.2. The continuous-time transfer function of the fitted X-axis model is

$$\begin{aligned} G_X(s) &= \frac{13077183.4436(s+113.4)}{s(s^2+61.57s+1.125\times10^4)} \times \cdots \\ & \quad \frac{(s^2+30.28s+2.13\times10^4)}{(s^2+227.9s+5.647\times10^4)(s^2+466.1s+6.142\times10^5)} \end{aligned} \tag{4.1}$$

The mechanical structure of the Y-axis and the Z-axis are simpler than that of the X-

axis. Sinewaves of 154 different frequencies from 0.1 Hz (0.628 rad/s) to 130 Hz (816.814 rad/s) were used in the Y-axis frequency response tests and 120 different frequencies from 0.1 Hz (0.628 rad/s) to 125 Hz (785.398 rad/s) were used in the Z-axis tests. The testing of these two axes was repeated three times and the resulting data were averaged to produce the following Bode plots (Figure 4.3 and Figure 4.4).

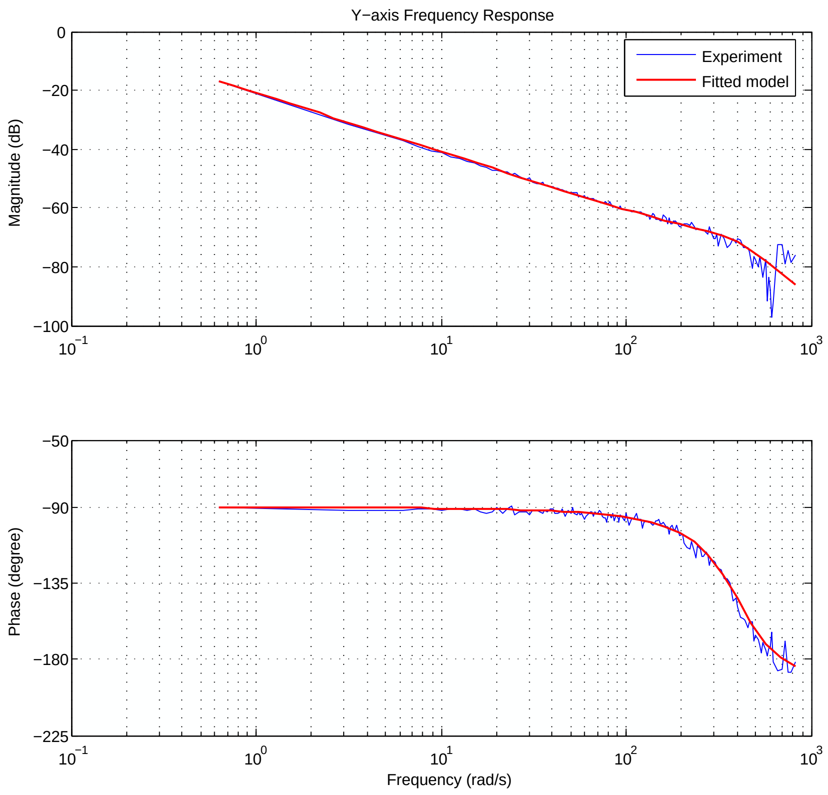


FIGURE 4.3: Frequency response testing results for the Y-axis of the gantry robot.

The Bode plots of the corresponding fitted models are shown in red. The fitted models are of 3rd order and the continuous-time transfer functions are given by (4.2) and (4.3).

$$G_Y(s) = \frac{23.7356(s + 661.2)}{s(s^2 + 426.7s + 1.744 \times 10^5)} \tag{4.2}$$

$$G_Z(s) = \frac{15.8869(s + 850.3)}{s(s^2 + 707.6s + 3.377 \times 10^5)} \tag{4.3}$$

By using Matlab commands, the corresponding discrete-time state space matrices can be generated which will be used in most of the practical implementation that follows in the case where a model is needed. The state space models for all axes were generated, and are given below. The sampling time can be varied depending on the requirements of the algorithms applied. However, in order to make sure all results are comparable, $T_s = 0.01$s is used in all experimental tests.

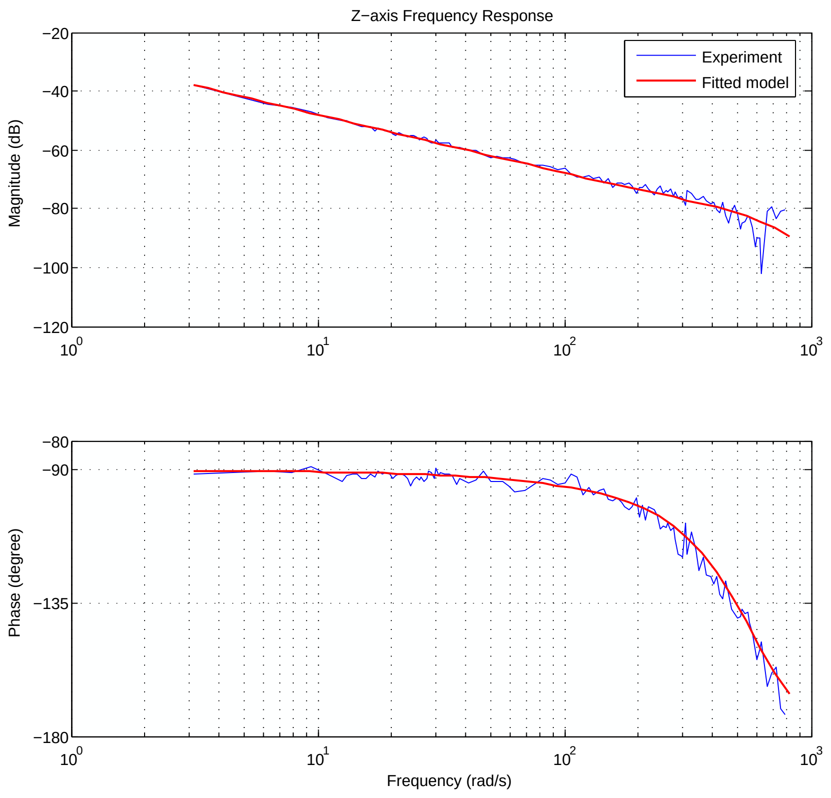


FIGURE 4.4: Frequency response testing results for the Z-axis of the gantry robot.

- X-axis 7th order model

$$A_X = \begin{bmatrix} 0.3879 & 1.0000 & 0.2138 & 0 & 0.1041 & 0 & 0.0832 \\ -0.3898 & 0.3879 & 0.1744 & 0 & 0.0849 & 0 & 0.0678 \\ 0 & 0 & -0.1575 & 0.2500 & -0.2006 & 0 & -0.1603 \\ 0 & 0 & -0.3103 & -0.1575 & -0.0555 & 0 & -0.0444 \\ 0 & 0 & 0 & 0 & 0.0353 & 0.5000 & 0.2809 \\ 0 & 0 & 0 & 0 & -0.0164 & 0.0353 & -0.2757 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

$$B_X = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.0910 \end{bmatrix}$$

$$C_X = \begin{bmatrix} 0.0391 & 0 & 0.0146 & 0 & 0.0071 & 0 & 0.0057 \end{bmatrix}$$

- Y-axis 3rd order model

$$
\begin{aligned}
A_Y &= \begin{bmatrix} -0.1067 & 0.1250 & 0.0777 \\ -0.0211 & -0.1067 & 0.1016 \\ 0 & 0 & 1.0000 \end{bmatrix} \\
B_Y &= \begin{bmatrix} 0 \\ 0 \\ 0.0286 \end{bmatrix} \\
C_Y &= \begin{bmatrix} 0.0360 & 0 & 0.0286 \end{bmatrix}
\end{aligned}
$$

- Z-axis 3rd order model

$$
\begin{aligned}
A_Z &= \begin{bmatrix} -0.0030 & 0.0625 & 0.0758 \\ -0.0134 & -0.0030 & 0.0637 \\ 0 & 0 & 1.0000 \end{bmatrix} \\
B_Z &= \begin{bmatrix} 0 \\ 0 \\ 0.0191 \end{bmatrix} \\
C_Z &= \begin{bmatrix} 0.0232 & 0 & 0.0191 \end{bmatrix}
\end{aligned}
$$

The order of the state-space models corresponds with that of the transfer function models. All the values in the matrices are truncated after 4 digits. A full precision representation of the state space matrices for the X, Y and Z axes are given in Appendix A.

4.2.3 Hardware and Software Setup

4.2.3.1 Hardware Settings

The gantry robot comprises three different motors (either linear motors or rotary motors) each incorporating a shaft encoder. In addition to the robot itself, there are several additional components connected to the gantry robot which are necessary in order to complete the whole control system and ensure safe operation. Besides the control computer, the test facility also includes two large drive cabinets, five smaller breakout boxes and steel conduit for signal cables. A general layout of the existing infrastructure is presented in Figure 4.5

The drive cabinets house the high power amplifiers and drives, as well as a selection of individual power supply units in a cooled environment, which is screened from Radio Frequency Interference (RFI). Different components of the test facility have varied power supply requirements, ranging from 240V alternating current (ac) for cooling fans and

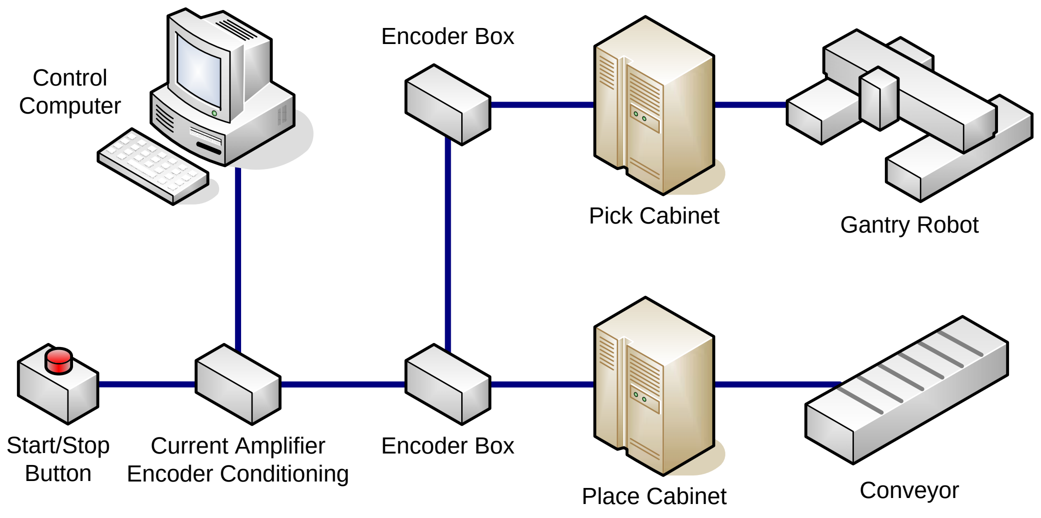


FIGURE 4.5: The layout of hardware components.

low voltage power supplies, 110V ac for the motor amplifiers, 12V direct current (dc) for the control signal amplifier circuits, +12V dc for the main ignition and emergency stop circuit and +5V for the encoder circuits. The drive cabinets have been named 'Pick' and 'Place' to reflect the operation which a robot would perform at that location along the conveyor; picking payloads off, or placing them on. The breakout boxes serve as connection points to which external hardware such as robots and conveyors can be connected. They also act as signal routing points, where signals and power cables from different components of the test facility can be connected to other components.

4.2.3.2 Software Developments

The control computer is a Pentium 4, Desktop PC, running the Linux operating system and using the 'C' programming language for software development. Linux is reported to be well suited to real-time control applications because interrupt handling routines generate significantly shorter delay latencies than other operating systems. However, a control system running under the Microsoft Windows operating system has been developed as well in order to make data storage and operation more straightforward as there are many data processing or technique charting software packages in the Windows platform. In terms of control signal processing, the 10 Volt control set-points are generated by a 16 channel, 12-bit Digital to Analogue (D/A) expansion card, while the encoder generated signals are read by a separate 4 channel, quadrature decoder expansion card. A third, binary input expansion card with an on-board timer is employed to achieve strict sample interval timing.

The ILC control algorithms have been implemented by using 'C' programming language with the following structure (Figure 4.6):

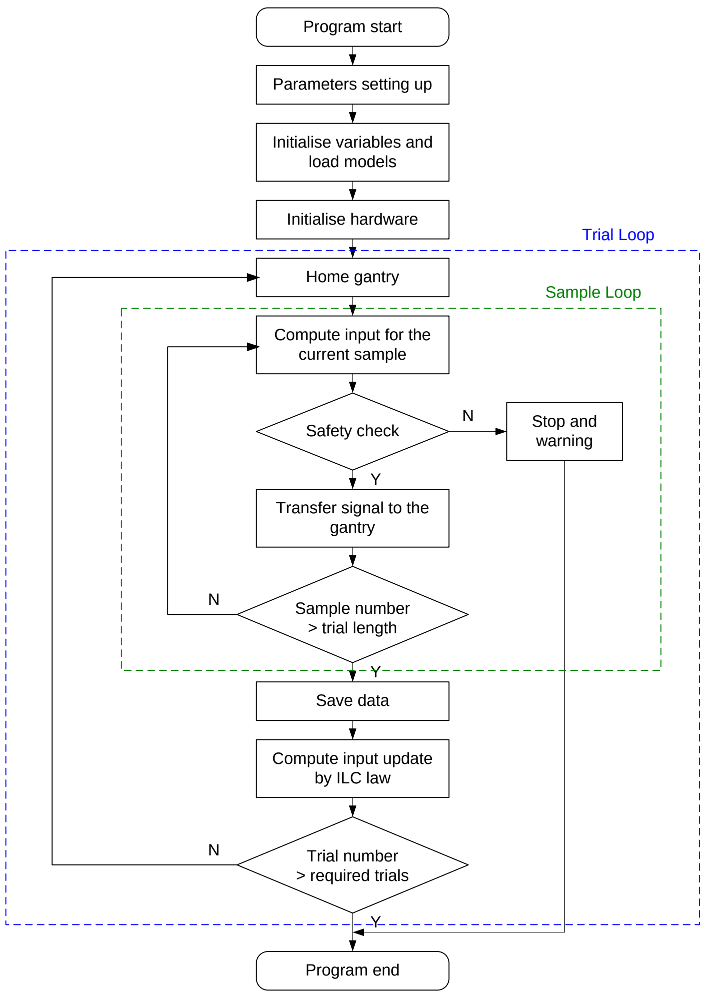

Program start
Parameters setting up
Initialise variables and load models
Initialise hardware
Trial Loop
Home gantry
Sample Loop
Compute input for the current sample
Safety check
N
Stop and warning
Y
Transfer signal to the gantry
N
Sample number > trial length
Y
Save data
Compute input update by ILC law
N
Trial number > required trials
Y
Program end

FIGURE 4.6: Block diagram of the ILC control program structure

4.2.4 Reference Trajectories

The gantry robot is designed to complete a pick-and-place motion on the conveyor over and over again. A reference trajectory for the gantry movement was pre-defined with the purpose of synchronising its the motion with that of the conveyor, which is running at a constant speed. Each axis is controlled individually and has its own reference trajectory (see Figure 4.7).

The 3-dimensional combined reference trajectory given in Figure 4.8 clearly shows the "pick-and-place" process. The signal length is 2 seconds in the default setting. A sample frequency of 1kHz is used for testing the basic algorithms, resulting in 2000 samples per trial. A sample frequency of 100Hz is used for testing the model-based algorithms. This frequency was deemed adequate, giving 200 sample instants per trial. Normally before commencing every recorded experiment, several hundred trials will be performed in order to make sure every mechanical and electronic component is in good condition.

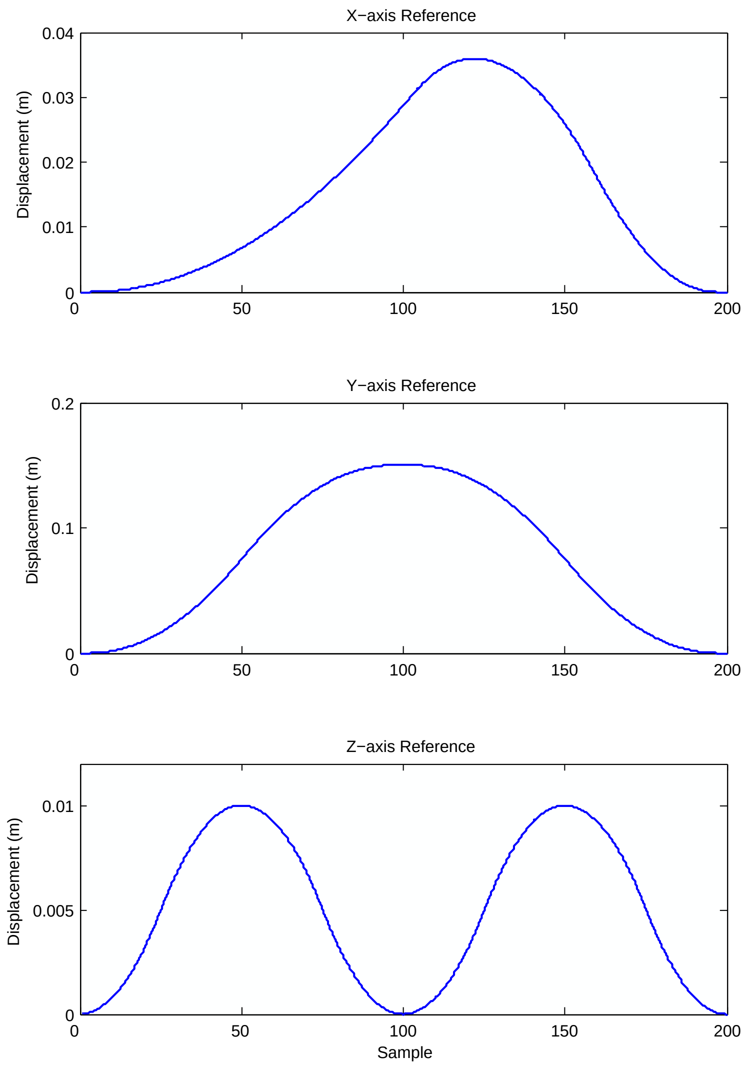


FIGURE 4.7: Individual reference trajectory for all axes.

4.3 Algorithm Description

4.3.1 Discrete Stochastic ILC Algorithm

The theory of the stochastic ILC algorithms which forms the basis of the experimental results in this thesis and its development were presented in Saab (2001a, 2003). Until present, no one has attempted to test these algorithms on a physical plant which is a crucial stage before the possibility of practical industrial application. A summary of this work is given here.

The proposed stochastic learning algorithm is derived for a discrete-time-varying linear system in a stochastic setting described by

$$\begin{aligned} x_k(t+1) &= A(t)x_k(t) + B(t)u_k(t) + \omega_k(t) \\ y_k(t) &= C(t)x_k(t) + \nu_k(t) \end{aligned} \tag{4.4}$$

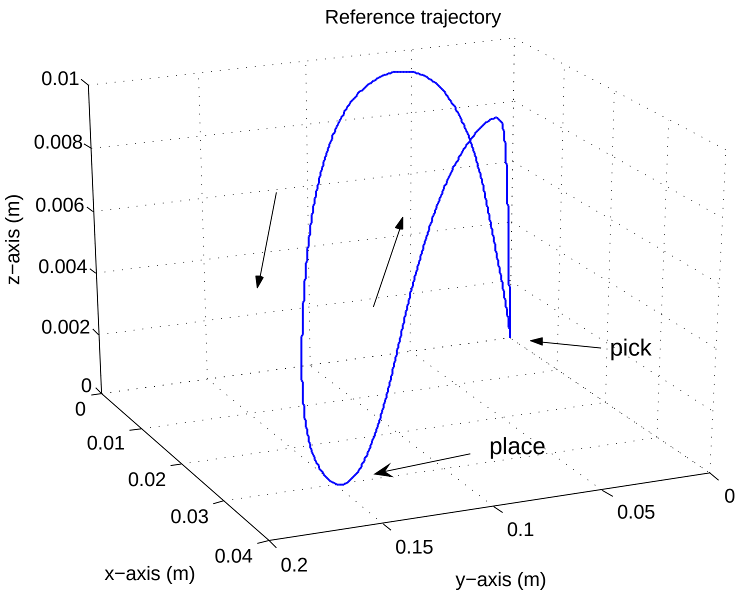


FIGURE 4.8: 3-D combined reference trajectory.

where at trial k, $t \in [0, T]$ and $T < \infty$ denotes the trial length. the system state $x_k(t) \in \Re^n$, the system input $u_k(t) \in \Re^p$, the state disturbance $\omega_k(t) \in \Re^n$, the system measured output $y_k(t) \in \Re^q$ and the system measurement error $\nu_k(t) \in \Re^q$. However, the gantry robot used for the practical implementation can be seen as a multi-axis, single-input-single-output (SISO), linear time invariant system. Therefore, (4.4) can be simplified to give a discrete linear time-invariant (LTI) system.

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t) + \omega_k(t) \\ y_k(t) &= Cx_k(t) + \nu_k(t) \end{aligned} \tag{4.5}$$

where $t \in [0, T]$, the system state $x_k(t) \in \Re^n$, the system input $u_k(t) \in \Re^p$, the state disturbance $\omega_k(t) \in \Re^n$, the system measured output $y_k(t) \in \Re^q$ and the system measurement error $\nu_k(t) \in \Re^q$. Attention will focus on the following algorithms from Saab (1995, 2001a) which are developed based on both D-type and P-type ILC. A summarised derivation of this approach is given below.

D-type ILC — this uses the approximate error differentiation instead of the error derivative considered in Arimoto et al. (1984a) for continuous-time systems

$$u_{k+1}(t) = u_k(t) + K_k[e_k(t+1) - e_k(t)] \tag{4.6}$$

The relevant convergence conditions for this discrete-time D-type ILC algorithm was analysed and given in Saab (1994) and Saab (1995).

P-type ILC — here

$$u_{k+1}(t) = u_k(t) + K_k e_k(t+1) \tag{4.7}$$

where K_k is the $(p \times q)$ learning control gain matrix, and $e_k(t)$ is the output error, i.e. $e_k(t) = y_d(t) - y_k(t)$ where $y_d(t)$ is the desired output trajectory. It is assumed that for any realisable output trajectory and an appropriate initial condition $x_d(0)$, there exists a unique control input $u_d(t)$ which satisfied:

$$\begin{aligned} x_d(t+1) &= Ax_d(t) + Bu_d(t) \\ y_d(t) &= Cx_d(t) \end{aligned} \tag{4.8}$$

If $C(t+1)B(t)$ is full-column rank, in this case (i.e. a SISO LTI system), the basic assumption is $CB \neq 0$. Then the unique control input can be given by

$$u_d(t) = [(CB)^T CB]^{-1}(CB)^T \cdot [y_d(t+1) - CAx_d(t)]$$

Define the state and input error vectors as

$$\delta x(t,k) = x_d(t) - x(t,k)$$

$$\delta u(t,k) = u_d(t) - u(t,k)$$

There are some assumptions that made before the development of the stochastic ILC algorithm

- $\{\omega(t,k)\}$ and $\{\nu(t,k)\}$ are sequences of zeros-mean white Gaussian noise.
- $\mathbf{Q}_t = E(\omega(t,k)\omega(t,k)^T)$ and $\mathbf{P}_{x,0} = E_k(\delta x(0,k)\delta x(0,k)^T)$ are positive semi-definite matrices.
- $\mathbf{R}_t = E_k(v(0,k)v(0,k)^T)$ is a positive definite matrix and $\mathbf{P}_{u,0} = E(\delta u(t,0)\delta u(t,0)^T)$ is a symmetric positive definite matrix.
- E and E_k are the expectation operators with respect to the time domain and the trial domain respectively.

The main purpose is to find a proper learning gain matrix K_k such that the input error variance is minimised. According to the definition and the D-type ILC control law, the input error for the $k+1^{th}$ trial can be given as

$$\begin{aligned} \delta u(t,k+1) &= \delta u(t,k) - K_k\{C[x_d(t+1) - x(t+1,k)] \\ &\quad -\nu(t+1,k)\} + K_k\{C[x_d(t) - x(t,k)] - \nu(t,k)\} \\ &= [I - K_kCB]\delta u(t,k) + K_k[C - CA]\delta x(t,k) \\ &\quad +K_k[\nu(t+1,k) - \nu(t,k) + C\omega(t,k)] \end{aligned}$$

Taking the assumptions into account, at the $k+1^{th}$ trial

$$\begin{aligned}
\mathbf{P}_{u,k+1} &\triangleq E[\delta u(t,k+1)\delta u(t,k+1)^T] \\
&= (I-K_kCB)\mathbf{P}_{u,k}(I-K_kCB)^T + K_kCA\mathbf{P}_{x,k}(CA)^TK_k^T \\
&\quad -(I-K_kCB)\mathbf{P}_{ux,k}(CA)^TK_k^T - K_kCA\mathbf{P}_{ux,k}^T(I-K_kCB)^T \\
&\quad +K_kC\mathbf{Q}_tC^TK_k^T + K_k\mathbf{R}_{t+1}K_k^T \qquad (4.9)
\end{aligned}$$

Since the initial state error $\delta x(0,k)$ and the initial input error $\delta u(t,0)$ are assumed to be zero-mean white noise, it can be shown that $\mathbf{P}_{ux,k} = E(\delta x(t,k)\delta x(t,k)^T) = 0$ (Saab, 2001a). Then (4.9) can be deduced to

$$\begin{aligned}
\mathbf{P}_{u,k+1} &= \mathbf{P}_{u,k} - \mathbf{P}_{u,k}(CB)^TK_k^T - K_kCB\mathbf{P}_{u,k} \\
&\quad +K_k[CB\mathbf{P}_{u,k}(CB)^T + CA\mathbf{P}_{x,k}(CA)^T \\
&\quad +C\mathbf{Q}_tC^T + \mathbf{R}_{t+1}]K_k^T \qquad (4.10)
\end{aligned}$$

Next the learning gain matrix is chosen such that the trace of $\mathbf{P}_{u,k+1}$ is minimised. Then set

$$\begin{aligned}
\frac{\mathrm{d}(\mathrm{trace}(\mathbf{P}_{u,k+1}))}{\mathrm{d}K_k} &\equiv 0 \\
\frac{\mathrm{d}(\mathrm{trace}(\mathbf{P}_{u,k+1}))}{\mathrm{d}K_k} &= -2\mathbf{P}_{u,k}(CB)^T + 2K_k[(CB)\mathbf{P}_{u,k}(CB)^T \\
&\quad +(CA)\mathbf{P}_{x,k}(CA)^T + C\mathbf{Q}_tC^T + \mathbf{R}_{t+1}] \equiv 0
\end{aligned}$$

Finally, the learning control gain matrix K_k is given together with the update law for $P_{u,k}$ and $P_{x,k}$

$$\begin{aligned}
K_k &= \mathbf{P}_{u,k}(CB)^T\cdot[(CB)\mathbf{P}_{u,k}(CB)^T + (C-CA) \\
&\quad \mathbf{P}_{x,k}(C-CA)^T + C\mathbf{Q}_tC^T + \mathbf{R}_t]^{-1} \qquad (4.11) \\
\mathbf{P}_{u,k+1} &= (I-K_kCB)\mathbf{P}_{u,k} \qquad (4.12) \\
\mathbf{P}_{x,k+1} &= A\mathbf{P}_{x,k}A^T + B\mathbf{P}_{u,k}B^T + \mathbf{Q}_t \qquad (4.13)
\end{aligned}$$

Note that the P-type learning gain matrix has the same form as the D-type learning gain matrix (Saab, 2003).

4.3.2 Discrete Sub-optimal Stochastic Learning Algorithm

The stochastic ILC algorithm requires the state matrix A to update the learning gain matrix. A sub-optimal algorithm was proposed following the stochastic learning algorithm in order to simplify the computation of the learning gain matrix by removing the

A matrix and the term $\mathbf{P}_{x,k}$ and hence the learning control gain matrix K_k is set to

$$\begin{aligned} K_k &= \mathbf{P}_{u,k}(CB)^T \cdot [(CB)\mathbf{P}_{u,k}(CB)^T + C\mathbf{Q}_t C^T + R_t]^{-1} \quad (4.14) \\ \mathbf{P}_{u,k+1} &= (I - K_k CB)\mathbf{P}_{u,k} \quad (4.15) \end{aligned}$$

where $\mathbf{P}_{u,0}$ and $\mathbf{Q}_t$ are the matrices described previously.

4.4 Controller Arrangement

4.4.1 Pure ILC Approaches

Firstly, the pure ILC design is considered. With only the feedforward signal from the ILC controller, the control inputs are updated by the previous error. Figure 4.9 shows the block diagram for this arrangement. The control input can be modified by either D-type algorithm or P-type algorithm.

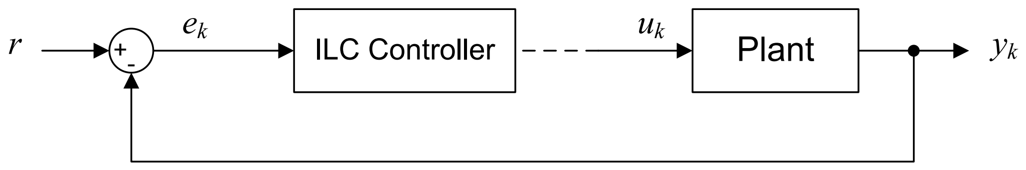


FIGURE 4.9: Block diagram of the controller structure and arrangement.

4.4.2 ILC with Feedback Controller

Figure 4.10 shows the structure of the controller arrangement which is used in the hybrid approaches. It is a parallel arrangement of a PID feedback controller and an ILC controller which is more capable of compensating for sudden changes in plant dynamics (Ratcliffe et al., 2005a). Also, it enables better tracking performance for the initial trials.

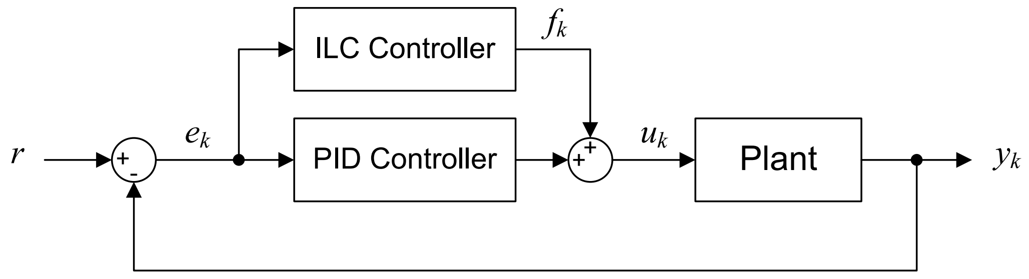


FIGURE 4.10: Block diagram of the controller structure and arrangement.

When the PID controller is turned on, a feedback/feedforward scheme is implemented.

Let f_k be the feedforward signal from the ILC controller, for the D-type algorithm (4.6) this is given by

$$f_{k+1}(t) = f_k(t) + K_k[e_k(t+1) - e_k(t)] \tag{4.16}$$

and for the P-type algorithm, by

$$f_{k+1}(t) = f_k(t) + K_k e_k(t+1) \tag{4.17}$$

Hence for the D-type algorithm

$$u_k(t) = f_k(t) + [K_p e_k(t) + K_i \int_0^t e_k(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e_k(t)}{\mathrm{d}t}] \tag{4.18}$$

Defining the operator $\mathrm{PID}[e_k(t)] \triangleq K_p e_k(t) + K_i \int e_k(t)\mathrm{d}t + K_d \frac{\mathrm{d}e_k(t)}{\mathrm{d}t}$ allows this to be written as

$$\begin{aligned} u_k(t) &= f_k(t) + \mathrm{PID}[e_k(t)] \\ u_{k+1}(t) &= f_{k+1}(t) + \mathrm{PID}[e_{k+1}(t)] \\ &= u_k(t) + \mathrm{PID}[e_{k+1}(t) - e_k(t)] + K_k[e_k(t+1) - e_k(t)] \end{aligned}$$

and similarly for the P-type algorithm

$$u_{k+1}(t) = u_k(t) + \mathrm{PID}[e_{k+1}(t) - e_k(t)] + K_k e_k(t+1) \tag{4.19}$$

With a PID feedback controller, the input update is consequently the sum of the ILC update plus the response of the PID controller to the difference between the current and previous trial outputs. Note that, after sampling with period T_s, (4.18) becomes

$$\begin{aligned} u_k(t) &= f_k(t) + [K_p e_k(t) + K_i \sum_{\tau=1}^{t} \frac{(e_k(\tau) + e_k(\tau-1))T_s}{2} + K_d \frac{e_k(t) - e_k(t-1)}{T_s}] \\ & t \in [0, T] \end{aligned}$$

4.4.3 Filter Design

Simulations and initial experimental tests lead to some results in which tracking error started to diverge after a number of trials. It has been reported (see for example Longman (2000)) that in many ILC algorithms high frequency noise will build up as the number of trials increase and tracking of the reference signal then begins to diverge. This is illustrated in Figure 4.11, where the upper graph shows the tracking error over all trials. Only 10 trials were completed and the experiment was then terminated due to the presence of instability over a narrow frequency band. The gantry actually started to vibrate which is harmful for the mechanical components. A frequency analysis was applied to the error signal (see Figure 4.11 lower graph), and a frequency of approximately 11-12Hz (around 70–75 rad/s) can be seen to build up. Looking back at the X-axis Bode plot (Figure 4.2) a frequency of around 70–75 rad/s corresponds exactly with the first resonant peak of the open loop system. More importantly, this is where the resonance causes the phase plot to drop rapidly towards $-180°$. From this result, it is logical to conclude that the plant vibrations are caused by the systems own resonance.

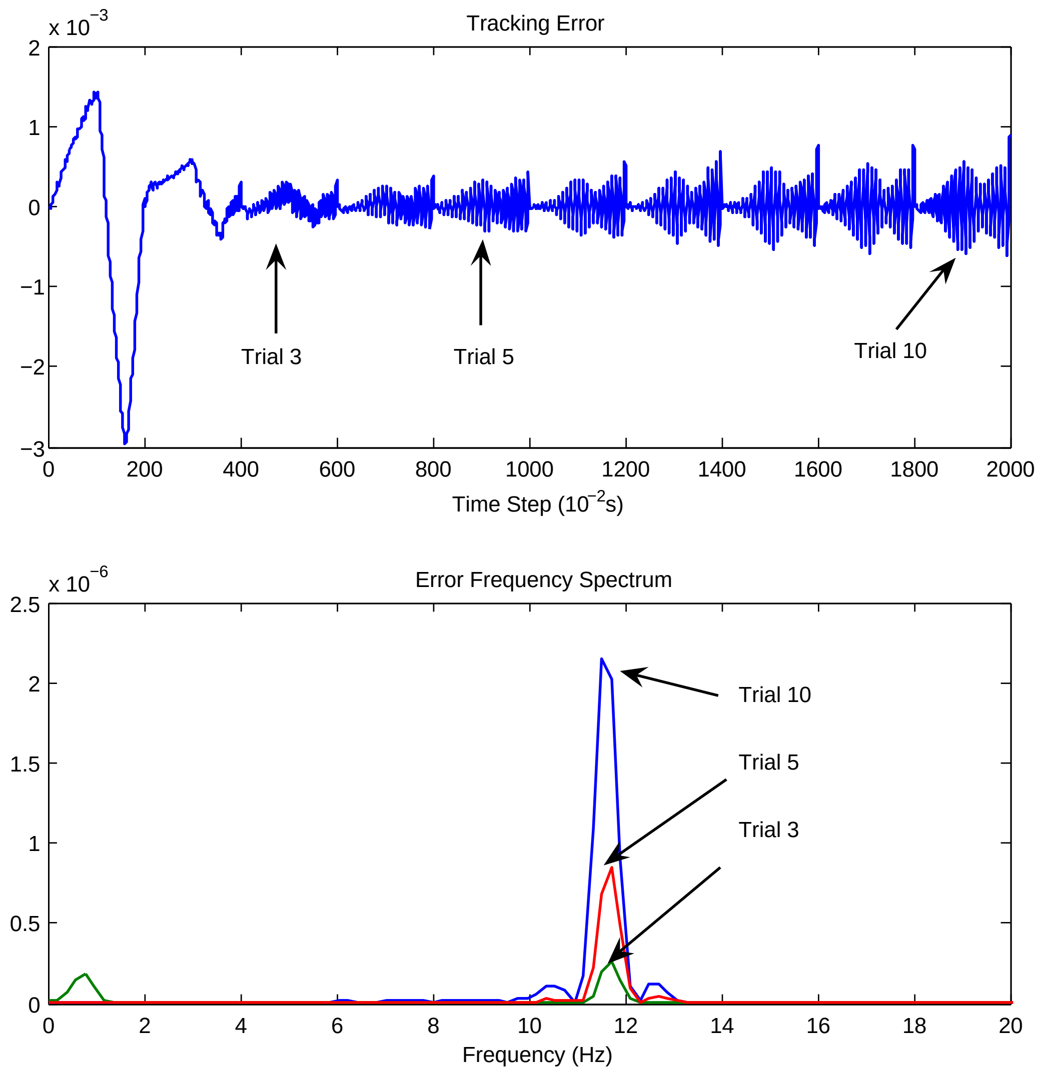
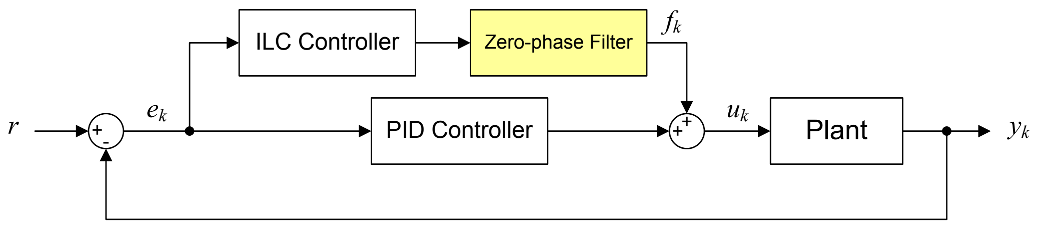


FIGURE 4.11: The frequency spectrum of tracking error for a range of trials.

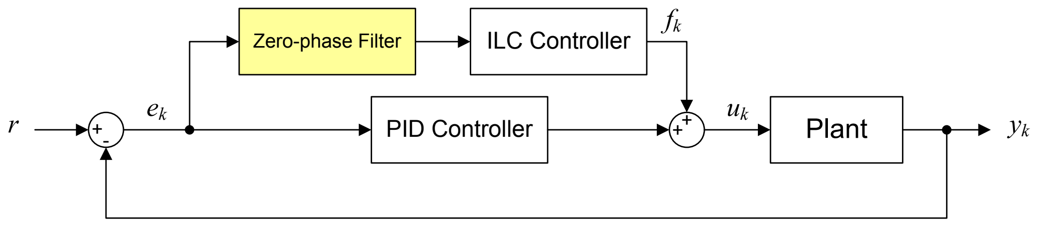
Since ILC design is anti-causal along-the-pass, zero-phase filtering (see, for example, Plotnik and Longman (1999)) is feasible and here a low-pass filter of this form has been used to mitigate the effects of high frequency noise (note that it has been reported Chen and Longman (2002) that excessive phase shift can cause the ILC controller to incorrectly compensate for the error and lead to an unstable system). A 4^{th} order Chebychev filter (see (4.20)) which has a cut off frequency of approximately 5Hz has been introduced and is given by

$$H(z) = \frac{0.0002 + 0.0007z^{-1} + 0.0011z^{-2} + 0.0007z^{-3} + 0.0002z^{-4}}{1 - 3.5328z^{-1} + 4.7819z^{-2} - 2.9328z^{-3} + 0.6868z^{-4}} \tag{4.20}$$

Moreover, since the filter placement in the overall scheme is also critical for convergence and tracking performance, two arrangements have been considered: In Figure 4.12(a) the filter is applied to the feedforward signal and in Figure 4.12(b) it is applied to the error signal prior to computation of the control signal for the next trial. Experiments have shown that the latter arrangement gives the superior performance, and so it will be adopted in the tests which follow.



(a) Filtering the output of the ILC controller.



(b) Filtering error vector before the ILC update

FIGURE 4.12: Filter arrangements.

Figure 4.13 shows the results for different filter arrangements. It is clear that without a filter, the experiment had to be stopped after only 15 trials because the high frequency noise led to big vibrations which might damage the robot. The experimental results show that using filters applied to the ILC update produces far superior results. Therefore this filter arrangement has been used in all experiment that follows.

4.5 Simulations

4.5.1 Stochastic Learning Algorithm

Prior to experimental implementation, a series of designs have been completed and their performance evaluated in simulation using the Matlab/Simulink control toolbox. The system model here is a SISO LTI model. Therefore the parameter $\mathbf{P}_{u,k}$ is then a scalar. $\mathbf{P}_{x,k}$ and $\mathbf{Q}_t$ are square matrices and have the same dimensions as the system order.

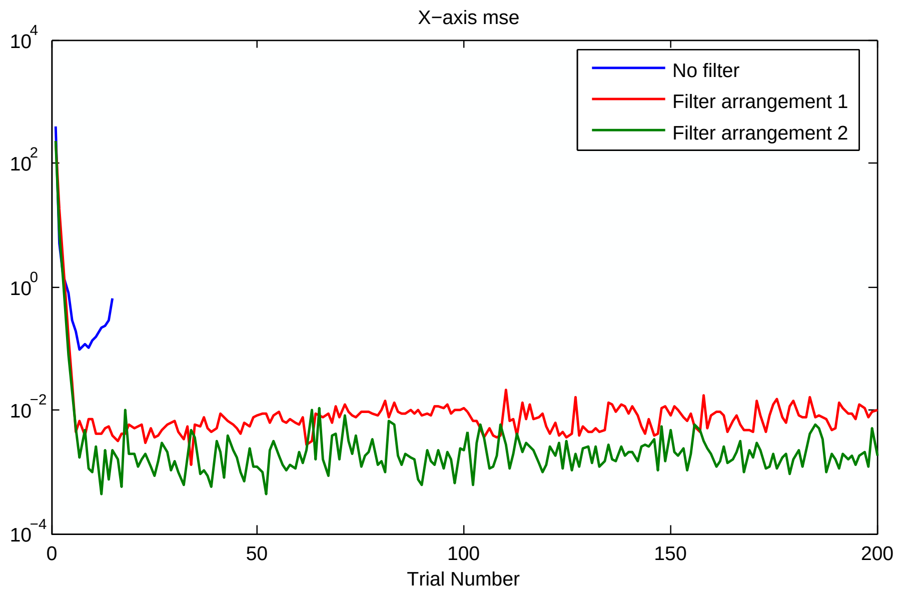


FIGURE 4.13: Experimental results for different filter designs ($P_{u,0} = 100, Q_t = 0.01$).

Figures 4.14 and 4.15 show the X-axis mse results obtained by varying $P_{u,0}$ (the initial value of $\mathbf{P}_{u,k}$) and $\mathbf{Q}_t$ where

$$\mathbf{Q}_t = Q_t I \tag{4.21}$$

$$\mathbf{P}_{x,k} = P_{x,k} I \tag{4.22}$$

Here Q_t and $P_{x,k}$ are scalars and I is the identity matrix of appropriate dimension. $\mathbf{R}_t$ is set to the mean value of the white noise, this being extracted from the error vector after each completed trial. These results indicate that selecting a larger $P_{u,0}$ and a smaller Q_t gives better tracking performance both in terms of convergence speed and final error. Although increasing the value of $P_{u,0}$ and decreasing the value of Q_t give improved performance, the simulations only use $P_{u,0} = 1000$ as the biggest and $Q_t = 10^{-4}$ as the smallest value because in the simulation there exists a quantiser block which uses the smallest step of the distance which the encoder is able to measure in practice. This prevents the tracking error further reducing. Also small amounts of random disturbance are introduced to simulate a more practical environment.

The simulation results corresponding to varying the parameter value of $P_{u,0}$ and Q_t lead to quite obvious conclusions. However, use of different values of $P_{x,0}$ have been tested in both simulation and experiment. The results appear to show hardly any difference. Therefore the subsequent series of tests did not take $P_{x,0}$ in consideration.

Figure 4.16 shows the mean squared error plot of the X-axis with different PID feedback parameters. PID= $\{0, 0, 0\}$ means the feedback controller is not used. From the simulation results, it can be seen that higher PID gains are able to give better performance only in the first trial, but do not assist the ILC controller. The overall performance is

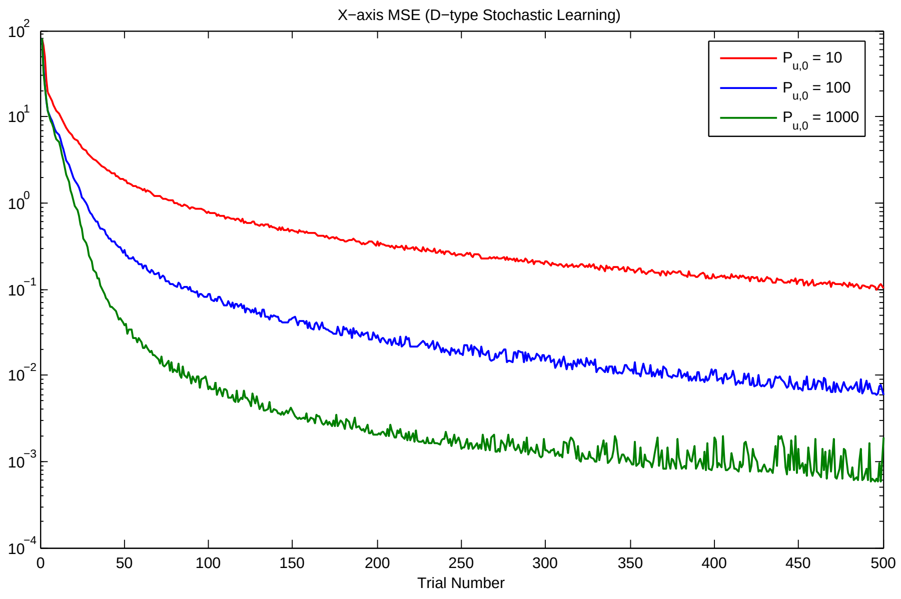


FIGURE 4.14: Simulations of varying $P_{u,0}$. (PID $= \{60, 30, 0.2\}, Q_t = 0.001, P_{x,0} = 0.1$)

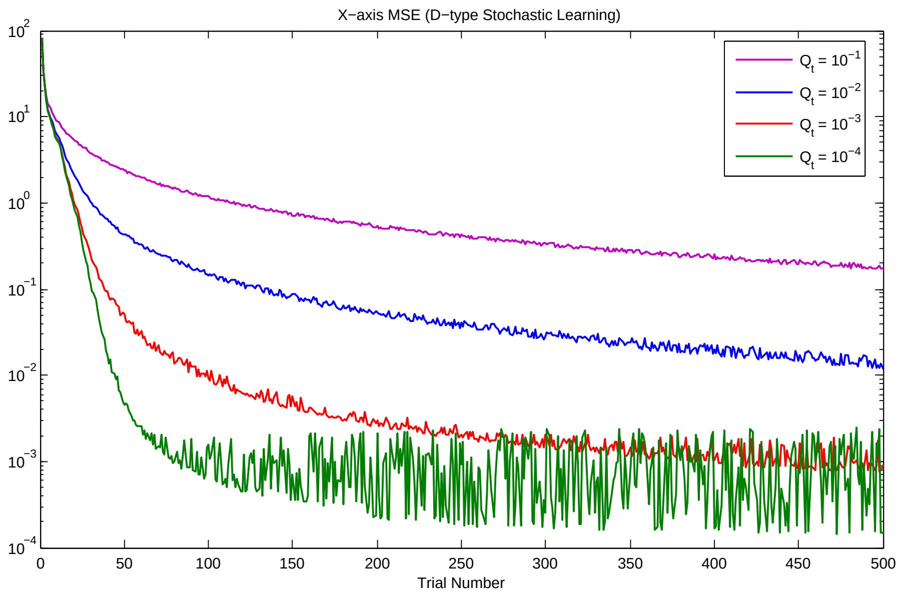


FIGURE 4.15: Simulations of varying Q_t. (PID $= \{60, 30, 0.2\}, P_{u,0} = 800, P_{x,0} = 0.1$)

degraded.

4.5.2 Sub-optimal Stochastic Learning Algorithm

The sub-optimal algorithm has also been simulated using various parameters. A batch of tests has been performed using different combinations of these parameters (see Table 4.1). Each combination was implemented for 100 trials, the PI_{100} (see Equation 4.23) performance has been used in the evaluation. The same evaluation method was used in Ratcliffe et al. (2005a) to obtain the optimal weighting matrices of the norm-optimal

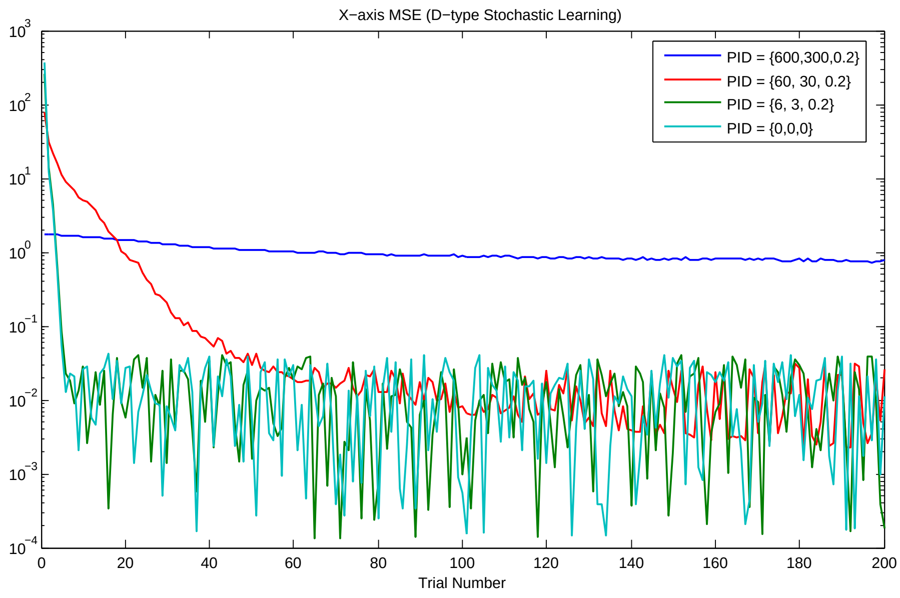


FIGURE 4.16: Simulations with varying PID gains. ($P_{u,0} = 1000, Q_t = 0.001, P_{x,0} = 0.1$)

ILC algorithm. Because the algorithm has two parameters, it is particularly suitable to plot the algorithm performance on a 3-D surface chart.

$$PI_{100} = \sum_{k=1}^{100} \bar{e}_k^2 \tag{4.23}$$

Where $\bar{\cdot}$ is the operator of computing mean value.

$P_{u,0}$	1	5	10	50	100	500	1000		
Q_t	10	5	1	0.5	0.1	0.05	0.01	0.005	0.001

TABLE 4.1: Testing parameters value

The results of PI_{100} can be found in Table 4.2 and a 3-D mesh plot has been generated using the data (see Figure 4.17) which displays the performance for the X-axis.

$P_{u,0} =$	1	5	10	50	100	500	1000
Q_t=10	1.7682	1.7606	1.7511	1.6766	1.5917	1.1183	0.8028
Q_t=5	1.7661	1.7509	1.7315	1.5919	1.4435	0.8031	0.5047
Q_t=1	1.7509	1.6776	1.5934	1.1223	0.8064	0.2375	0.1293
Q_t=0.5	1.7330	1.5952	1.4488	0.8107	0.5101	0.1298	0.0736
Q_t=0.1	1.6085	1.1591	0.8428	0.2490	0.1344	0.0425	0.0357
Q_t=0.05	1.4936	0.8805	0.5619	0.1400	0.0773	0.0358	0.0364
Q_t=0.01	1.1070	0.3574	0.1811	0.0448	0.0359	0.0414	0.0446
Q_t=0.005	0.9385	0.2285	0.1070	0.0358	0.0367	2.7598	0.7943
Q_t=0.001	0.5413	0.0549	0.0359	3.9844	3.3938	3.0401	2.9749

TABLE 4.2: PI_{100} Results for the X-axis ($\times 10^4$)

To the left of the chart is a region of poor tracking performance, where the PI_{100} value

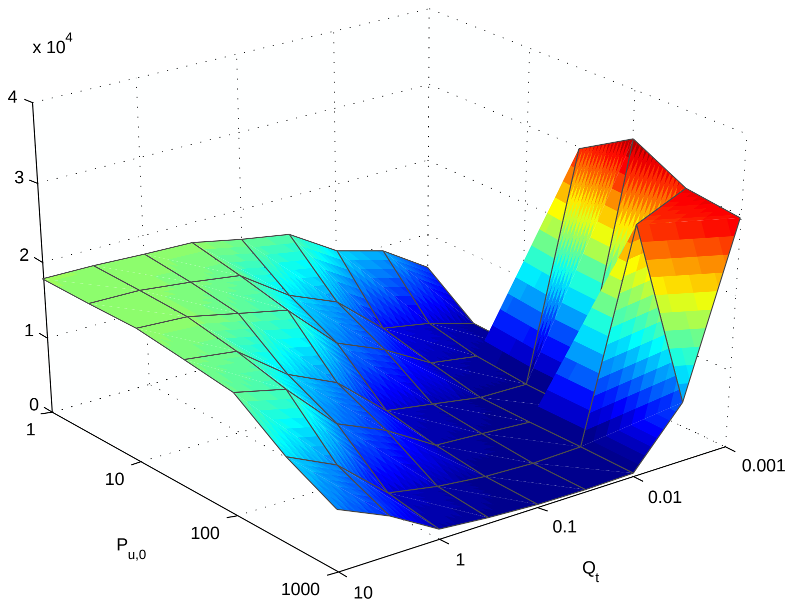


FIGURE 4.17: X-axis PI_{100} for various $P_{u,0}$ and Q_t

is very large, indicating that virtually nothing is learnt during the 100 trials of the test. As the ratio of $P_{u,0}$ and Q_t increases, gradually PI_{100} reduces, indicating that the performance is improving. This is represented by the slope to the left side of the chart. As the $P_{u,0}$ and Q_t ratio continues to increase, PI_{100} is reduced to very small values, indicating that the perfect trajectory is learned in very few trials. This implies that the balance of $P_{u,0}$ and Q_t is now approaching optimality. Temporarily increasing the ratio of $P_{u,0}$ and Q_t has little effect on the performance, until the system becomes unstable and PI_{100} jumps back to a very large value. This is represented by the channel, and then the steep slope to the right of the chart.

4.6 Experimental Results

4.6.1 Test Parameters

The ILC controller has been applied to all three axes of the gantry robot, and a zero-phase filter has been used in all experiments. The reference trajectories (Figure 4.8) are the same as those used in all previously reported results in which ILC algorithms have been implemented on the gantry robot (to enable the broadest possible comparison to be made). A defining feature of ILC is that there is an undefined stoppage time between trials, in which the control computation can be taken. The gantry axes are homed to a predefined point, before each trial begins. Axis homing is performed to within ± 30 microns to minimise the effects of initial state error. The sampling time $T_s = 0.01$s (sampling frequency: 100Hz) has been used in all tests.

4.6.2 D-type Algorithm

Figure 4.18 shows the resulting errors for all axes using the pure ILC arrangement (no feedback controller), for various values of $P_{u,0}$. In contrast to the simulation results, use of larger values of $P_{u,0}$ does not lead to appreciable differences in the levels of error produced.

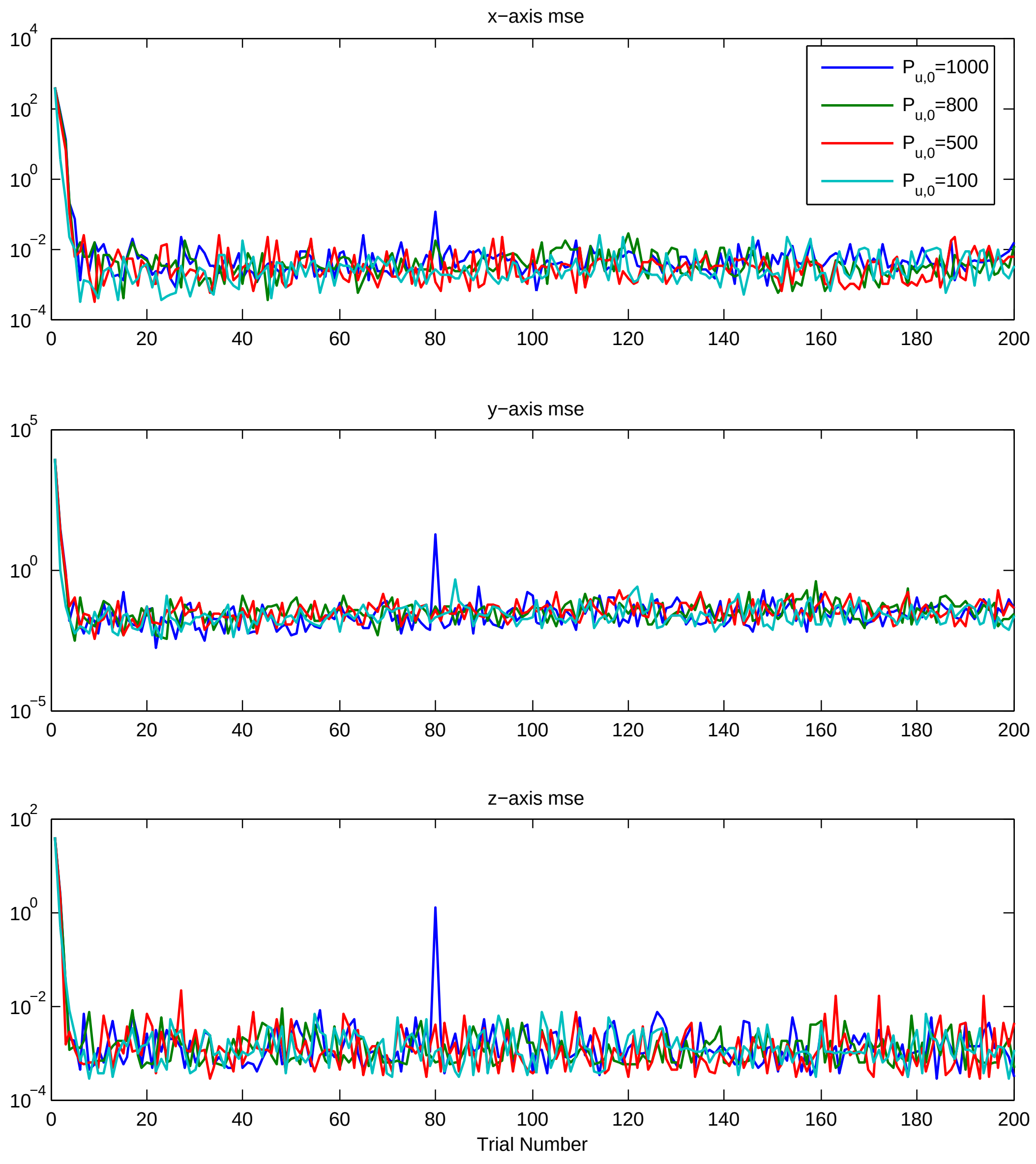


FIGURE 4.18: Experimental results ($Q_t = 0.001, P_{x,0} = 0.1$).

Figure 4.19 shows the errors for all axes again without a PID feedback controller, using various values of Q_t. With smaller values of Q_t, the performance is improved, especially for the Y and Z-axes. However, reducing Q_t further provides progressively less advantage in terms of performance. From the mse curves, it can be seen that the value of Q_t significantly influences the learning speed.

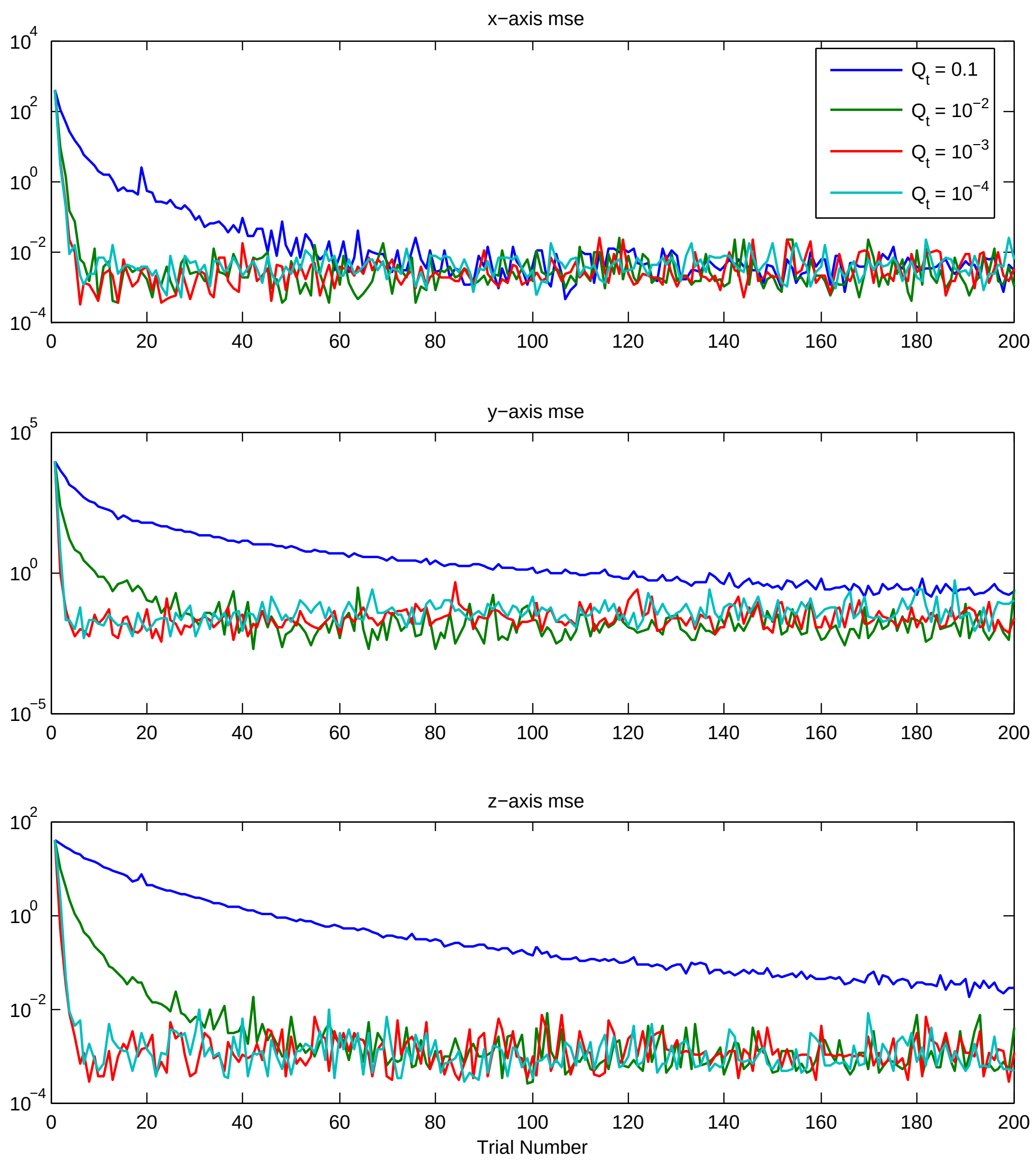


FIGURE 4.19: Experimental results ($P_{u,0} = 100, P_{x,0} = 0.1$).

It has been found that the use of the PID feedback controller provides a higher level of tracking performance over initial trials. This is illustrated by Figure 4.20 in which error plots are given for all axes which show that, with small PID gains, the ILC controller is able to cooperate more effectively with the PID controller. Without the PID controller, the convergence rates for all axes are higher, but the performance in terms of the final level of error is diminished, especially for the Y-axis. One possible reason for this is that the reference signal for this axis is much higher than for the other two.

With the PID feedback controller in place, another series of experiments have been conducted in order to compare the effects of varying $P_{u,0}$. The results are given in Figure 4.21, and it can be seen that the performance for different initial values of $P_{u,0}$ is generally quite similar. Exceptions occur in the case of the Y-axis, where the initial

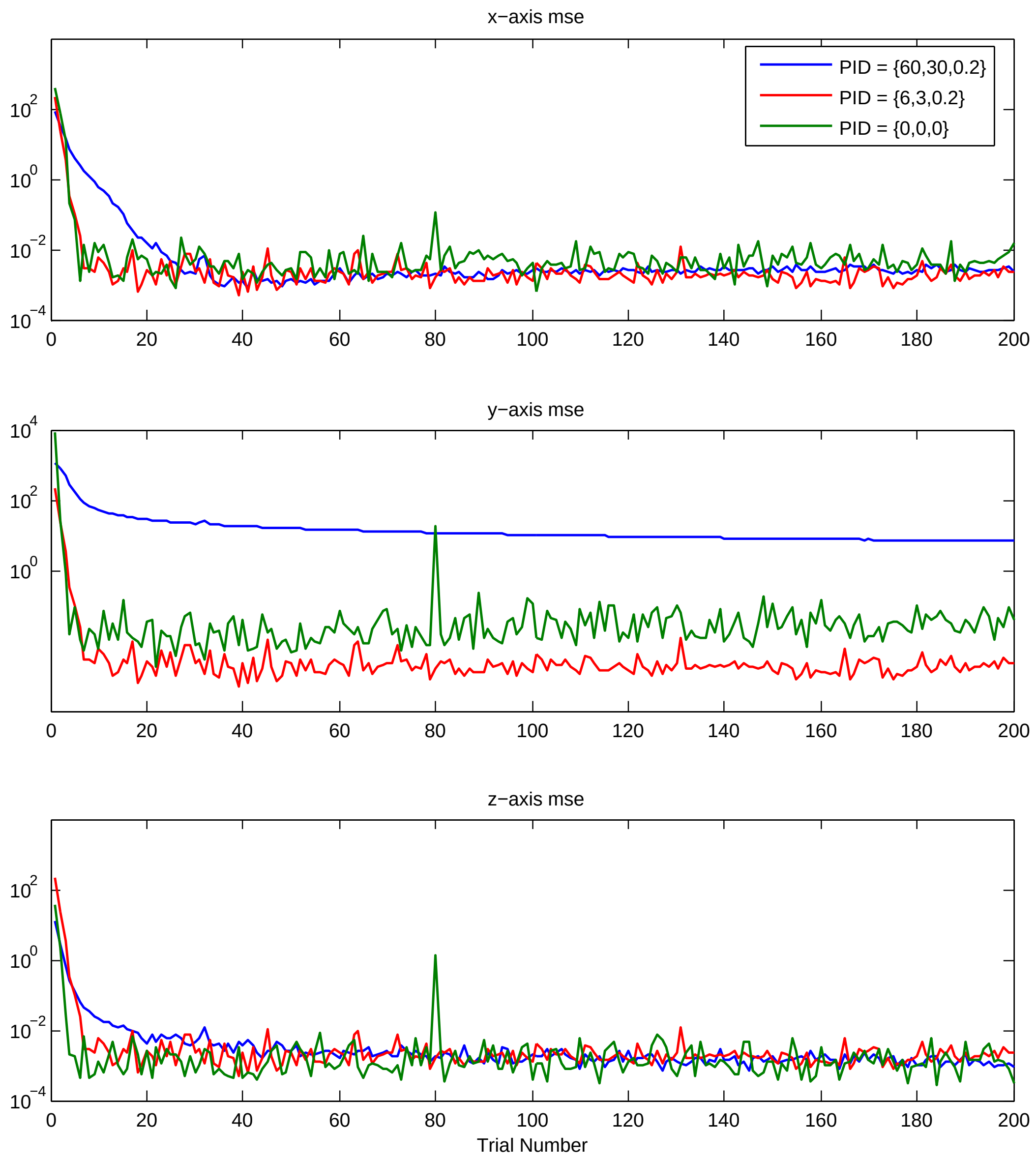


FIGURE 4.20: Experimental results ($P_{u,0} = 1000, Q_t = 0.001, P_{x,0} = 0.1$).

values of $P_{u,0} = 500$ or $P_{u,0} = 800$ clearly improve over those using $P_{u,0} = 100$. These experimental results as a whole suggest that the algorithm has significant robustness and disturbance rejection potential. Note also that very large unexpected errors can arise on some trials (see for example, Figure 4.21, at about trial 100 with $P_{u,0} = 100$, and at about trial 120 with $P_{u,0} = 200$) but overall they do not lead to long-lasting negative effect.

4.6.3 P-type Algorithm

The P-type algorithm has also been tested using the pure ILC arrangement but the learning speeds were extremely slow with various parameter settings. So the following

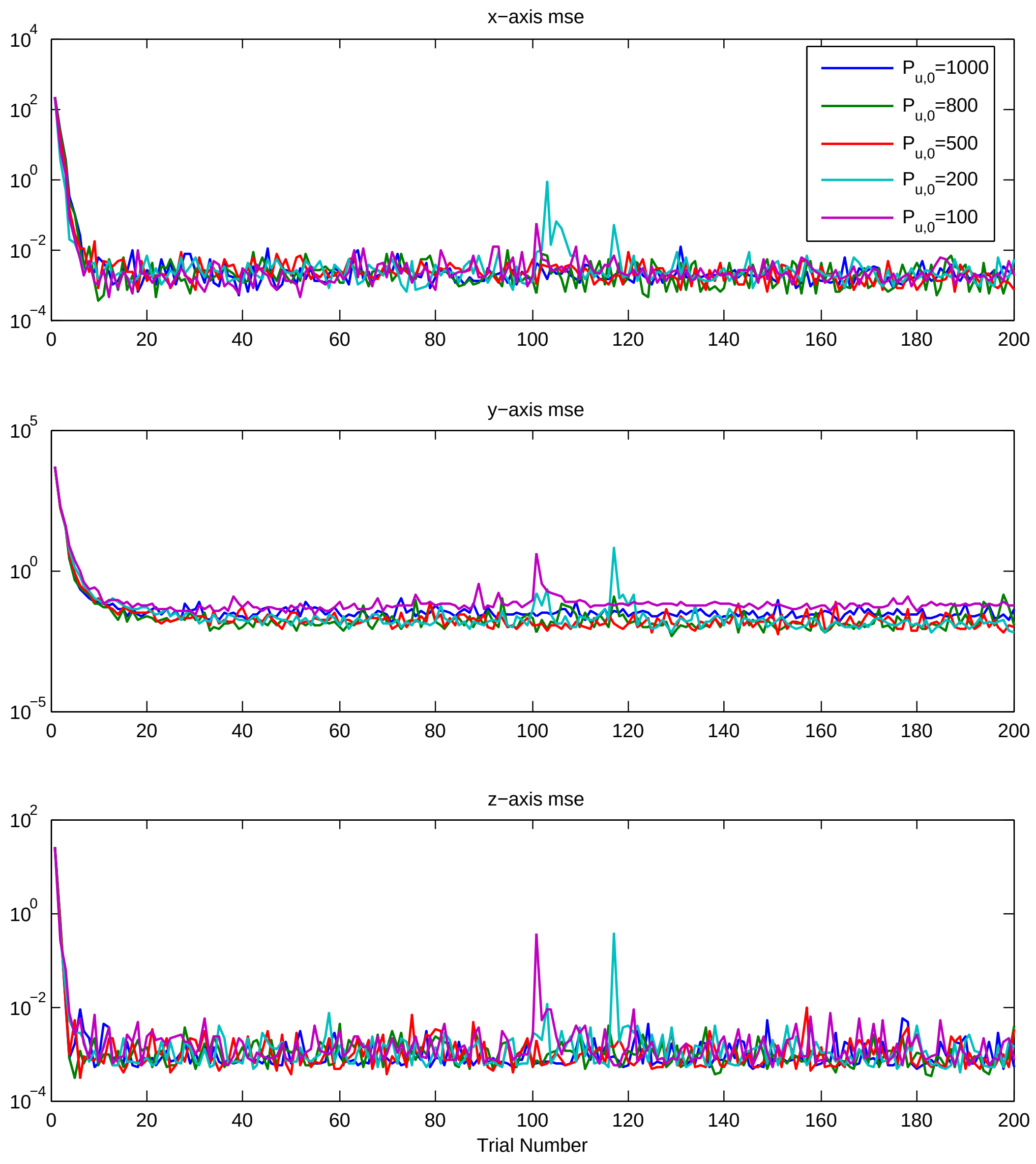


FIGURE 4.21: Experimental results (PID=$\{6, 3, 0.2\}, Q_t = 0.001, P_{x,0} = 0.1$).

results are all generated using the hybrid approach. Figure 4.22 shows a series of experimental results for all axes obtained with the modified filter in place. It is clear that larger initial values of $P_{u,k}$, $P_{u,0}$ provide superior performance, especially for the Y-axis and Z-axis.

Figure 4.23 shows a series of experimental results using various values of Q_t. Although simulation studies indicated that smaller values of Q_t lead to improved performance, experimental results for the X-axis show little difference in practice. Furthermore, the smaller values were not able to provide superior results initially. For the Y-axis and Z-axis, smaller values of Q_t do lead to a reduced level of final error. However, as with the X-axis results, too small a value of Q_t produces poorer performance over initial trials.

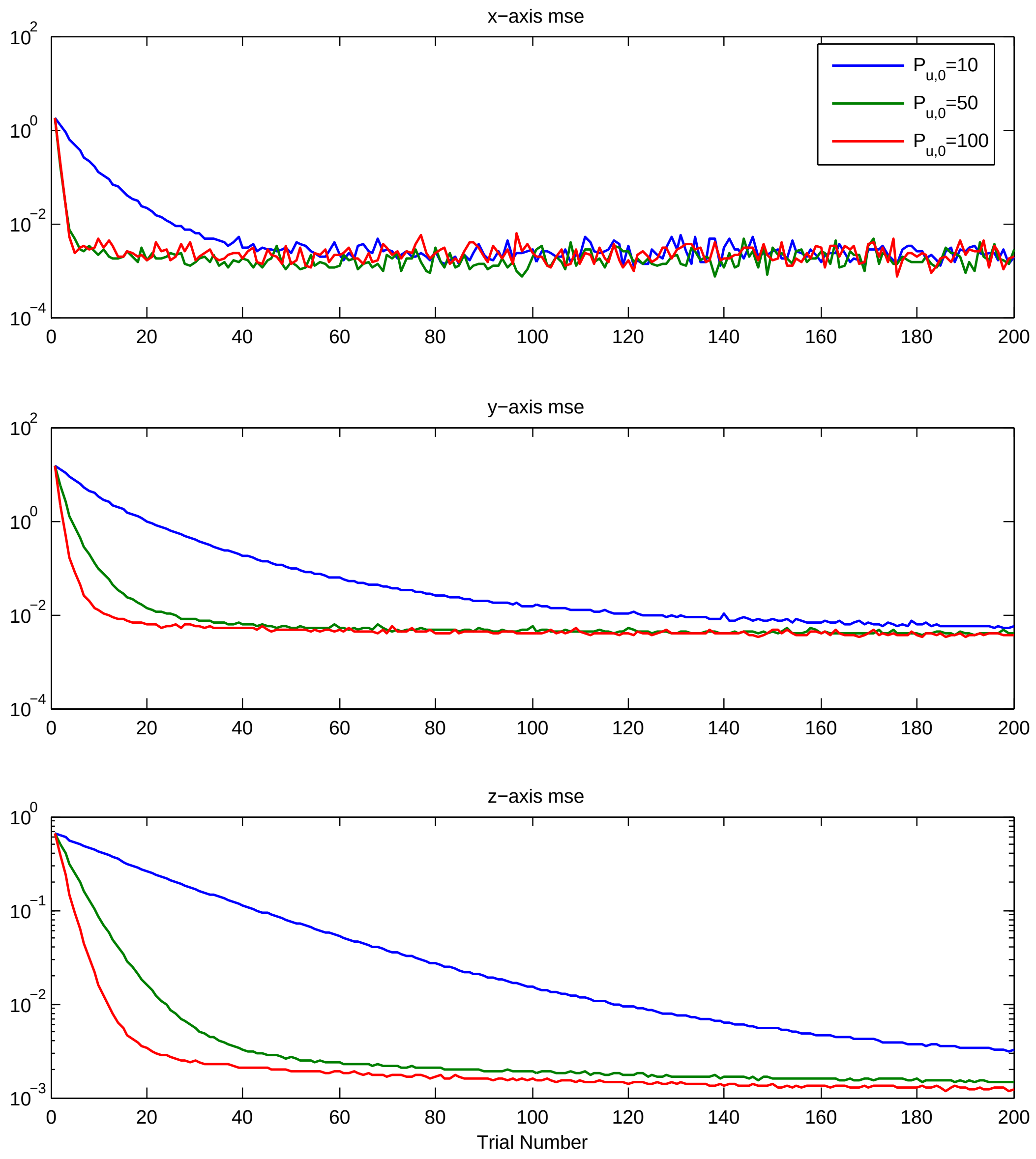


FIGURE 4.22: Experimental results (PID=$\{600, 300, 0.2\}$, $Q_t = 0.1$, $P_{x,0} = 0.1$).

4.6.4 Sub-optimal Algorithm

The sub-optimal algorithm has been implemented using the parameters that where found to lead to high level performance in the simulation study (see Section 4.5.2). In the simulated results, the best performance was achieved by setting the value of $P_{u,0} = 100$ and the value of $Q_t = 0.01$. Then using the fixed choice of $P_{u,0} = 100$ or $Q_t = 0.01$, a series of tests has been carried out. Figure 4.24 and Figure 4.25 show that increasing $P_{u,0}$ and decreasing Q_t by a certain amount is able to improve the performance. However, increasing the ratio of $P_{u,0}$ and Q_t too much will negatively affect the performance. The best performance can be found when $P_{u,0} = 200$ and $Q_t = 0.01$. When using a larger ratio of $P_{u,0}$ and Q_t, convergence speeds are improved while long term performance are not so good as shown in Figure 4.24 which shows mse curves using $P_{u,0} = 500$ cross

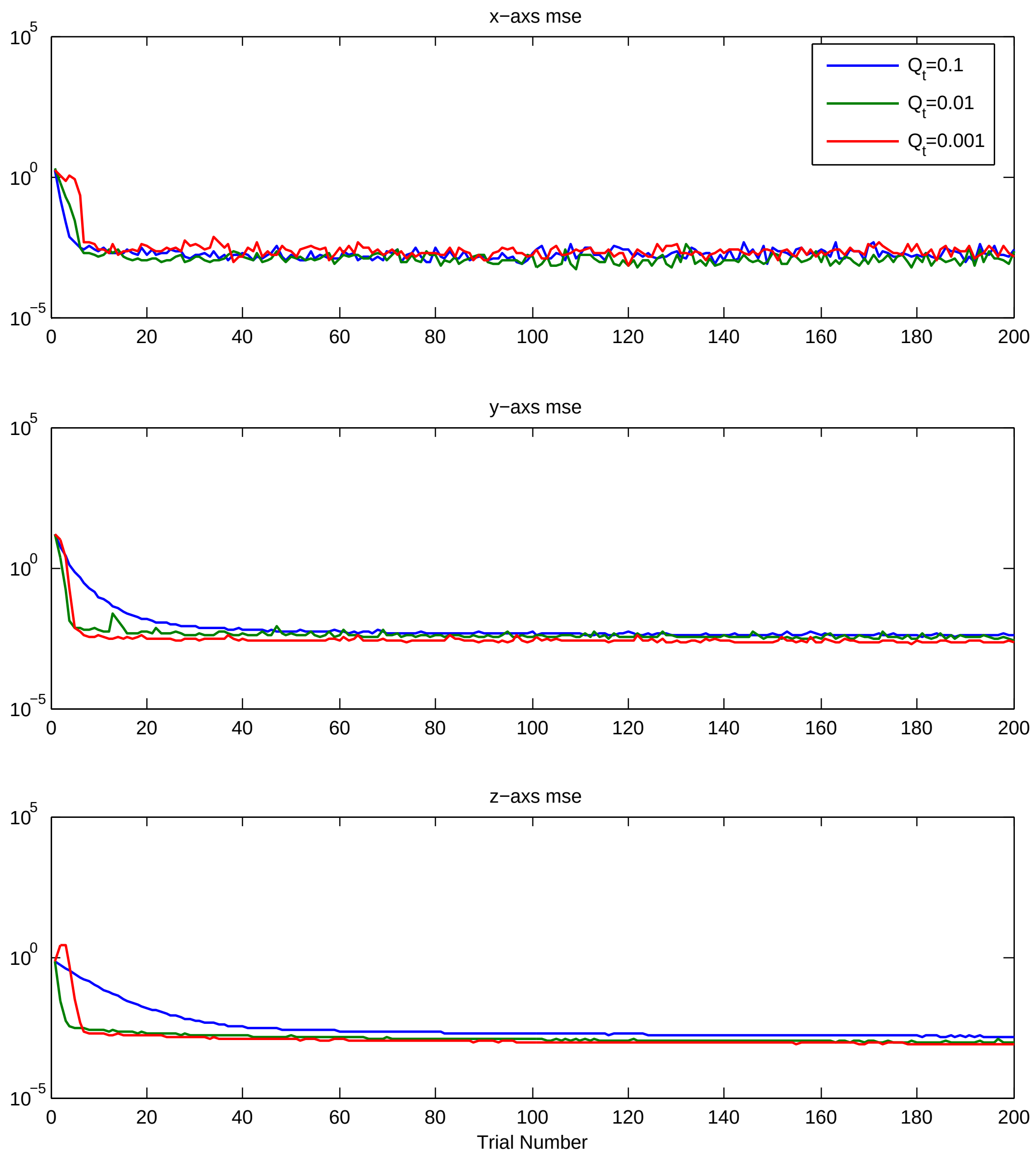


FIGURE 4.23: Experimental results (PID=$\{600, 300, 0.2\}$, $P_{u,0} = 50$, $P_{x,0} = 0.1$).

the curves of $P_{u,0} = 200$ for all axes. The X-axis has the most significant difference. This is also true when using different values of Q_t as shown in Figure 4.25, the mse curve of $Q_t = 0.005$ has crossed the curve of $Q_t = 0.01$ after around 100 trials but this does not occur in the case of Y-axis and the Z-axis whose models are far less complex than that of the X-axis. Therefore, parameter tuning for each axis should be conducted independently (See Appendix B for more experimental results).

4.6.5 Comparison of Experimental Results

The best overall performing results achieved for the D-type, P-type stochastic learning and Sub-optimal algorithms are compared in Figure 4.26. This shows that the D-type/P-

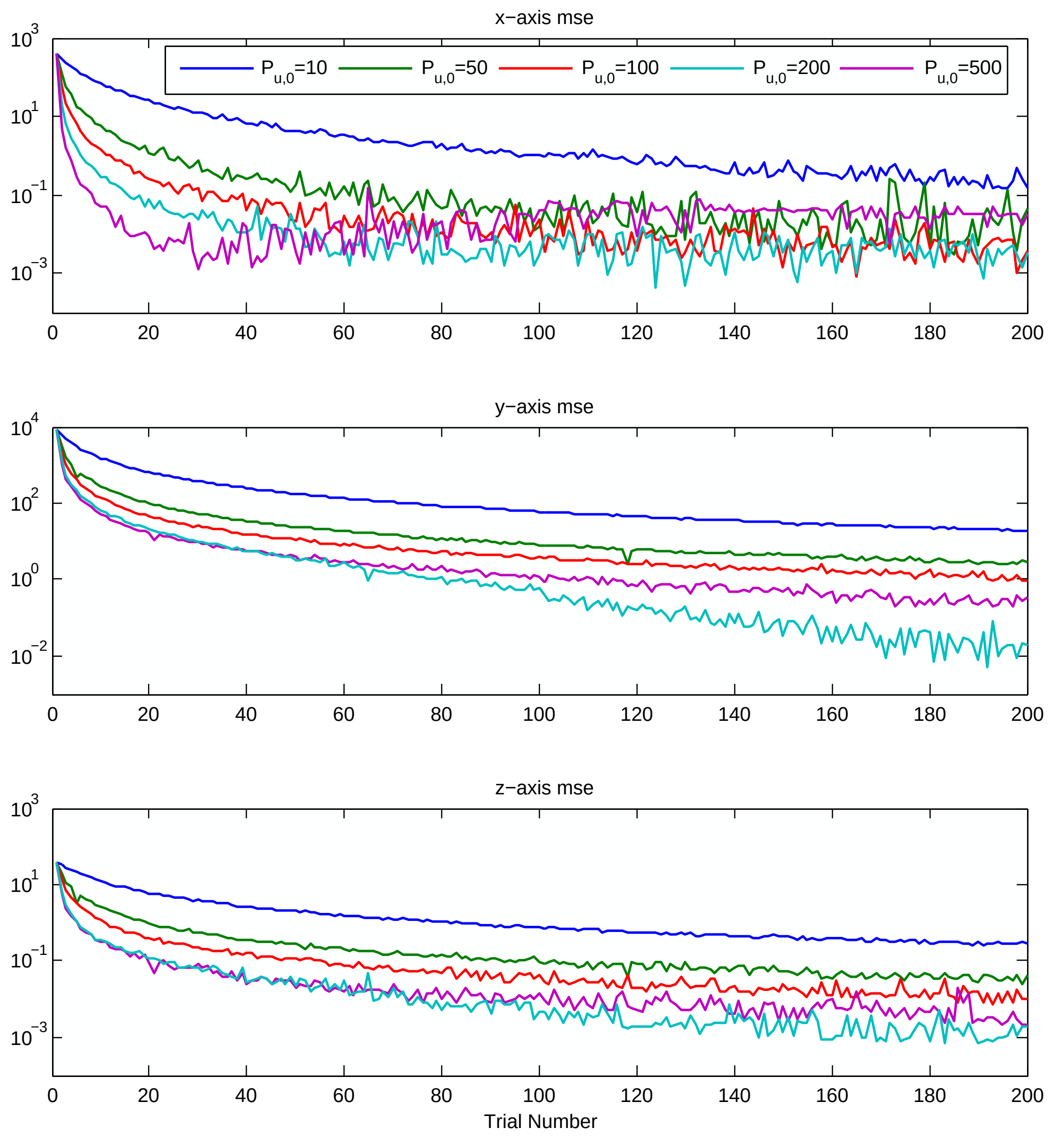


FIGURE 4.24: Experimental results for sub-optimal algorithm ($Q_t = 0.01$).

type stochastic learning algorithm outperforms the sub-optimal algorithm for all three axes in terms of learning speed. Meanwhile, the P-type algorithm produces superior performance for the Y-axis compared with the use of the D-type update. The P-type algorithm slightly improves on the results of the D-type when applied to the X-axis. However, the performance is approximately equal in the case of the Z-axis, but the P-type algorithm holds the final error with significantly less fluctuation.

Compared with other algorithms that have been implemented on the gantry robot, which include the basic P-type algorithm with an aliasing filter, the inverse algorithm, and norm-optimal ILC (see Ratcliffe et al. (2004, 2005a, 2006b) for details), a similar performance is achieved using the discrete stochastic learning algorithms. Figure 4.27 shows that the convergence speed for the stochastic learning algorithms is much more rapid than the P-type with an signal aliasing filter, and slightly slower than when using

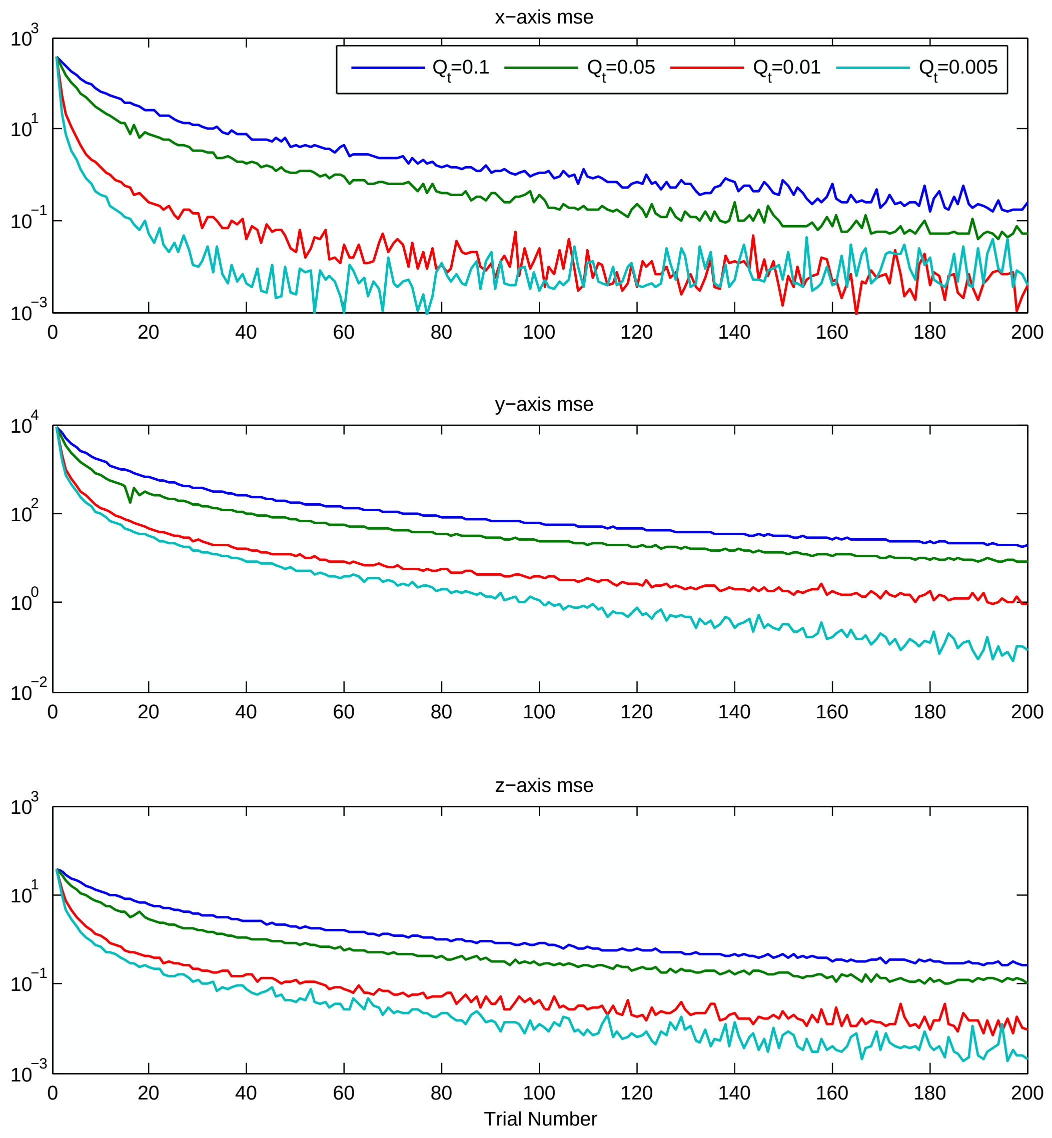


FIGURE 4.25: Experimental results for sub-optimal algorithm ($P_{u,0} = 100$).

norm-optimal ILC and the inverse algorithm. In terms of tracking error, the stochastic learning algorithms outperform the inverse algorithm, but are unable to reach the level attained by norm-optimal ILC. P-type with signal aliasing outperforms the remaining methods in terms of reducing the fluctuation of the final error.

4.7 Summary

In this chapter, one of the experimental facilities — the multi-axis gantry robot has been introduced along with the modelling of all three axes. The mechanical structure and hardware/software setup for algorithm implementation have been provided as well. The settings have been used for all the experiments carried out in this thesis.

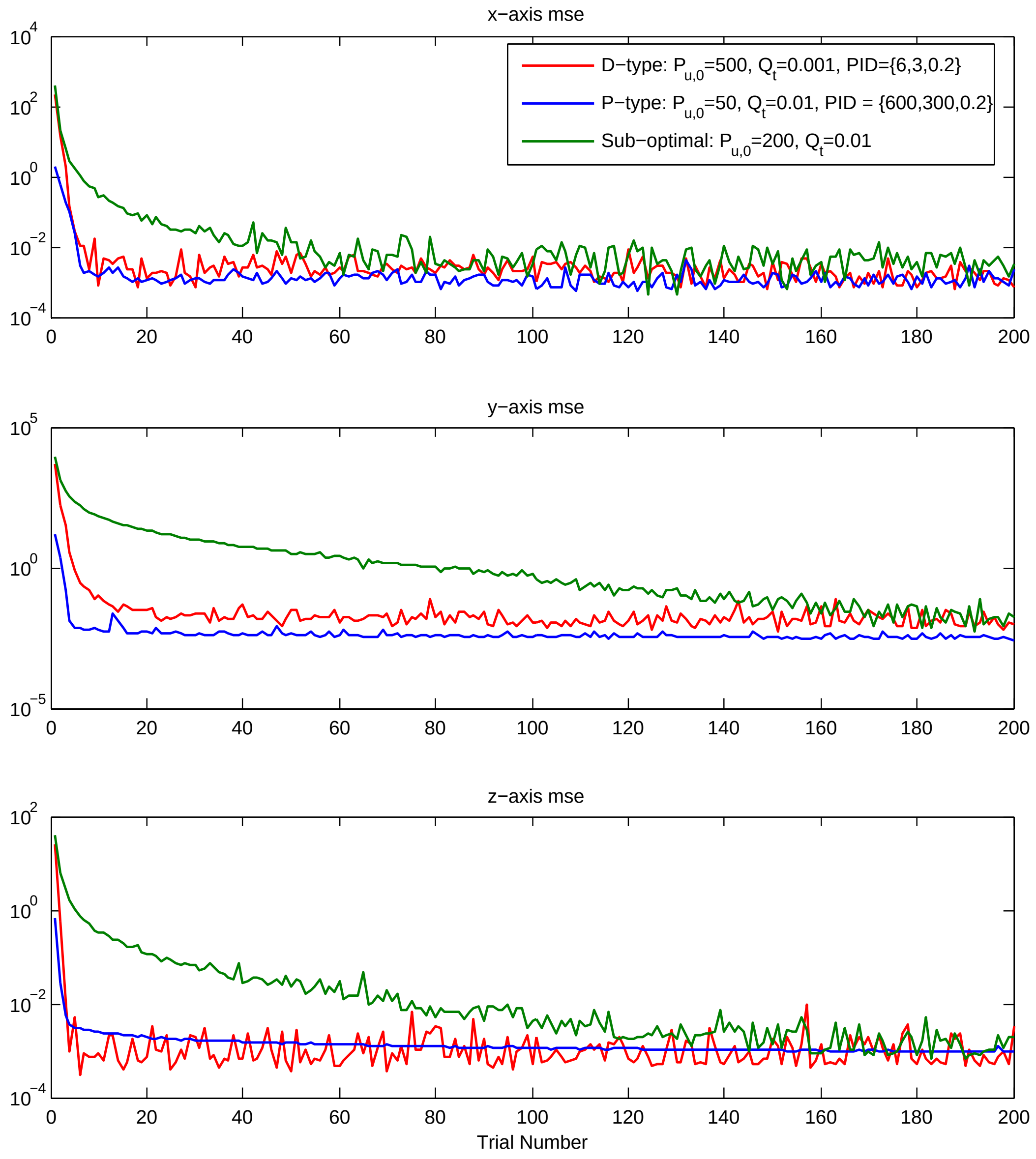


FIGURE 4.26: A comparison of stochastic learning algorithm applied on D-type and P-type .

Also, a series of stochastic learning algorithms (P-type algorithm, D-type algorithm and sub-optimal algorithm) have been reviewed and implemented for on the multi-axis gantry robot. It is the first ever experimental implementation and verification of these stochastic ILC algorithms. Their performance has been assessed as well. The ILC controller has been combined with a PID feedback controller in a parallel arrangement. Experimental results have shown that the highest level of performance is achieved when the PID controller is tuned using small parameter values. Moreover, it is found that use of a robustness filter applied to the error signal prior to the ILC update greatly improves upon the performance of previous implementations. In comparing the algorithms working with P-type ILC and D-type ILC, it has been found that the performance of the P-type stochastic learning algorithm was slightly superior to that of the D-type stochastic

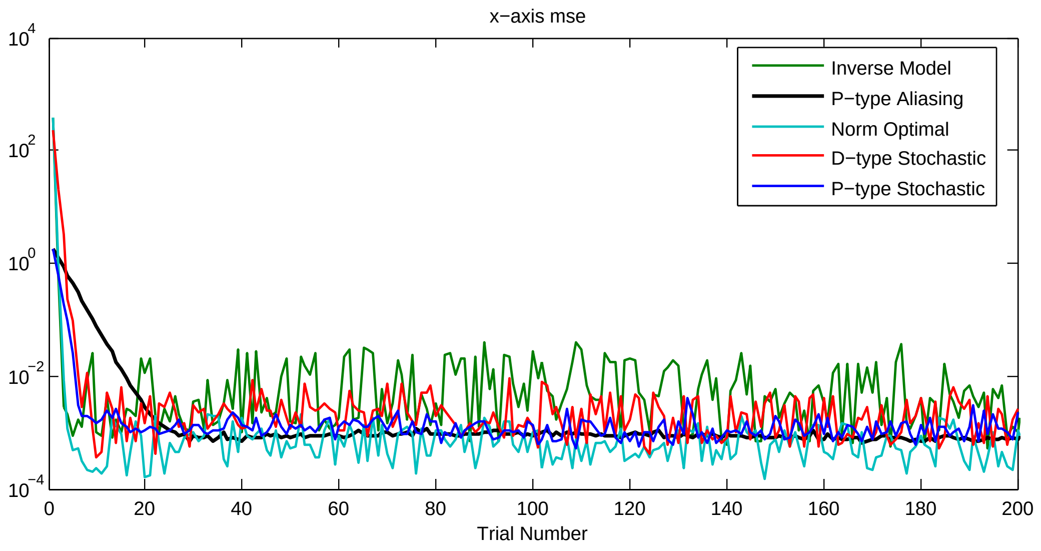


FIGURE 4.27: Comparison of MSE for x-axis with other algorithms.

learning algorithm. Both these two approaches perform better than the sub-optimal approach in terms of learning speed. Furthermore, when compared with other ILC algorithms implemented on the same system, it has been found that the performance of the stochastic learning algorithm compares favourably, it only being eclipsed by the far more computationally intensive norm-optimal ILC algorithm.

Chapter 5

ILC based on 2D System Theory

The formulation of ILC specifies that all related control signals and measurements are defined over a finite time interval which is called a pass, a trial or an iteration. ILC itself is a learning process conducted over a series of trials. Therefore ILC can be easily described in a 2-dimensional (2D) setting. The first dimension is time (or samples in discrete problems), and the second dimension is the trial number. This is advantage, since 2D systems have a significant amount of supporting theory in terms of convergence, stability and related properties. In this chapter, some ILC algorithms based on the 2D systems theory will be developed and implemented. Some analysis is given to illustrate how these approaches deal with the tracking and stabilisation problem in both the dimensions of time and trial number.

5.1 Introduction

In Roesser (1975), the following 2D state space model was introduced as an extension of the well-known standard state space approach

$$\begin{aligned} \begin{bmatrix} x^h(i+1,j) \\ x^v(i,j+1) \end{bmatrix} &= A \begin{bmatrix} x^h(i,j) \\ x^v(i,j) \end{bmatrix} + Bu(i,j) \\ y(i,j) &= C \begin{bmatrix} x^h(i,j) \\ x^v(i,j) \end{bmatrix} \end{aligned} \tag{5.1}$$

where x^h and x^v are the horizontal and vertical state components and i and j are nonnegative integer-valued horizontal and vertical coordinates. Later in Fornasini and

Marchesini (1978), a second order model representing 2D system was introduced as

$$\begin{aligned} x(i+1,j+1) &= A_1x(i,j+1)+A_2x(i+1,j) \\ &\quad +A_0x(i,j)+Bu(i,j) \\ y(i,j) &= Cx(i,j) \end{aligned} \tag{5.2}$$

Here the state updating takes place from (i,j) to $(i+1,j+1)$ in contrast to the shifting operator found in the normal discrete-time state space model. The system is now described in two dimensions and each one can be treated as a standard state space system.

With respect to ILC, one dimension is the sample or time interval during an trial (described by i) while the other dimension j represents the trial number. The ILC algorithm developed by Arimoto et al. (1984a) can be concisely represented in 2-D system theory. As discussed in Chapter 2, ILC can be treated as a 2D linear system where one direction of information propagation in the 2D models above represents the dynamics along the trial and the other the trial-to-trial dynamics. An approach was presented by Geng et al. (1990) which applied 2D system theory to ILC in order to design a controller for the linear multi-variable system. Kurek and Zaremba (1993) then analysed the stability of the basic P-type ILC using 2D system theory and also gave another ILC control law that can be designed based on 2D system theory in the following form

$$u(t,k+1)=u(t,k)-K_1[x(t,k+1)-x(t,k)]+K_2[y_r(t+1)-y(t+1,k)] \tag{5.3}$$

where t,k are the time or sample number along the trial and the trial number respectively, $K_2=(CB)^T[CB(CB)^T]^{-1}$, $K_1=K_2CA$ and $u(t),x(t),y_r(t),y(t)$ denote the control input, the system state, the reference signal and the measured output respectively where A,B,C are the Markov matrices from the model.

A distinct class of 2D systems are called repetitive processes, also called termed multipass processes in the early literature, which are characterised by a series of sweeps, termed passes, through a set of dynamics where the duration, or length, of each pass is finite. An output is produced on each trial which acts as a forcing function and contributes to the next output. The concept of a repetitive process was first introduced in the early 1970s for the modelling of control of long-wall coal cutting and metal rolling operations which are two typical applications in industry (Rogers et al., 2007).

In ILC, a major objective is to achieve convergence of the trial-to-trial error and often this has been treated as the only objective. In fact, it is possible that enforcing fast convergence could lead to unsatisfactory along the trial performance. As an example, to illustrate this last point, consider the case of a linear continuous-time system whose dynamics are modeled by the transfer-function:

$$G(s) = \frac{(s+5)(s+1)}{(s+3)(s^2+4s+29)}$$

which is to be controlled in the ILC setting using the P-type law:

$$u_{k+1} = u_k + Ke_{k+1}(t) \tag{5.4}$$

with, in particular, $K = 3$. Figure 5.1 shows the response of the controlled system over 50 trials when the reference signal is a unit step function of 2 seconds duration, applied at t = 0. Figure 5.2 shows the performance of the controlled system for the 30th trial. These responses confirm that trial-to-trial error convergence occurs but along the trial performance can be very poor.

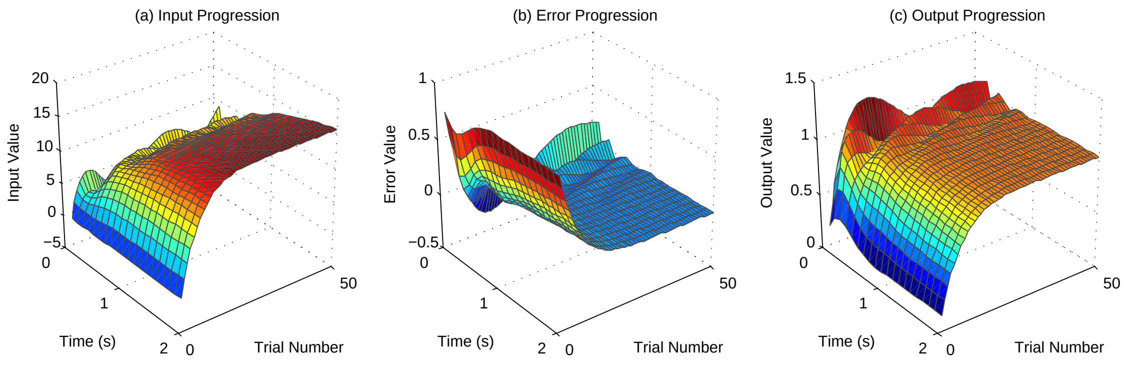


FIGURE 5.1: An example of ILC: learning process

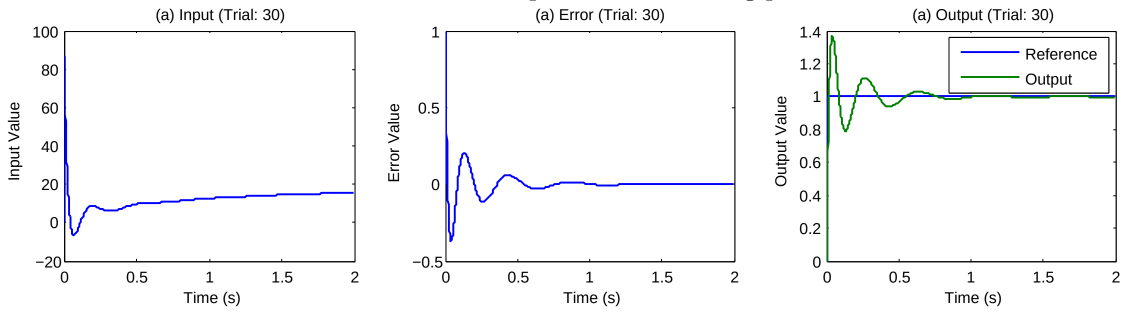


FIGURE 5.2: An example of ILC: individual trial

As another example consider the case where

$$G(s) = \frac{(s+5)(s-1)}{(s+3)(s^2+4s+13)}$$

which is to be controlled in the same ILC setting using the same P-type law as in the previous example (5.4) with the learning gain $K = 3$. Figure 5.3 shows the response of the controlled system over 20 trials when the reference signal is the reference trajectory defined from the gantry robot in Chapter 4, and is of 2 seconds duration. Figure 5.4 shows the performance of the controlled system for the 5th trial. These responses also confirm that trial-to-trial error convergence occurs but along the trial performance is even worse than in the last example. This time the system displays unacceptable along the trial behaviour. In fact treated in isolation as a 1D system, i.e. consider only the trial, the behaviour here is unstable and the dynamics is unacceptable over a finite interval.

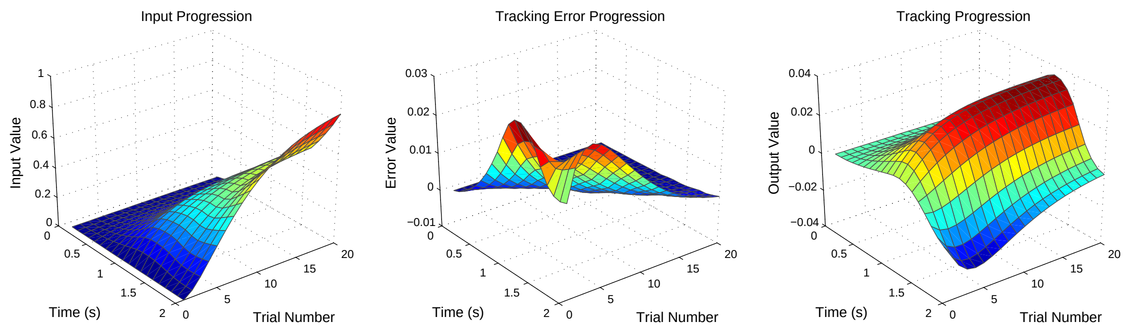


FIGURE 5.3: An example of ILC: learning process

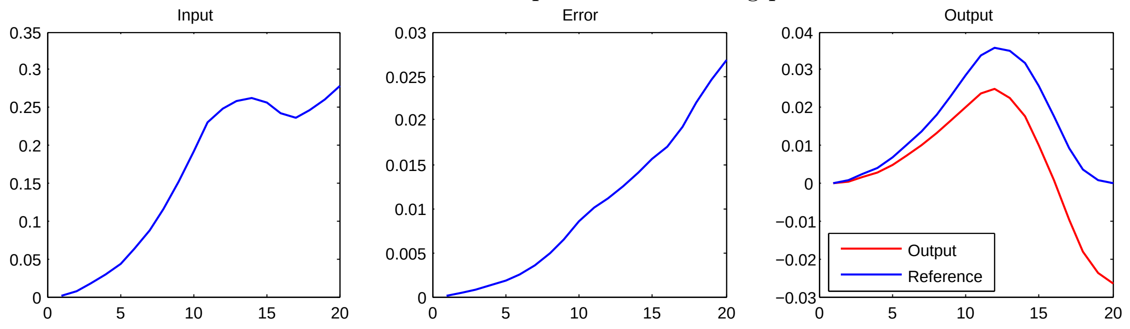


FIGURE 5.4: An example of ILC: individual trial

These two examples demonstrate that ILC sometimes operates well in one dimension (trial to trial) but is far from ideal in the other dimension (along the trial). The first example showed the poor transient performance and the second example illustrated the problem of instability along the trial. The design of control law which can provide good performance in both dimensions would lead to a significant improvement in ILC design.

In this chapter, the problem is addressed by first showing that ILC schemes can be designed for a class of discrete linear systems by, in effect, extending techniques developed for 2D systems using the framework of linear repetitive processes. This allows the use of the strong concept of stability along the pass (or trial) for these processes in an ILC setting as a possible means of dealing with poor/unacceptable transients in the along the trial dynamics. The results developed here yield control law design algorithms which can be implemented via LMIs. The resulting controllers are able to guarantee the stability along the trial and also control over the along the trial dynamics.

There are two sub-algorithms developed and implemented in this chapter, which are the state-feedback 2D algorithm (theoretical details can be found in Hladowski et al.

(2008b)) and the output-feedback 2D algorithm (Hladowski et al., 2008a). Both algorithms are developed based on the same fundamental structure and the output-feedback controller is actually for the case when the states of the system are not observable or can not be measured or computed accurately. Finally, the two resulting control laws are experimentally validated on the gantry robot introduced in last chapter executing a pick and place operation where the plant models used for design are obtained via frequency response tests. The symbols $M \succ 0$, respectively $M \prec 0$, are used in this chapter to denote a symmetric positive definite, respectively negative, definite matrix. The next section gives an overview of the representation and initial analysis of ILC schemes in a repetitive process setting.

5.2 Problem Setup

The plants considered in this chapter are assumed to be differential linear time-invariant systems described by the state-space triple $\{A, B, C\}$ which in an ILC setting is written as

$$\begin{aligned} \dot{x}_k(t) &= Ax_k(t) + Bu_k(t), 0 \le t \le T \\ y_k(t) &= Cx_k(t) \end{aligned} \tag{5.5}$$

where on trial k, $x_k(t) \in \mathbb{R}^n$ is the state vector, $y_k(t) \in \mathbb{R}^m$ is the output vector, $u_k(t) \in \mathbb{R}^r$ is the vector of control inputs, and the trial length $T < \infty$. If the signal to be tracked is denoted by $r(t)$ then $e_k(t) = r(t) - y_k(t)$ is the error on trial k. The most basic requirement then is to force the error to convergence in k. This, however, cannot always be addressed independently of the dynamics along the trial as the following analysis demonstrates.

Consider the case where on trial k+1 the control input is calculated using

$$u_{k+1}(t) := \sum_{j=1}^{M} \alpha_j u_{k+1-j}(t) + \sum_{j=1}^{M} (K_j e_{k+1-j}(t) + (K_0 e_{k+1})) \tag{5.6}$$

In addition to the 'memory' M, the design parameters in this control law are the static scalars α_j , $1 \le j \le M$, the linear operator K_0 which describes the current pass error contribution, and the linear operator K_j , $1 \le j \le M$, which describes the contribution from the error on pass $k + 1 - j$.

It is now routine to show that convergence of the error here holds if, and only if, all roots of

$$z^M - \alpha_1 z^{M-1} - \cdots - \alpha_{M-1} z - \alpha_M = 0 \tag{5.7}$$

have modulus strictly less than unity. Also the error dynamics on trial k+1 here can be written in convolution form as

$$e_{k+1}(t) = r(t) - (Gu_{k+1})(t), 0 \leq t \leq T$$

Suppose also that (5.7) holds. Then the closed-loop error dynamics converge (in the norm topology of $L_p[0, T]$) to

$$e_\infty = (I + GK_{\text{eff}})^{-1} r \tag{5.8}$$

where the so-called effective controller K_{eff} is given by

$$K_{\text{eff}} := \frac{K}{1-\beta}$$

and

$$\beta := \sum_{i=1}^{M} \alpha_i, \quad K = \sum_{i=1}^{M} K_i$$

Also, suppose that the condition of Equation (5.7) holds. then the resulting error sequence is bounded by an expression of the form

$$||\hat{e}_k - \hat{e}_\infty|| \leq M_1 \{\max(||e_0||, \cdots, ||e_{M-1}||) + M_2\} \lambda_e^k \tag{5.9}$$

where $\hat{e}_k = [e_{k+1-M}^T(t) \cdots e_k^T]^T$ is the so-called error super vector, M_1 and M_2 are positive real scalars, and $\lambda_e \in (\max|\mu_{\text{i}}|, 1)$ and μ_i, $1 \leq i \leq M$, is a solution of (5.7).

The result here counter-intuitive in the sense that stability is largely independent of the plant and the controllers used. This is a direct result of the fact that the trial duration T is finite and over such an interval a linear system can only produce a bounded output irrespective of its stability properties. Hence even if the error sequence generated is guaranteed to converge to a limit, this terminal error may be unstable and/or possibly worse than the first trial error, i.e. the use of ILC has produced no improvement in performance.

We have the following (see Owens et al. (2000) for the details).

1. Convergence is predicted to be 'rapid' if λ_e is small and will be geometric in form, converging approximately with λ_e^k.
2. The limit error is nonzero but is usefully described by a (1D linear systems) unity negative feedback system with effective controller K_{eff} defined above. If $\max_i(|\mu_i|) \to 0+$ then the limit error is essentially the first learning iterate, i.e. use of ILC has little benefit and will simply lead to the normal large errors encountered in simple feedback loops. There is hence pressure to let $\max_i |\mu_i|$ be close to unity when K_{eff} is a high gain controller which will lead (roughly speaking) to small limit errors.

3. Zero limit error can only be achieved if $\sum_{i=1}^{M} \alpha_i = 1$. (This situation — again see Owens et al. (2000) — for the details — is reminiscent of classical control where the inclusion of an integrator (on the stability boundary) in the controller results in zero steady state (limit) error in response to constant reference signals.)

There is a conflict in the above conclusions which has implications on the systems and control structure from both the theoretical and practical points of view. In particular, consider for ease of presentation the case when $K_i = 0, 1 \leq i \leq M$. Then small learning errors will require high effective gain yet GK_0 should be stable under such gains.

To guarantee an acceptable (i.e. stable (as the most basic requirement)) limit error and acceptable along the trial transients, a stronger form of stability must be used. Here we consider the use of so-called stability along the trial (or pass) from repetitive process theory. In effect, this demands convergence of the error sequence with a uniform bound on the along the trial dynamics. We also work in the discrete domain and so assume that the along the pass dynamics have been sampled at a uniform rate T_s seconds to produce a discrete-state space model of the form (where for notational simplicity the dependence on T_s is omitted from the variable descriptions)

5.3 State-feedback Control Scheme

5.3.1 Algorithm Setup

Consider the system model is given by

$$\begin{aligned} x_k(p+1) =& \ \hat{A}x_k(p) + \hat{B}u_k(p), 0 \leq p \leq \alpha \\ y_k(p) =& \ \hat{C}x_k(p) \end{aligned} \tag{5.10}$$

Consider now the so-called discrete linear repetitive processes described by the following state-space model over $p = 0, 1, \cdots, T-1, k \geq 1$

$$\begin{aligned} x_k(p+1) =& \ \hat{A}x_k(p) + \hat{B}u_k(p) + \hat{B}_0 y_{k-1}(p) \\ y_k(p) =& \ \hat{C}x_k(p) + \hat{D}u_k(p) + \hat{D}_0 y_{k-1}(p) \end{aligned} \tag{5.11}$$

where $x_k(p) \in \mathbb{R}^n$, $u_k(p) \in \mathbb{R}^r$, $y_k(p) \in \mathbb{R}^m$ are the state, input and pass profile vectors respectively. Note here the information of the previous pass has contribution to the current pass profile. Also rewrite the state equation of the process model in the form

$$x_k(p) = Ax_k(p-1) + Bu_k(p-1) \tag{5.12}$$

and introduce

$$\begin{aligned} \eta_{k+1}(p+1) &= x_{k+1}(p) - x_k(p) \\ \Delta uk+1(p) &= u_{k+1}(p) - u_k(p) \end{aligned} \tag{5.13}$$

Then we have

$$\eta_{k+1}(p+1) \quad = \quad A\eta_{k+1}(p) + B\Delta uk + 1(p-1) \tag{5.14}$$

Consider also a control law of the form

$$\Delta uk + 1(p) = K_1\eta_{k+1}(p+1) + K_2 e_k(p+1) \tag{5.15}$$

and hence

$$\eta_{k+1}(p+1) = (A + BK_1)\eta_{k+1}(p) + BK_2 e_k(p) \tag{5.16}$$

Also $e_{k+1}(p) - e_k(p) = y_k(p) - y_{k+1}(p)$ and we then obtain

$$e_{k+1}(p) - e_k(p) \quad = \quad CA(x_k(p-1) - x_{k+1}(p-1)) \tag{5.17}$$

$$+CB(u_k(p-1) - u_{k+1}(p-1)) \tag{5.18}$$

Using (5.35) we now obtain

$$e_{k+1}(p) - e_k(p) = -CA\eta_{k+1}(p) - CB\Delta uk + 1(p-1) \tag{5.19}$$

or, using (5.15),

$$e_{k+1}(p) \quad = \quad -C(A + BK_1)\eta_{k+1}(p) \tag{5.20}$$

$$-C\eta_{k+1}(p) + (I - CBK_2)e_k(p) \tag{5.21}$$

Also introduce

$$\begin{aligned} \hat{A} &= A + BK_1 \\ \hat{B}_0 &= BK_2 \\ \hat{C} &= -C(A + BK_1) \\ \hat{D}_0 &= I - CBK_2 \end{aligned} \tag{5.22}$$

Then clearly (5.16) and (5.20) can be written as

$$\begin{aligned} \eta_{k+1}(p+1) &= \hat{A}\eta_{k+1}(p) + \hat{B}_0 e_k(p) \\ e_{k+1}(p) &= \hat{C}\eta_{k+1}(p) + \hat{D}_0 e_k(p) \end{aligned} \tag{5.23}$$

which is of the form of linear repetitive processes (see Equation (5.11)) and hence the repetitive process stability theory can be applied to this ILC control scheme. In particular, stability along the trial is equivalent to uniform bounded input bounded output stability (defined in terms of the norm on the underlying function space), i.e. independent of the trial length,and hence we can (potentially) achieve trial-trial error convergence with acceptable along the trial dynamics.

The stability theory for linear repetitive processes is critically dependent on the structure of the boundary conditions and, in particular, the state initial vector sequence Here we assume that

$$\eta_{k+1}(0) = 0, \ k \geq 0$$

It is also possible to write the ILC scheme in the form consider by Kurek and Zaremba (1993) - see (5.3). The results which follow move beyond only trial-to-trial convergence analysis. Moreover they are supported by experimental results which also formed one the few currently reported experimental applications of repetitive processes control theory.

5.3.2 Stability Analysis and State Feedback Design

By using the stability theory in linear repetitive processes from Rogers et al. (2007), the following result gives stability along the trial under control action together with formulas for control law design. The proof of this result is in Hladowski et al. (2008b).

Theorem 5.1. *The ILC scheme of (5.23) is stable along the trial if there exist compatibly dimensioned matrices $X_1 \succ 0$, $X_2 \succ 0$, R_1 and R_2 such that the following LMI is feasible*

$$M = \begin{bmatrix} -X_1 & 0 & X_1A^T + R_1^TB^T & -X_1A^TC^T - R_1^TB^TC^T \\ 0 & -X_2 & R_2^TB^T & X_2 - R_2^TB^TC^T \\ AX_1 + BR_1 & BR_2 & -X_1 & 0 \\ -CAX_1 - CBR_1 & X_2 - CBR_2 & 0 & -X_2 \end{bmatrix} \prec 0 \tag{5.24}$$

If (5.24) holds, the control law matrices K_1 and K_2 can be computed using

$$\begin{aligned} K_1 &= R_1X_1^{-1} \\ K_2 &= R_2X_2^{-1} \end{aligned} \tag{5.25}$$

In practical applications, it is often beneficial (or indeed essential) to bound the entries (above or below) in the control law matrices. In the ILC setting, there could well be cases where it is beneficial to keep the entries in the control law matrix K_2 as large as possible. Note, however, that direct manipulation of the entries in K_2 is difficult to achieve in an LMI setting and hence other approaches must be employed. As an example in this latter category is described next drawing on the work of Siljak and Stipanovic (2000) where it was first proposed (in a non ILC setting)). The basic result is that if L and $k_l > 0$ are real scalars subject to the constraint

$$L^2 < k_l \tag{5.26}$$

then in LMI terms this can be written as

$$\begin{bmatrix} -k_l & L^T \\ L & -1 \end{bmatrix} \prec 0 \tag{5.27}$$

This operation can also be applied in the matrix case — the scalar L^2 is replaced by the matrix $L^T L$, k_l by $k_l I$, where I is an identity matrix of compatible dimensions, and less than is replaced by a negative definite constraint.

5.3.3 Simulations and Experiments

The control law has been implemented on the gantry robot described in Chapter 4. The algorithm needs a discrete-time plant model for all three axes, and each model is discretised using a sampling time of $T_s = 0.01$s (sampling frequency: 100Hz) to ensure there is enough time between each time interval for the computation. The discrete-time transfer function for the X-axis is given below (details of the Y-axis and the Z-axis can be found in Chapter 4 or Appendix A).

$$\begin{aligned} G(z) &= \frac{0.00051745(z+0.5823)(z-0.3014)}{(z-1)(z^2-0.07057z+0.009459)} \\ &\quad \cdot \frac{(z^2-0.09718z+0.008969)(z^2-0.2046z+0.7846)}{(z^2+0.3149z+0.1024)(z^2-0.7757z+0.5403)} \end{aligned} \tag{5.28}$$

what leads to the state-space models used for the design (where the subscript X is used to distinguish this axis from the others we are considering)

$$\begin{aligned} A_X &= \begin{bmatrix} 2.41 & -0.86 & 0.85 & -0.59 & 0.30 & -0.19 & 0.32 \\ 4.00 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.00 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.00 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.00 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.25 & 0 \end{bmatrix} \\ B_X &= \begin{bmatrix} 0.0313 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\ C_X &= \begin{bmatrix} 0.0095 & -0.0023 & 0.0048 & -0.0027 & 0.0029 & -0.0011 & 0.0029 \end{bmatrix} \end{aligned}$$

At this stage, Theorem 1 can be used to undertake control law design, where it should be noted that the LMI setting actually produces a family of solutions. As one example, we consider the case when the following additional LMI constraints are imposed (where for this particular case X_2 is a scalar)

$$X_2 < 1 \times 10^{-4} \tag{5.29}$$

$$X_1 \prec 1 \times 10^{-2} \tag{5.30}$$

$$R_1 \prec 1 \times 10^{-2} \tag{5.31}$$

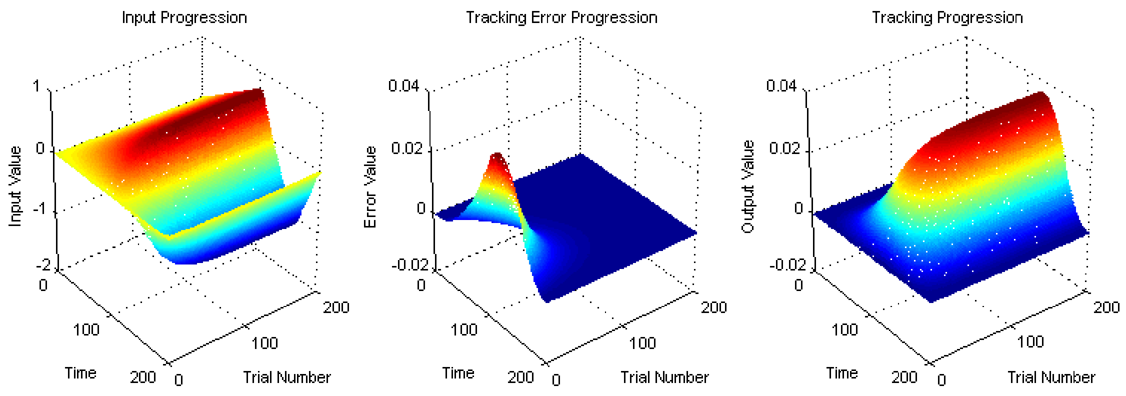


FIGURE 5.5: Simulation results for the state-feedback algorithm

The control law matrices of (5.15) for this data are given by

$$K_1 = \begin{bmatrix} 7.3451 & -2.7245 & 0.1499 & 7.6707 & 2.7540 & -3.6088 & -20.4519 \end{bmatrix} \tag{5.32}$$

$$K_2 = 82.4119 \tag{5.33}$$

Here the system states are obtained by a Kalman state estimator. As it is assumed that the system is a discrete SISO system in the form of (5.10), by setting the initial state to zero, the state can be computed sample by sample when a system model is known. The estimator has the following state equation:

$$x(p+1) = Ax(p) + Bu(p) + L[y(p) - Cx(p)]$$

Where L is the Kalman filter gain matrix which is derived by solving a discrete Riccati equation. In the simulations and experiments, the filter gain matrix L is computed in Matlab and

$$L = \begin{bmatrix} 0.0004 & 0.0004 & 0.0034 & 0.0041 & 0.0025 & 0.0017 & 0.0020 \end{bmatrix}$$

Informative simulation studies are a critical stage prior to the experiments as it is essential to verify that algorithm performs well and to ensure there are no harmful signals which would be applied to the experimental facilities. Here the controlled process has been simulated in Matlab/Simulink and typical results are shown in Figure 5.5. These show that convergence to zero error is possible. The speed of convergence is somewhat slow but, as stated previously, the LMI design produces a family of control laws and hence allows the possibility of tuning these designs to good effort in this case.

Having done some initial experimental tests, it was found that some high frequency components build up in this control design as they did in the implementation of the stochastic learning algorithms (see Chapter 5 for details). One option to limit this is to employ a zero-phase low pass filter to remove such noise (and retain stability along the trial). Figure 5.6 shows a case where the trial error without filtering starts to diverge after (approximately) 100 trials but the addition of a filter of this type is able to maintain (this aspect of overall) performance.

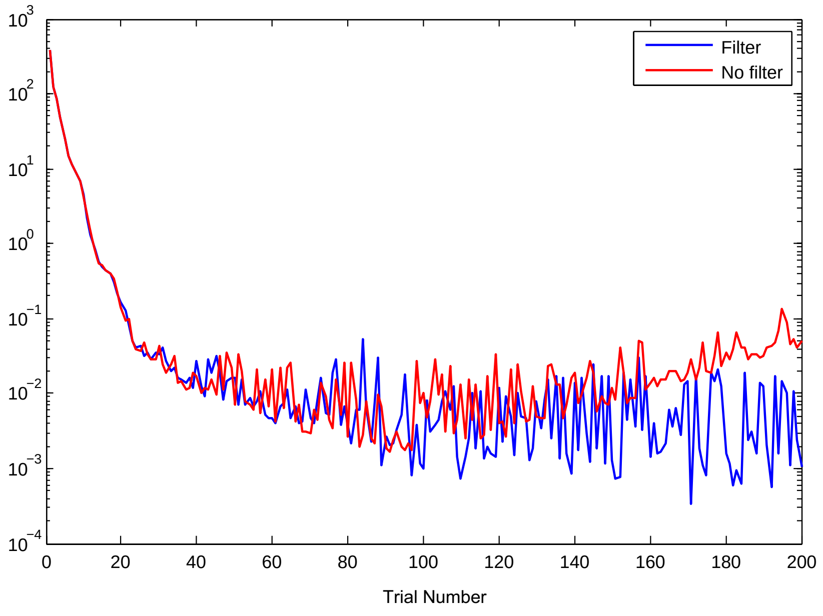
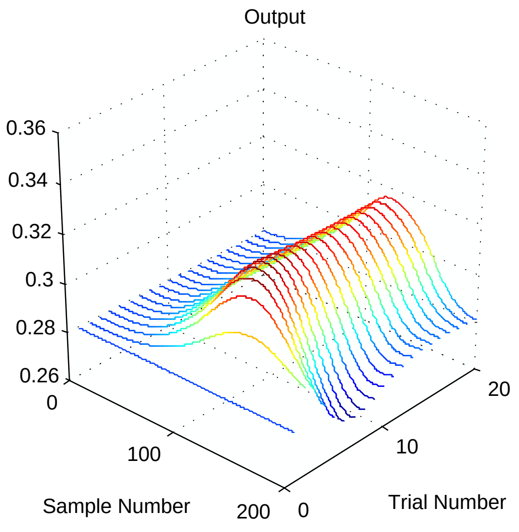


FIGURE 5.6: The effect of filtering

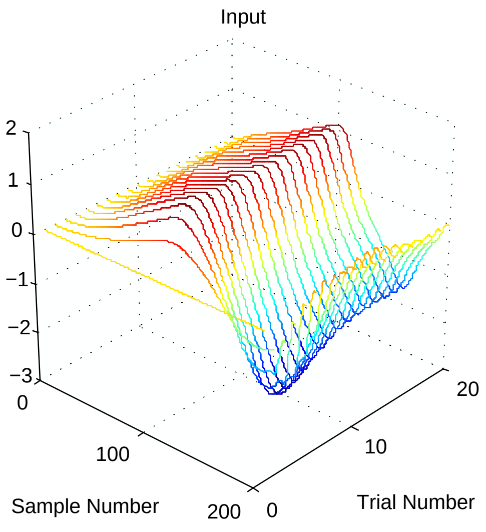
After applying the zero-phase filter, a series of experiments has been carried out on the gantry robot. Figure 5.7(a) shows the outputs produced by the X-axis of the gantry for 20 trials. Figure 5.7(b) and Figure 5.7(c) show the corresponding control input and error dynamics. It can be seen that within only 20 trials, the tracking error has been reduced to a very small value.

The along-the-trial performance is shown in Figure 5.8 which provides the results of the 200th trial recorded in one of the experiments. It is obvious that there is no sign of instability from the plots, and that the tracking performance is acceptable.

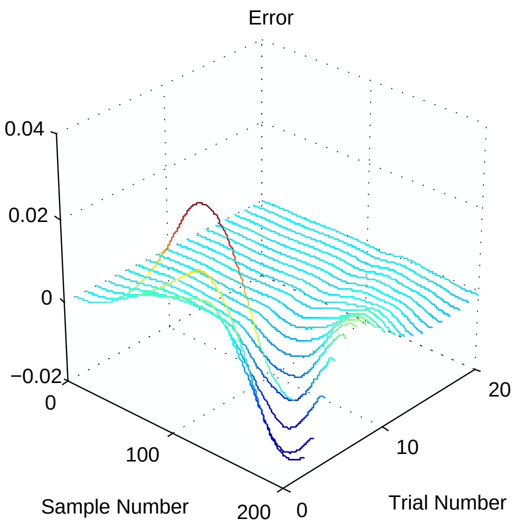
After the success of the initial test programme on the X-axis, the design exercise has been repeated for the Y (perpendicular to the X-axis in the same plane) and Z (perpendicular to the X-Y plane)-axes. Figure 5.9 shows the mean square error for all axes in comparison to those from a simulation study for each corresponding axis with the ILC control law applied. More detailed results of the Y-axis and Z-axis (i.e. along the trial performance and learning progression mesh plots) can be found in Appendix B.



(a)



(b)



(c)

FIGURE 5.7: (a) Evolution of the output of X-axis for the first 20 trials in one experiment, (b) evolution of the control input, (c) evolution of the error dynamics.

5.4 Output-feedback Control Scheme

The control law of the previous sub-section employs state feedback and hence leads to the requirement that all elements in the vector can be measured for implementation. In this section, control laws that do not require measurement of the state vector are developed and experimentally tested and, hence there is no need for an observer.

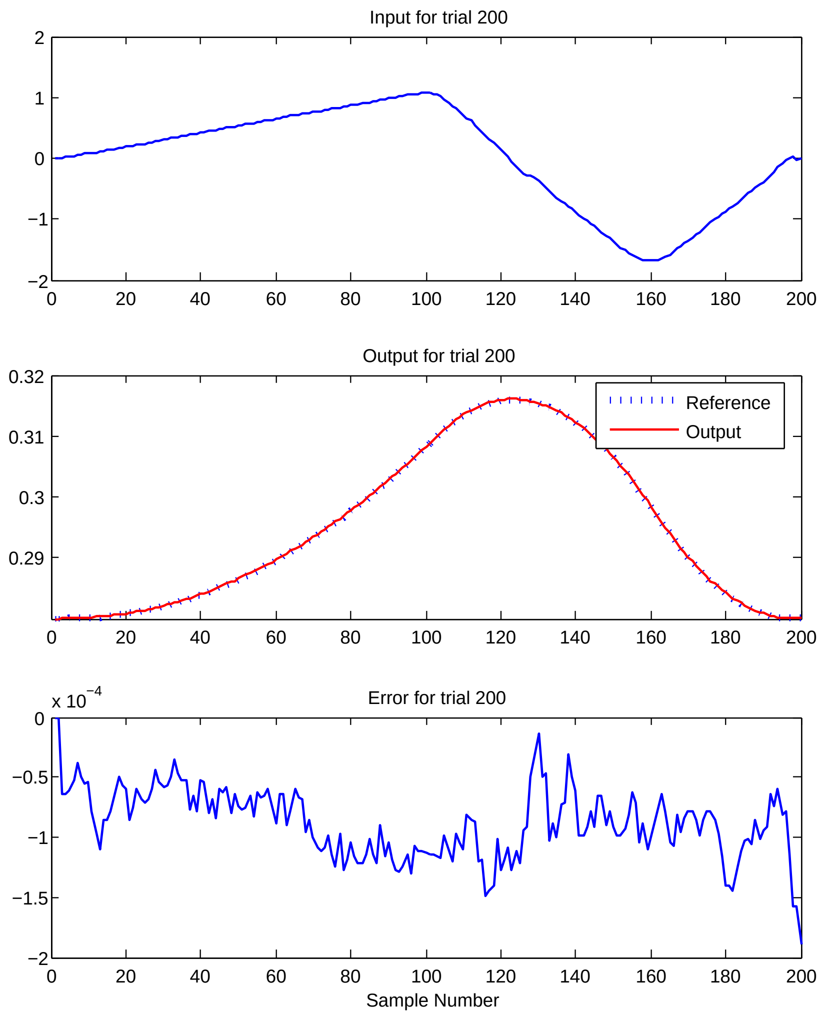


FIGURE 5.8: Along the trial performance for State-feedback 2D ILC algorithm (X-axis).

5.4.1 Algorithm Setup

Define the error signal $e_k(p)$ on trial k as

$$e_k(p) = y_{ref}(p) - y_k(p) \tag{5.34}$$

where $y_{ref}(p)$ again denotes the reference signal to be learnt. Now introduce

$$\begin{aligned} \eta_{k+1}(p+1) &= x_{k+1}(p) - x_k(p) \\ \Delta u_{k+1}(p) &= u_{k+1}(p) - u_k(p) \end{aligned} \tag{5.35}$$

Then it is possible to proceed as in the last section (or see for example Hladowski et al. (2008b)) and use an ILC law which requires the current trial state vector $x_k(p)$ of the plant. In practical applications, this vector may not be available for measurement or, at best, only some of its entries are and hence in general an observer will be required. here

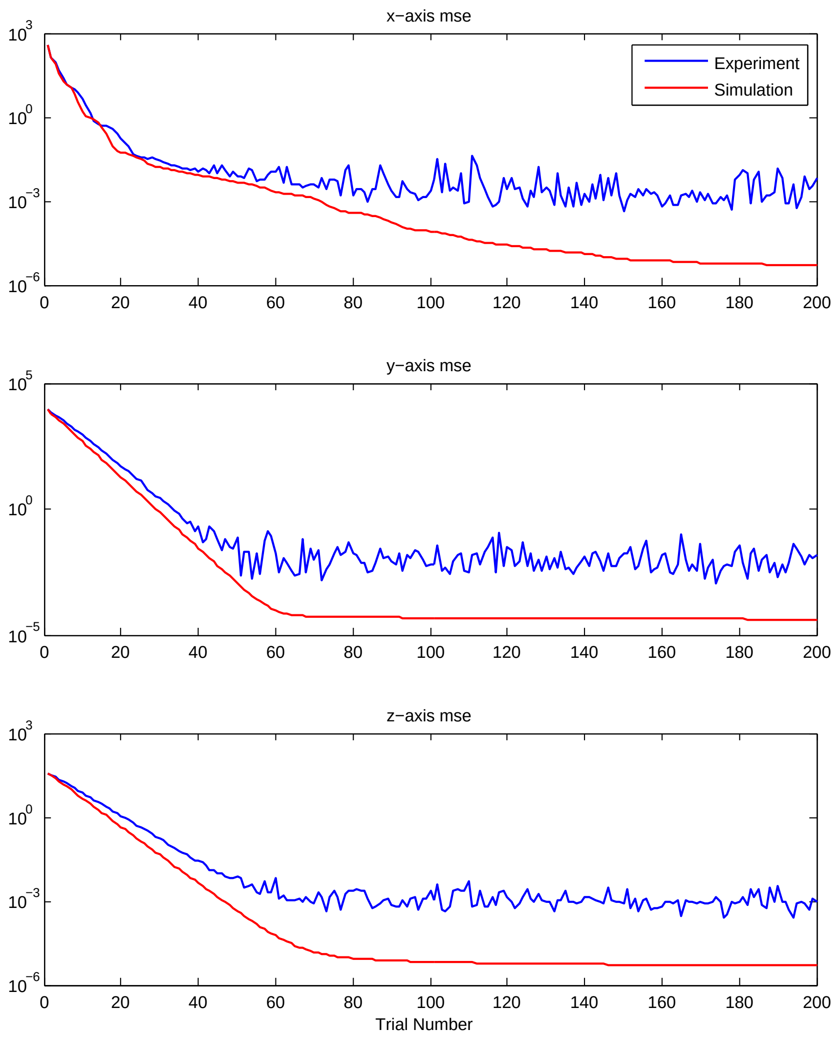


FIGURE 5.9: Mean squared error for all axes

we avoid the use of an observer by using the control law

$$\Delta u_{k+1}(p) = K_1\mu_{k+1}(p+1) + K_2\mu_{k+1}(p) + K_3e_k(p+1) \tag{5.36}$$

where

$$\mu_k(p) = y_k(p-1) - y_{k-1}(p-1) = C\eta_k(p) \tag{5.37}$$

The extra term $\mu_{k+1}(p)$ (i.e. the output difference of the previous sample) in the control law considered here has been added as a means, if necessary, of compensating for the effects of replacing pure state information.

Using (5.34) and (5.37), we can write (5.36) as

$$\Delta u_{k+1}(p-1) = K_1C\eta_{k+1}(p) + K_2C\eta_{k+1}(p-1) + K_3e_k(p) \tag{5.38}$$

and hence

$$\begin{aligned} \eta_{k+1}(p+1) &= (A+BK_1C)\eta_{k+1}(p) \\ &\quad +BK_2C\eta_{k+1}(p-1) \\ &\quad +BK_3e_k(p) \\ e_{k+1}(p) &= (-CA-CBK_1C)\eta_{k+1}(p) \\ &\quad -CBK_2C\eta_{k+1}(p-1) \\ &\quad +(I-CBK_3)e_k(p) \end{aligned} \tag{5.39}$$

Substituting

$$\tilde{\eta}_{k+1}(p+1) = \begin{bmatrix} \eta_{k+1}(p+1) \\ \eta_{k+1}(p) \end{bmatrix} \tag{5.40}$$

in (5.39) now gives

$$\begin{aligned} \tilde{\eta}_{k+1}(p+1) &= \hat{A}\tilde{\eta}_{k+1}(p) + \hat{B}_0 e_k(p) \\ e_{k+1}(p) &= \hat{C}\tilde{\eta}_{k+1}(p) + \hat{D}_0 e_k(p) \end{aligned} \tag{5.41}$$

where

$$\begin{aligned} \hat{A} &= \left[\begin{array}{c|c} A+BK_1C & BK_2C \\ I & 0 \end{array}\right] \\ \hat{B}_0 &= \begin{bmatrix} BK_3 \\ 0 \end{bmatrix} \\ \hat{C} &= \left[\begin{array}{c|c} -CA-CBK_1C & -CBK_2C \end{array}\right] \\ \hat{D}_0 &= (I-CBK_3) \end{aligned} \tag{5.42}$$

which is of the form (5.11) and hence the repetitive process stability theory can be applied to this ILC control scheme. In particular, stability along the trial is equivalent to uniform bounded input bounded output stability (defined in terms of the norm on the underlying function space), i.e. independent of the trial length,and hence we can (potentially) achieve trial-trial error convergence with acceptable along the trial dynamics.

5.4.2 Stability Analysis and Output Feedback Design

Similarly, using the stability theory of 2D systems is able to provide the analysis for this control scheme. The stability theory (Rogers et al., 2007) for linear repetitive processes consists of two distinct concepts. Asymptotic stability for (5.11) holds if, and only if, $\mathrm{r}(\hat{D}_0) < 1$. Also if this property holds and the control input sequence applied $\{\hat{u}_k\}_k$ converges strongly to u_∞ as $k \to 1$ then the resulting output pass profile sequence $\{\hat{y}_k\}_k$ converges strongly to $\hat{y}_\infty$ — the so-called limit profile defined (with $\hat{D} = 0$ for ease of presentation) over $0 \le p \le \alpha - 1$ by

$$\hat{x}_\infty(p+1) = (\hat{A} + \hat{B}_0(I-\hat{D}_0)^{-1}\hat{C})x_1(p) \tag{5.43}$$

$$+Bu_\infty(p) \tag{5.44}$$

$$\hat{y}_1(p) = (I-\hat{D}_0)^{-1}\hat{C}\hat{x}_\infty(p), \tag{5.45}$$

$$\hat{x}_1(0) = \hat{d}_1 \tag{5.46}$$

where $\hat{d}_\infty$ is the strong limit of the sequence d_k. In effect, this result states that if a process is asymptotically stable then its repetitive dynamics can, after a sufficiently large number of passes, be replaced by those of a 1D differential linear system. Note, however, that this property does not guarantee that the limit profile is stable as a 1D discrete linear system, i.e. $\mathrm{r}(\hat{A} + \hat{B}_0(I - \hat{D}_0)^{-1}\hat{C}) < 1$ — a point which is easily illustrated by the case when $\hat{A} = -0.5$, $\hat{B} = 0$, $\hat{B}_0 = 0.5 + \beta$, $\hat{C} = 1$, $\hat{D} = 0$, $\hat{D}_0 = 0$ and $\beta > 0$ is a real scalar such that $|\beta| \geq 1$.

The reason why asymptotic stability does not guarantee a limit profile which is stable along the pass is due to the finite pass length. In particular, asymptotic stability is easily shown to be bounded-input bounded-output (BIBO) stability with respect to the finite and fixed pass length. Also in cases where this feature is not acceptable, the stronger concept of stability along the pass must be used. In effect, for the model (5.11), this requires that the BIBO stability property holds uniformly with respect to the pass length α.

Theorem 5.2. *A discrete linear repetitive process described by (5.11) is stable along the pass if there exist matrices $Y \succ 0$ and $Z \succ 0$ such that the following LMI holds*

$$\begin{bmatrix} Y - Z & * & * \\ 0 & -Z & * \\ \hat{A}_1 Y & \hat{A}_2 Y & -Y \end{bmatrix} \prec 0 \tag{5.47}$$

where

$$\hat{A}_1 = \begin{bmatrix} \hat{A} & \hat{B}_0 \\ 0 & 0 \end{bmatrix}, \hat{A}_2 = \begin{bmatrix} 0 & 0 \\ \hat{C} & \hat{D}_0 \end{bmatrix} \tag{5.48}$$

The proof of this result can be found in Rogers et al. (2007).

Now we have the following result

Theorem 5.3. *An ILC scheme described by (5.41) is stable along the trial if there exist matrices $Y \succ 0$, $Z \succ 0$, N_1, N_2 and N_3 such that the following LMI with linear constraints holds*

$$\begin{gathered}\begin{bmatrix} Z - Y & * & * \\ 0 & -Z & * \\ \Omega_1 & \Omega_2 & -Y \end{bmatrix} \prec 0 \\ CY_1 = PC \\ CY_2 = QC \end{gathered} \tag{5.49}$$

where

$$Y = \begin{bmatrix} Y_1 & 0 & 0 \\ 0 & Y_2 & 0 \\ 0 & 0 & Y_3 \end{bmatrix} \tag{5.50}$$

and

$$\Omega_1 = \left[\begin{array}{c|c|c} AY_1 + BN_1C & BN_2C & BN_3 \\ Y_1 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right]$$

$$\Omega_2 = \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -CAY_1 - CBN_1C & -CBN_2C & Y_3 - CBN_3 \end{array}\right] \tag{5.51}$$

The matrices P and Q are additional decision variables. If the LMI with equality constraints of (5.49) is feasible, the control law matrices can be calculated using

$$\begin{aligned} \hat{K}_1 &= N_1P^{-1} \\ \hat{K}_2 &= N_2Q^{-1} \\ \hat{K}_3 &= N_3Y_3^{-1} \end{aligned} \tag{5.52}$$

The proof of this result is in Hladowski et al. (2008a)

Note here for a SISO system, the controller K_1, K_2, K_3 are just scalars. The control law (5.38) therefore has a similar form of the previous discussed phase-lead ILC algorithm in Chapter 3. This provides the possibility of tuning the gains in the phase-lead algorithm using LMI approach as it produces a family of available control laws.

5.4.3 Simulations and Experiments

The output-feedback scheme has been experimentally tested on the gantry robot. The condition (5.49) is an LMI with constraints. A freely-available Scilab software is used to solve the LMI problems. In this case a sampling time $T_s = 0.05$s has been used rather than the previous $T_s = 0.01$s since the use of Scilab toolbox is not able to yield a solution to the problem when using $T_s = 0.01$s (although the solution might exist). However the hardware setup of the gantry robot does not support such a sampling frequency, which makes the controller impossible to implement. Also, the frequency response of the axes show that there exists significant difference in the system dynamic between using $Ts = 0.01$s and $Ts = 0.05$s, therefore, even if a zero order hold method can be used to simulate the results of $Ts = 0.05$s when using $Ts = 0.01$s, the measured outputs are not close to what are expected. Therefore a multi-sampling technique has been introduced to cope with the problem.

A multi-sampling technique uses more than one controller in the system loop. Each one runs individually at a sampling frequency of the designed controller. The number of controllers using in the system depends on the designed sampling time and sampling time which is feasible for practical application. For example, in this case, the gantry robot is able to run at 100Hz ($T_s = 0.01$s) and the controller design could only give results when using $T_s = 0.05$s. To attempt to solve this, 5 controllers can be used in order to fill in all required unknown points at the higher sampling (Figure 5.10 shows an example of this technique).

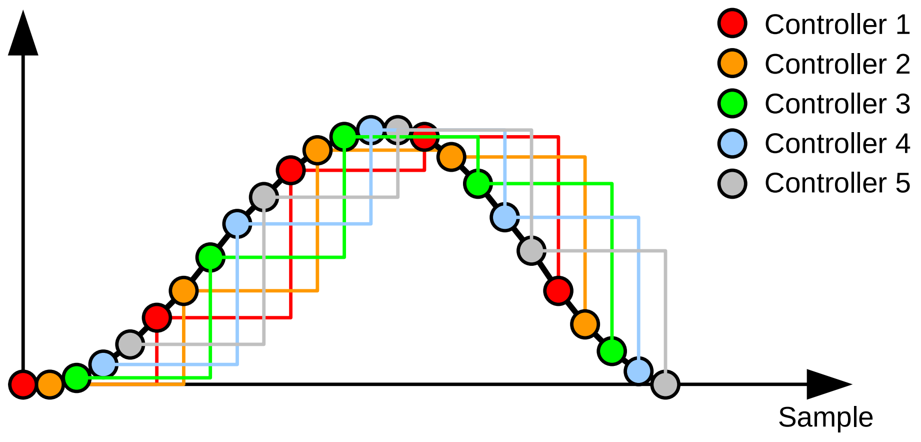


FIGURE 5.10: An example of the multi-sampling technique.

The implementation of the multi-sampling technique is straightforward as each algorithm of the 5 controllers is unchanged from the original one. In this case $y_k(p-1)$ is required, this being the output value of the previous sample. Therefore, all that required is to simply modify $y_k(p-1)$ to become $y_k(p-N)$ in the control program, where N denotes the number of controllers (note the first N points should be set equal to zero as an initial condition).

Another technique which is useful here is the linear interpolation technique. The controller is designed using a lower sampling frequency. The controller computes the value of the key points and the other sample points between each two key points can be found using linear interpolation (see Figure 5.11). Similar ideas have been considered include an aliasing filter by Ratcliffe et al. (2005a). In this case, the controller computes an input

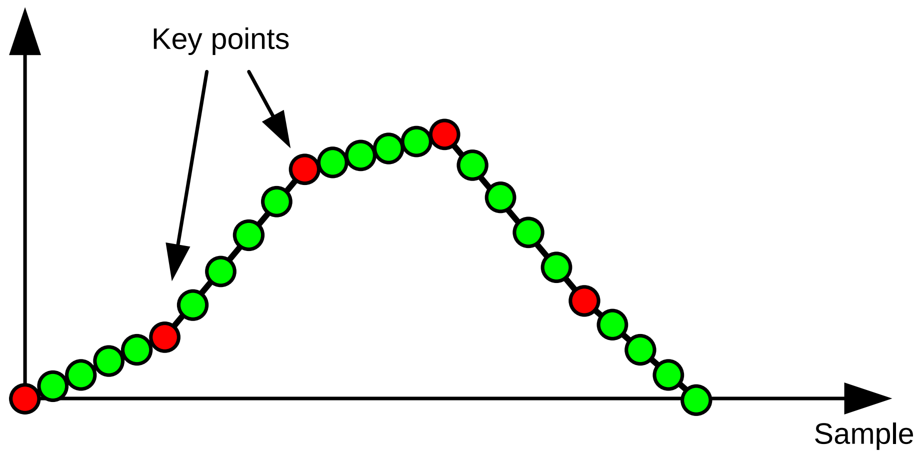


FIGURE 5.11: An example of the linear interpolation.

signal point every 5 points, the other 4 points being generated by linear interpolation. It is not possible to use cubic spline interpolation as at the current point, only two points of data are available.

Simulations have been carried out to evaluate these two methods and the results, are shown in Figure 5.12, show that the multi-sampling method approved produce superior results. From the results, it is clear that the learning speeds using both methods are the same, therefore, the difference between the two methods can be seen only when error is tiny. Using the multi-sampling technique is able to produce more accurate results. One possible reason for this is that the multi-sampling method uses more information than linear interpolation does in this learning process. The linear interpolation method might miss some useful points when trajectory changes suddenly. Therefore, the following results produced on the gantry robot are generated by mean of the multi-sampling method. However, the multi-sampling technique requires further investigation as there is not any formal mathematical formulation which supports this technique.

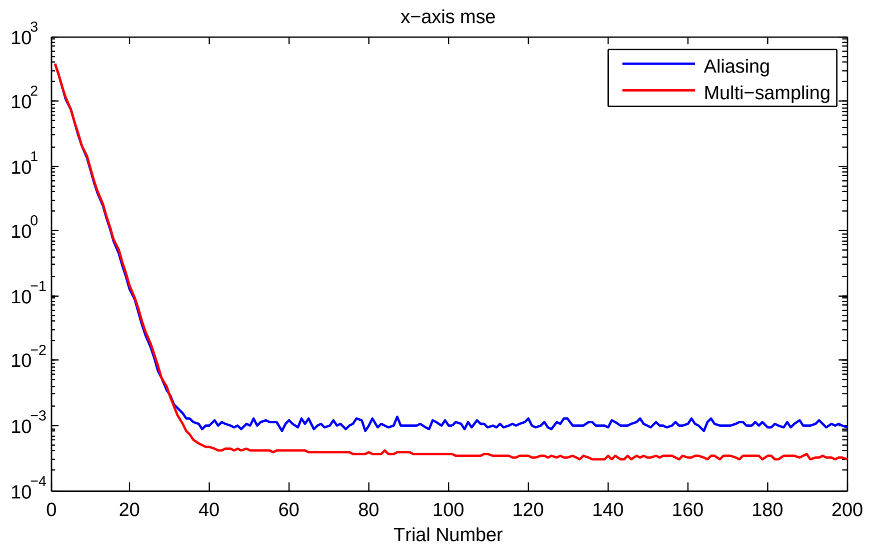


FIGURE 5.12: Comparison of multi-sampling method and linear interpolation.

By converting the continuous transfer function, the state-space matrices of the X-axis used here are as follow:

$$A_X = \begin{bmatrix} 0.0760 & 1.0000 & 0 & 0 & 0 & 0 & 0 \\ -0.0402 & 0.0760 & 0.0001 & -0.0403 & 0 & 0.0047 & 0.1970 \\ 0 & 0 & -0.0018 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.0018 & 0 & 0.0049 & 0.2046 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.0241 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{5.53}$$

$$B_X = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0.0625 \end{bmatrix}^T \tag{5.54}$$

$$C_X = \begin{bmatrix} 0.0101 & -0.0084 & 0.0001 & -0.0137 & 0 & 0.0016 & 0.0671 \end{bmatrix} \tag{5.55}$$

One possible ILC control law of the form (5.38) for this case is

$$\begin{aligned} K_1 &= -250.71696 \\ K_2 &= 116.94689 \\ K_3 &= 47.76410 \end{aligned}$$

In this design, the high-frequency component buildup was observed in the initial tests as well. This caused vibrations and greatly increased the error $e_k(p)$ (as shown in Figure 5.13). To overcome this problem a 6th order zero-phase Chebyshev low pass filter was

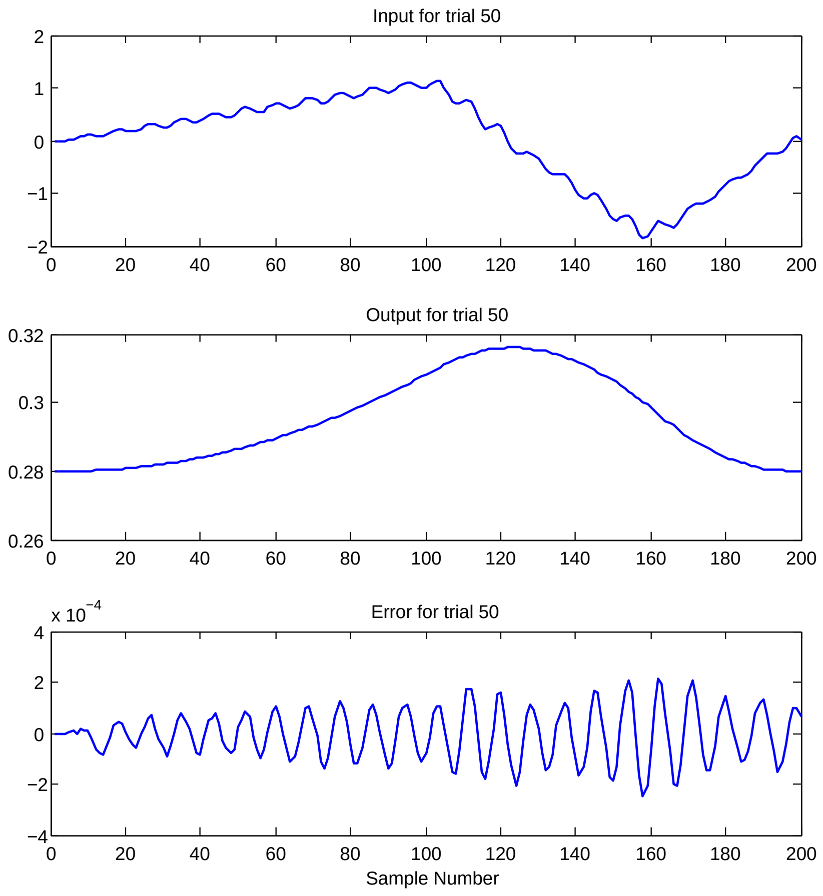


Figure 5.13: The Input/Output/Error signals at 50th trial without filtering

employed. The parameters of the filter were tuned so that the best performance was achieved but at the same time no high-frequency component buildup was observed. It must be stressed that the cutoff frequency of the filter influences the magnitude of obtained error and furthermore influences the performance (see Figure 5.14). Too aggressive a filter will provide a more stable system but the final tracking performance will be degraded as the filter is likely to cut off useful lower frequency information as well.

With a suitable filter applied to the system, better tracking results can be achieved.

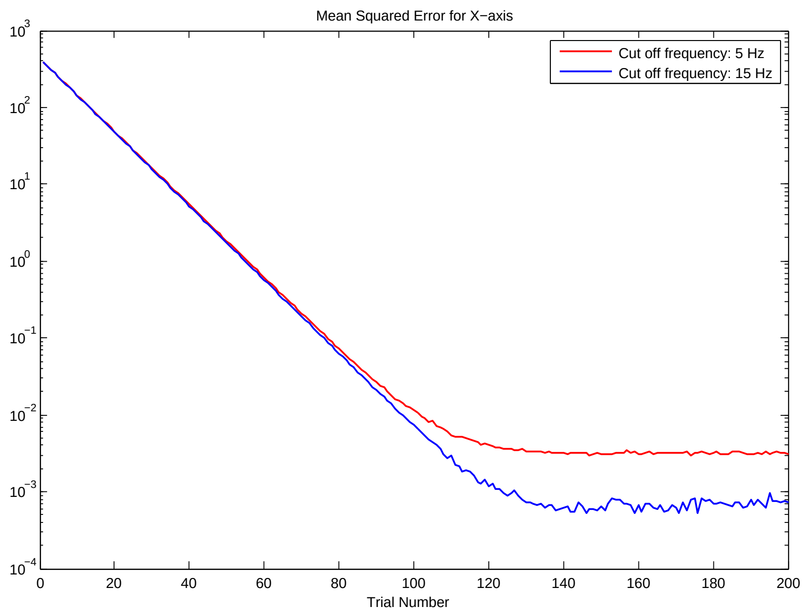


FIGURE 5.14: Influence of cutoff frequency on the error $e_k(p)$

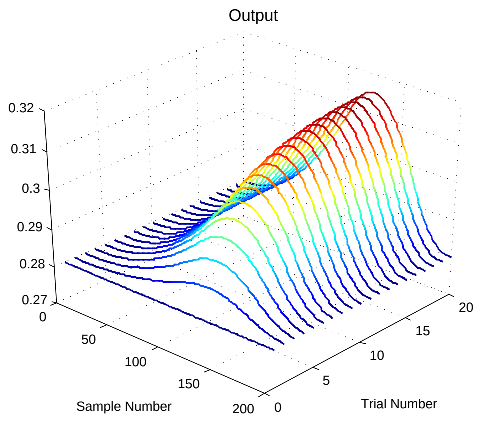
Figure 5.15(a) shows the learning progression of the outputs produced by the X-axis of the gantry over 20 trials. Figure 5.15(b) and Figure 5.15(c) show the corresponding control input and error dynamics.

Long term performance test experiments have then been carried out. No trend of instability has been seen in the learning progress. Figure 5.16 shows the along-the-trial performance from one experiment on the 200th trial. Another problem is indicated by this result: in the error plot, the final part of the error has a sudden jump signal which appears repeatedly and could not be eliminated. Referring back to the ILC control law (5.38), the error with one sample shifted is used to produce the updating input. At the end point there is no error information available and a zero-order-hold signal or an empty point has been used instead .

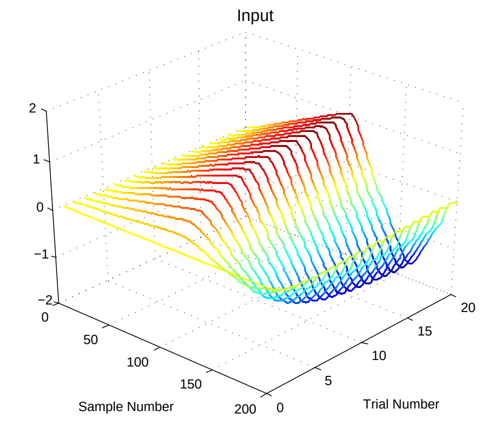
This problem can be easily overcome by extending the reference signal by a few samples of empty points in the end. After this small modification, the overall performance improved obviously (see Figure 5.17 for the detailed input, output and tracking error). Figure 5.18 shows the mse improvement resulting from the extending reference signal. It can clearly be seen that using the extended reference helps not only the final tracking performance, but also the initial convergence.

Adjusting the objective function allows a set of controller matrices to be obtained by solving the LMI of Theorem 5.3. To compare the results, consider the changes of K_3

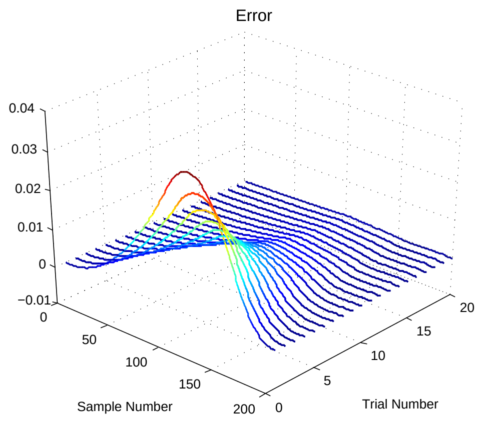
- $K_3 = 17.74083$,$(K_1 = -326.4815, K_2 = 4.3525 \cdot 10^{-13})$
- $K_3 = 47.7641$,$(K_1 = -262.8253, K_2 = 2.87865 \cdot 10^{-11})$



(a)



(b)



(c)

FIGURE 5.15: (a) Evolution of the output of X-axis for the first 20 trials in one experiment, (b) evolution of the control input, (c) evolution of the error dynamics

- $K_3 = 239.375$,$(K_1 = -207.8933, K_2 = -4.7220 \cdot 10^{-7})$

A comparison of the results of the proposed controllers are shown in Figure 5.19. Note that the peak occurs at around the 100th trial, and was caused by an unknown disturbance.

Studying Figure 5.19 reveals that the value of the control matrices influences the obtained result. Increasing the value of K_3 in (5.36) greatly increases the convergence rate of the algorithm but at the same time causes the increase of the final error.

In Hladowski et al. (2009), an approach is provided which leads to a stability condition expressed in terms of an LMI with immediate formulas for computing the control law matrices for the widely encountered case of a SISO linear plant state-space model where the first Markov parameter is zero (i.e. $CB = 0$). Many designs have been proposed for this problem in the literature but none of these have been experimentally verified or are able to deal with any other performance specification beyond trial-to-trial error convergence.

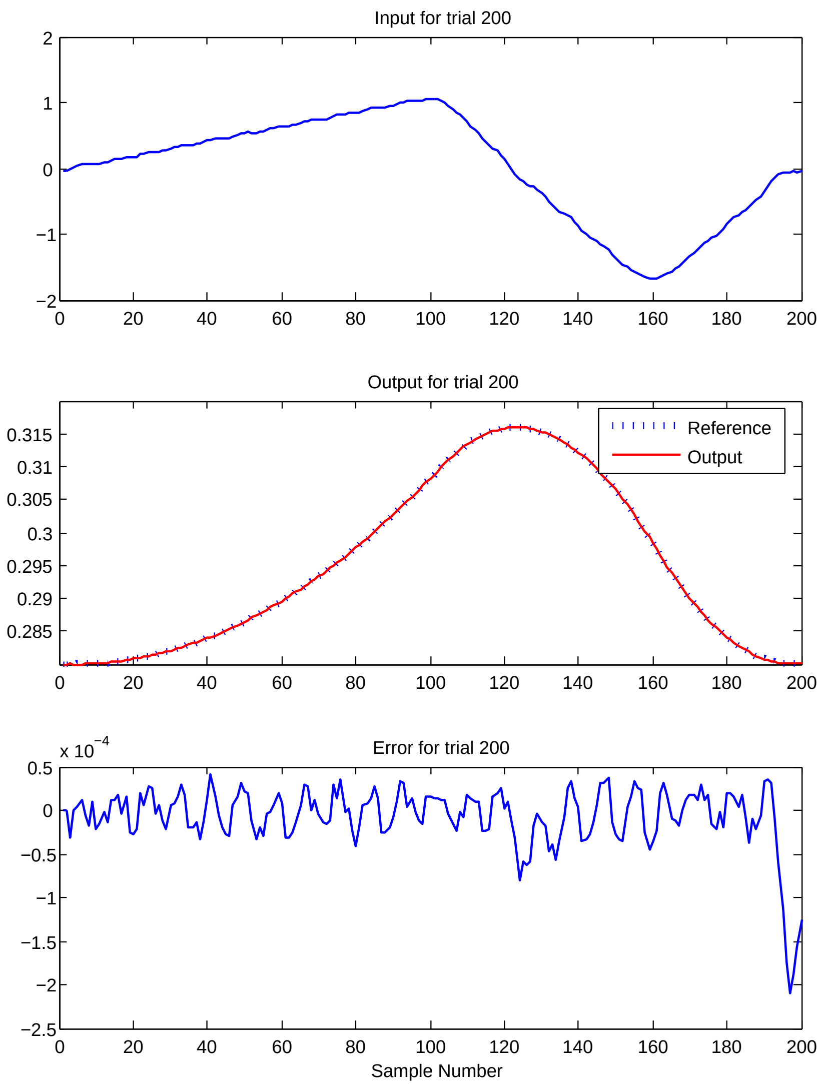


Figure 5.16: Along the trial performance of the 200th trial in one experiment.

5.5 Comparison of Results

All the experimental results have been compared in order to analysis the respective algorithm performance. Figure 5.19 compares the results of the output-feedback control scheme with different parameters. Figure 5.20 shows the results of the state-feedback control scheme and the output-feedback control scheme. The output-control scheme clearly performs much better in terms of final error. It is able to not only achieve smaller error values, but also maintain the error at a low level. They have a similar performance in terms of learning speed as the controller matrices (K_2 in state feedback controller and K_3 in output feedback controller) are very close to each other. These matrices are used to update the control input vector, so similar learning progression in the initial stage have been found.

Figure 5.21 shows a comparison of 2D repetitive process based ILC algorithms with

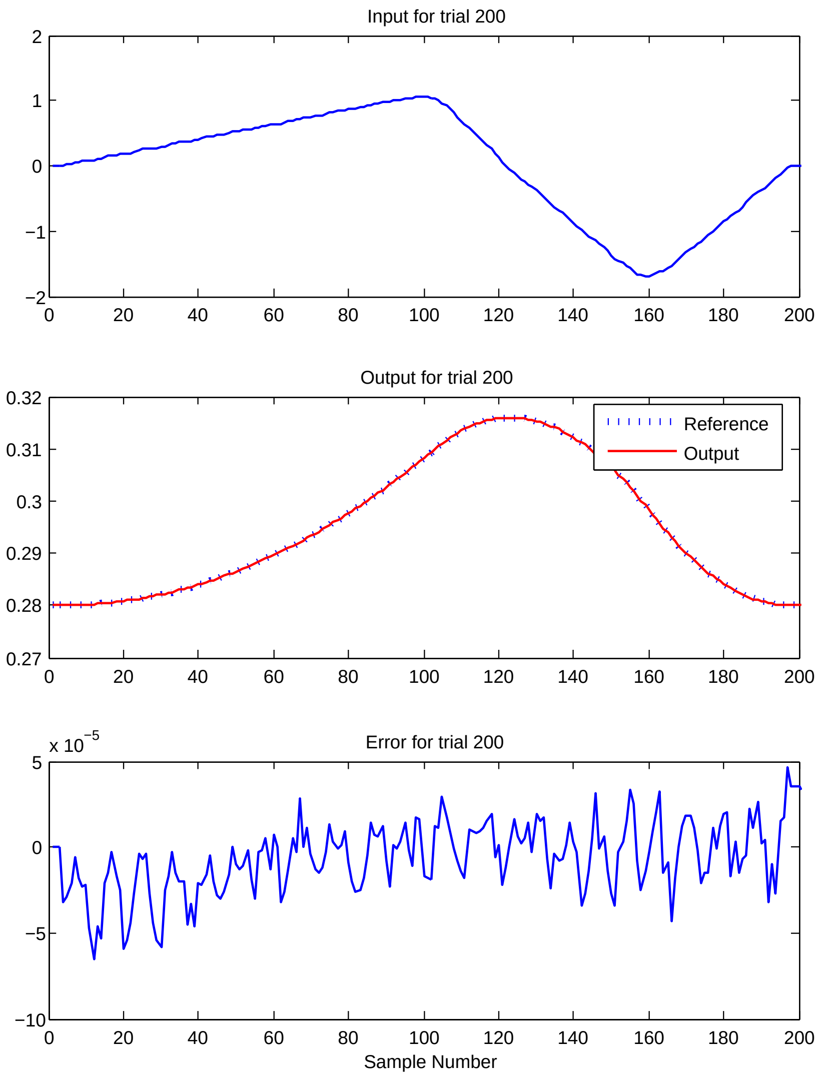


FIGURE 5.17: Along the trial performance of the 200th trial with extended reference.

some other algorithms (e.g. the basic P-type ILC, the inverse model ILC, the norm-optimal ILC and the stochastic ILC algorithms described in Chapter 4) in terms of mean squared errors. It is clear that the state-feedback controller actually provided the best performance in terms of a very small final error value and holding the error within a small band. The state-feedback controller does not perform well as only a very simple state observer is used. Therefore, state observation could be not performed with sufficient accuracy. More comparison results can be found in Chapter 6.

5.6 Summary

This chapter has considered the design of ILC schemes in a 2D linear systems setting and, in particular, the theory of discrete linear repetitive processes. This releases a stability theory for application which demands uniformly bounded along the trial dy-

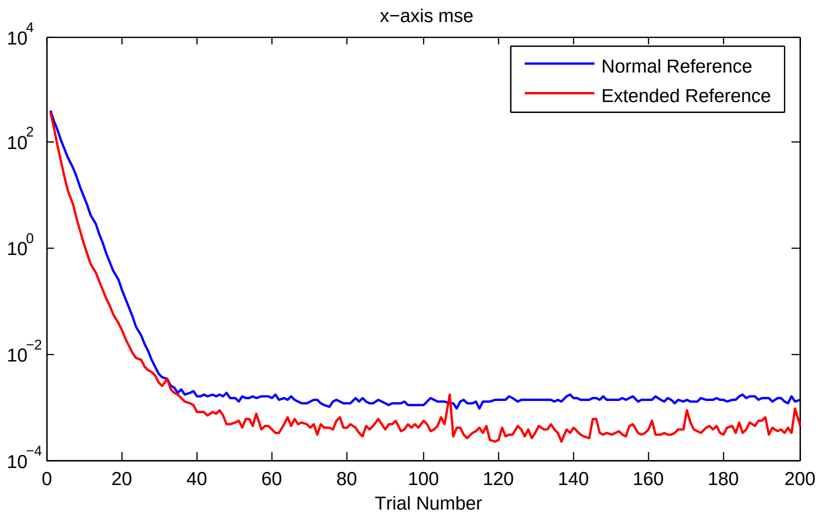


FIGURE 5.18: Comparison of using normal reference and extended reference

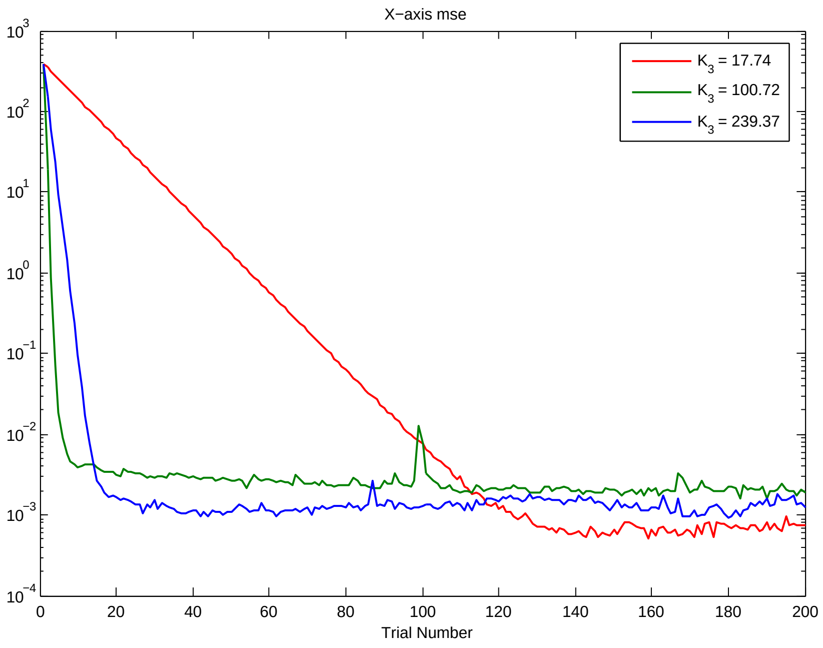


FIGURE 5.19: Influence of K_3 on the error $e_k(p)$

namics (whereas previous approaches only demand bounded dynamics over the finite trial length). This approach leads to a stability condition expressed in terms of an LMI with immediate formulas for computing the control law matrices. This is a potentially powerful approach in this general area which also makes a significant step forward in the application of 2D linear repetitive process theory. Two controller designs (the state-feedback design and the output-feedback design) have been developed and both designs have been experimentally validated on the gantry robot system whose basic task is to continually execute a pick and place operation.

The results here establish the basic feasibility of this approach in terms of both theory

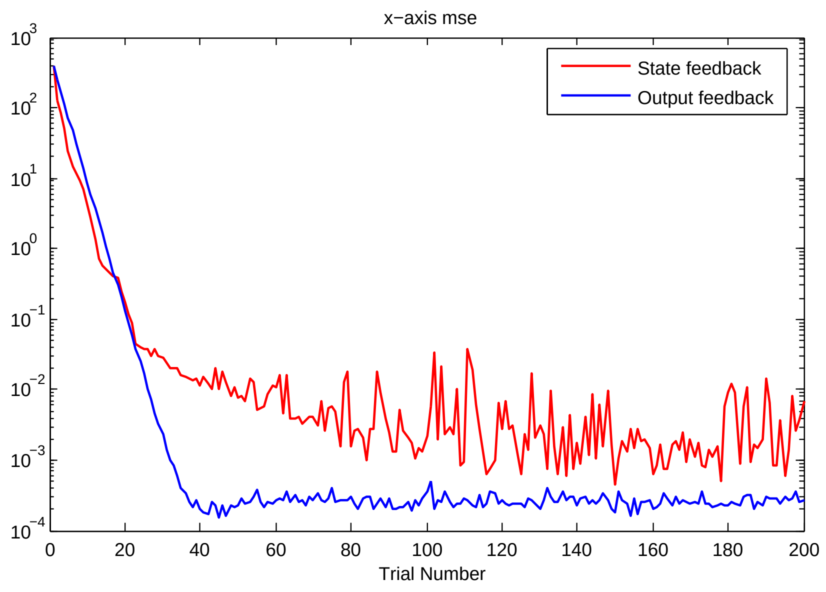


FIGURE 5.20: Comparison of state-feedback controller and output-feedback controller

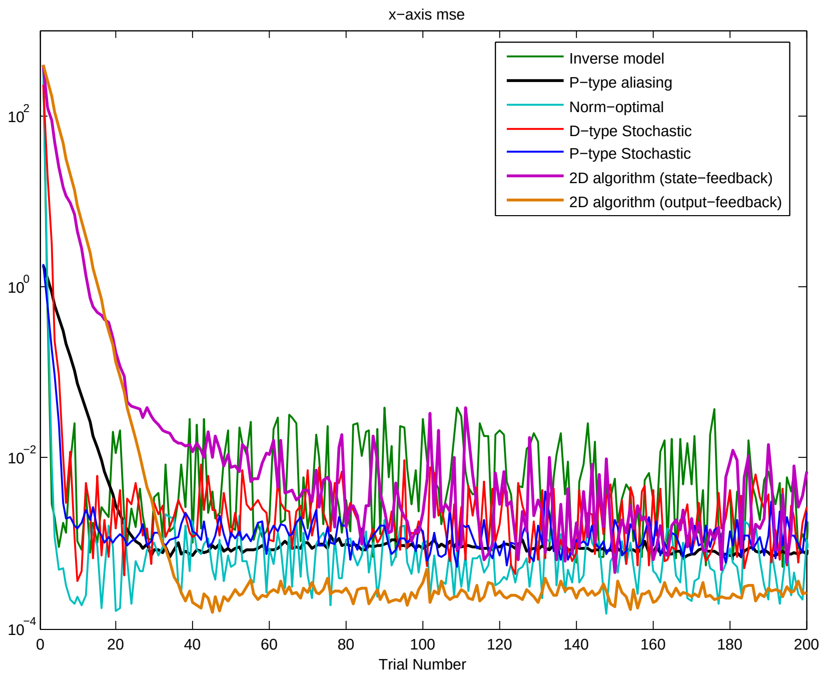


FIGURE 5.21: Comparison of 2D algorithms with other algorithms (X-axis).

and experimentation. There is a significant degree of flexibility in the resulting design algorithm and current work is undertaking a detailed investigation of how this can be fully exploited. One aspect which clearly requires investigation is to attempt design without the need to use current trial state feedback (particularly as the control law actually uses the difference of the state vector on two successive trials). Another is to extend this analysis to other classes of ILC and seek ways to reduce the possible conservativeness arising from the use of sufficient, but not necessary, stability conditions. Moreover, the gantry robot system has been used in experimentally testing a wide range of other non LMI based ILC designs and hence it will also be possible in due course to compare relative performance (an essential item in terms of end users).

Chapter 6

Comparative Analysis of ILC algorithms

Experimental benchmarking of ILC algorithms based on linear plant models is a major part of this thesis. In this chapter, the results obtained are considered under a number of key aspects that would form the basis for decision making as to which one to chose in a given situation. The comparison is mainly in terms of mathematical formulation, computational implementation and experimental results. Following this, the overall performances are shown.

6.1 Comparison of Mathematical Structures

Each ILC algorithm has its unique control input update equation. Table 6.1 summarises a number of algorithms in terms of mathematical formulation. The difference in complexity can be gauged from the number of equations in each algorithm. However, the number of equations is not the only indication of how complex the algorithm is as complex computation can occur in just a single equation or a very simple operator (for example, multiplication of large matrices, inversion of large matrix). Also, from the table, some of the algorithms compute the input vector offline which means all data can be calculated by using the measured data from previous trials and the computation can be achieved in the gap between two trials. Some algorithms need to compute the next input sample using the data of the last sample and data from another source. This is called online computation and needs to be completed within the sampling time which is typically very small. for example, the reference shift algorithm only needs to compute the next input using previous error information, therefore full computation can be carried out between trials. But the algorithms based on 2D system theory are likely to use the state or the output data from the very last sample. Offline computation has no time limitation, so even if the algorithm is very complex, there is no influence on

Algorithm	Control Law Equations
Basic P-type	$u_{k+1}(t) = u_k(t) + Le_k(t)$
Basic D-type	$u_{k+1}(t) = u_k(t) + L\dot{e}_k(t)$
Reference Shift	$u_{k+1}(t) = r(t-\tau'_{k+1}) + f_{k+1}(t)$ $f_{k+1}(t) = f_k(t-\tau''_{k+1}) + Le_k(t-\tau''_{k+1}+\tau_k)$ $\tau'_{k+1} = \max(\tau_k, \tau'_k)$ $\tau''_{k+1} = \max(0, \tau_k - \tau'_k)$ $\tau_k = \arg\min_\tau\{\parallel r'_k(t-\tau) + f_k(t-\tau) - y_k(t) \parallel_2^2\}$
Stochastic ILC	$u_{k+1}(t) = u_k(t) + K_k e_k(t+1)$ $K_k = \mathbf{P}_{u,k}(CB)^T \cdot [(CB)\mathbf{P}_{u,k}(CB)^T + (C-CA)$ $\mathbf{P}_{x,k}(C-CA)^T + C\mathbf{Q}_t C^T + \mathbf{R}_t]^{-1}$ $\mathbf{P}_{u,k+1} = (I - K_k CB)\mathbf{P}_{u,k}$ $\mathbf{P}_{x,k+1} = A\mathbf{P}_{x,k}A^T + B\mathbf{P}_{u,k}B^T + \mathbf{Q}_t$
Stochastic sub-optimal	$u_{k+1}(t) = u_k(t) + K_k e_k(t+1)$ $K_k = \mathbf{P}_{u,k}(CB)^T \cdot [(CB)\mathbf{P}_{u,k}(CB)^T + C\mathbf{Q}_t C^T + R_t]^{-1}$ $\mathbf{P}_{u,k+1} = (I - K_k CB)\mathbf{P}_{u,k}$
2D State-feedback ILC algorithm	$u_{k+1}(t) = u_k(t) + K_1\eta_{k+1}(t+1) + K_2 e_k(t+1)$ $\eta_{k+1}(t+1) = x_{k+1}(t) - x_k(t)$
2D Output-feedback ILC algorithm	$u_{k+1}(t) = u_k(t) + K_1\mu_{k+1}(t+1) + K_2\mu_{k+1}(t) + K_3 e_k(t+1)$ $\mu_k(t) = y_k(t-1) - y_{k-1}(t-1)$

TABLE 6.1: Mathematical formulation of algorithms

the control process. However, if the computation is too complex, it might have great impact to the whole control process when the sampling frequency is high because too much computation might not be completed in the very short inter-sampling time which will lead to loss of data or incorrect computation during the subsequent control.

6.2 Comparison of Implementation Code

As mentioned in the last section, the number of equations does not truly reflect the complexity of the control algorithm. The equations would be eventually converted into computer language in the process of implementation. Taking the stochastic learning algorithm as an example:

$$\begin{aligned}
u_{k+1}(t) &= u_k(t) + K_k e_k(t+1) \\
K_k &= \mathbf{P}_{u,k}(CB)^T \cdot [(CB)\mathbf{P}_{u,k}(CB)^T + (C-CA) \\
&\quad \mathbf{P}_{x,k}(C-CA)^T + C\mathbf{Q}_t C^T + \mathbf{R}_t]^{-1} \\
\mathbf{P}_{u,k+1} &= (I - K_k CB)\mathbf{P}_{u,k} \\
\mathbf{P}_{x,k+1} &= A\mathbf{P}_{x,k}A^T + B\mathbf{P}_{u,k}B^T + \mathbf{Q}_t
\end{aligned}$$

Here, A, B, C are matrices from the plant model and $\mathbf{P}_{u,k}$,$\mathbf{P}_{x,k}$,$\mathbf{Q}_t$ are parameter matrices which are of the same order as the plant model. The algorithm computes K_k which is the learning gain in the control law, and it needs at least 14 matrix multiplications, one matrix inversion and several instances of matrix addition or subtraction. Table 6.2 compares the stochastic learning algorithm and the output-feedback control based on 2D theory.

Stochastic learning algorithm

```
P_xk[axis_number] = (A[axis_number] * P_xk[axis_number] * A[axis_number].Transpose)
    + (B[axis_number] * P_uk[axis_number] * B[axis_number].Transpose) + Q_t[axis_number];
T1 = C[axis_number] * B[axis_number];
T2 = C[axis_number] * A[axis_number];
S = C[axis_number]- T2 * P_xk[axis_number] * (C[axis_number]- T2).Transpose
    + C[axis_number] * Q_t[axis_number] * C[axis_number].Transpose + R_t[axis_number];
T = (T1 * P_uk[axis_number] * T1.Transpose + S).Inverse;
K_k[axis_number] = P_uk[axis_number] * T1.Transpose * T;
P_uk[axis_number] = (1 - K_k[axis_number] * T1) * P_uk[axis_number];
```

Output-feedback control based on 2D theory

```
axis_displacement[axis_number] = obtain_m323a_axis_count(axis_number);
sample_trajectory[axis_number] = profile[axis_number][sample_count];
y_k_p = axis_displacement[axis_number] - place_home[axis_number];
if(iteration_number == ITERATION_START_POINT)
    y_km1_p = 0;
else
    y_km1_p = data_save.data_array[raw_encoder][axis_number][sample_count]
        - place_home[axis_number];
if(sample_count==SAMPLE_START_POINT)
{
    y_k_pm1 = 0;
    y_km1_pm1[axis_number][sample_count] = 0;
}
else
    y_k_pm1 = data_save.data_array[raw_encoder][axis_number][sample_count-1]
        - place_home[axis_number];
output_update[axis_number] = K1[axis_number] * (y_k_p - y_km1_p)
    + K2[axis_number] * (y_k_pm1 - y_km1_pm1[axis_number][sample_count]);
K1part[sample_count] = K1[axis_number] * (y_k_p - y_km1_p);
K2part[sample_count] = K2[axis_number] * (y_k_pm1 - y_km1_pm1[axis_number]);
y_km1_pm1[axis_number][sample_count] = y_k_pm1;
output_voltage[axis_number] = output_update[axis_number] + learnt_profile[axis_number][sample_count];
learnt_profile[axis_number][sample_count] = output_voltage[axis_number];
```

Table 6.2: Algorithm computer code comparison

Purely from the amount of computer code, the complexity or efficiency of the algorithm still can not be determined as some of the code still include sub procedures or sub functions which return values in batches. Later in this chapter the time complexity will be provided by recording the processing time of each algorithm and also the space complexity will be evaluated by recording the physical memory used. This will allow better and more accurate analysis of the algorithm efficiency.

6.3 Comparison of Experimental Results

Comparing the results is the most common method of comparing algorithms. As argued in Chapter 2, the mean squared error plots are the most direct way to differentiate between algorithms as it clearly displays the learning speed, final error value, trend of divergence, stability and error oscillation. To illustrate that, Figure 6.1 (the results are taken from Ratcliffe (2005)) shows a comparison of four ILC algorithms performed on

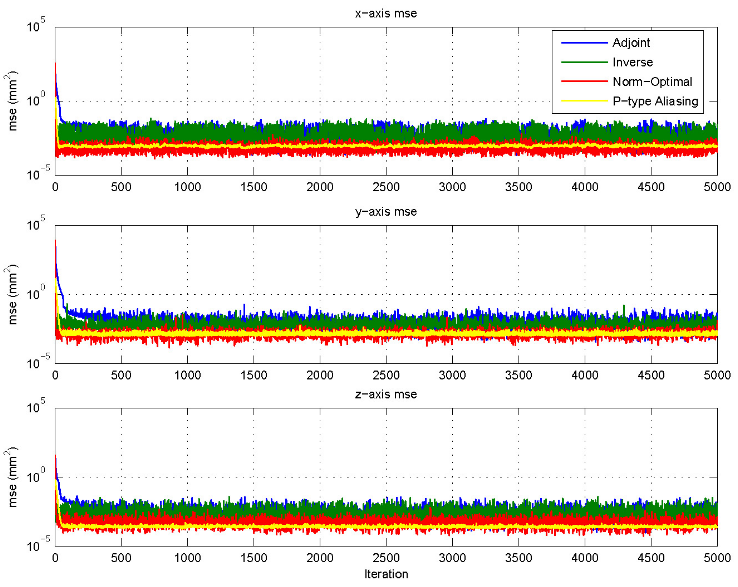


FIGURE 6.1: Comparison of four algorithms implemented on a multi-axis gantry robot.

the gantry robot over 5000 trials. Since the the experiment was carried out for a large number of trials, they clearly indicate that all four algorithms have good performance in terms of long-term performance. For the results of the X-axis, the norm-optimal algorithm outperforms the others as it produces the smallest error on average. The P-type algorithm with an aliasing filter achieved very good performance in holding error between small values without excessive oscillation. In terms of convergence speed, this figure is not as clear as the trial number axis (horizontal) shows a large number of data points, therefore a magnified view is needed to show the difference in learning speed. Figure 6.2 shows the first 200 trials of the results in Figure 6.1, and it clearly shows that norm-optimal algorithm and inverse model algorithm rapidly reduce the error down to a small value within 10 trials. The adjoint algorithm appears to be the slowest.

In the previous chapters, there are many comparisons of experimental results, especially showing the influence of parameter tuning. The following section provides a comparison between the algorithms which have been developed and implemented in this thesis.

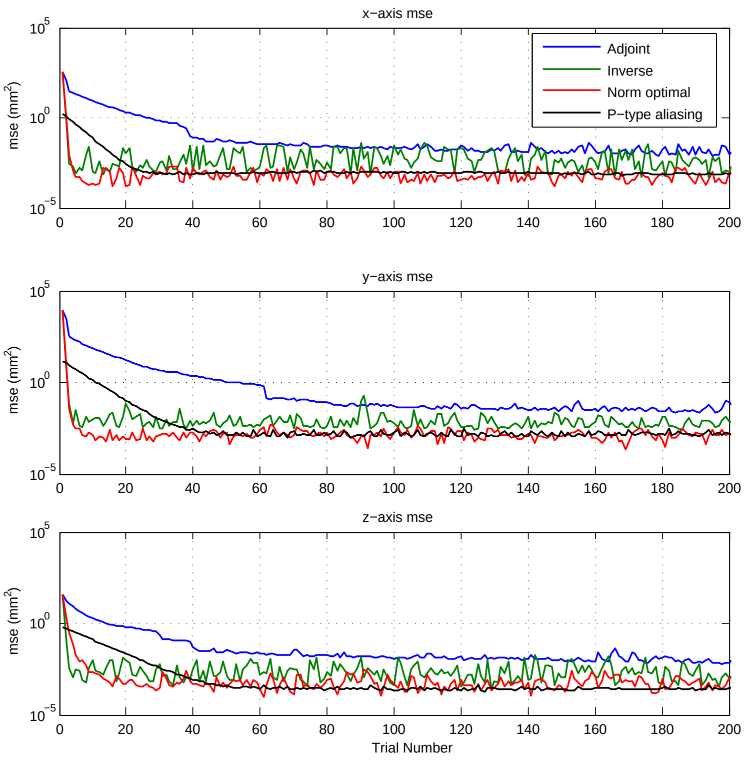


FIGURE 6.2: Comparison of four algorithms implemented on a multi-axis gantry robot (magnified).

6.3.1 MSE Comparison

Figure 6.3 shows a comparison of mean squared errors for the five ILC algorithms which have been implemented on the three axes of the gantry robot. The results of each algorithm are the best from all experimental results collected. The convergence speed and final error level for each algorithm are clearly shown. It is clear that the output-feedback controller of the 2D algorithm outperforms all other algorithms in terms of final error level. The D-type/P-type stochastic learning algorithm converge much faster than the others. Due to the help of the PID feedback controller, the P-type stochastic learning algorithm has a good initial tracking performance. After around 100 trials, most of the algorithms, except for the output-feedback 2D algorithm perform well and produce results of a similar level. The P-type stochastic learning algorithm and the output-feedback design of 2D algorithm perform very well in terms of preventing large oscillations in the error.

6.3.2 Tracking Comparison

Mean squared error plots are able to highlight differences in convergence and final error values. However, at some stage, individual trial performance must be examined. Therefore, comparison between individual tracking data are also essential. Some of the individual trial performance results have been shown in previous chapters in the course

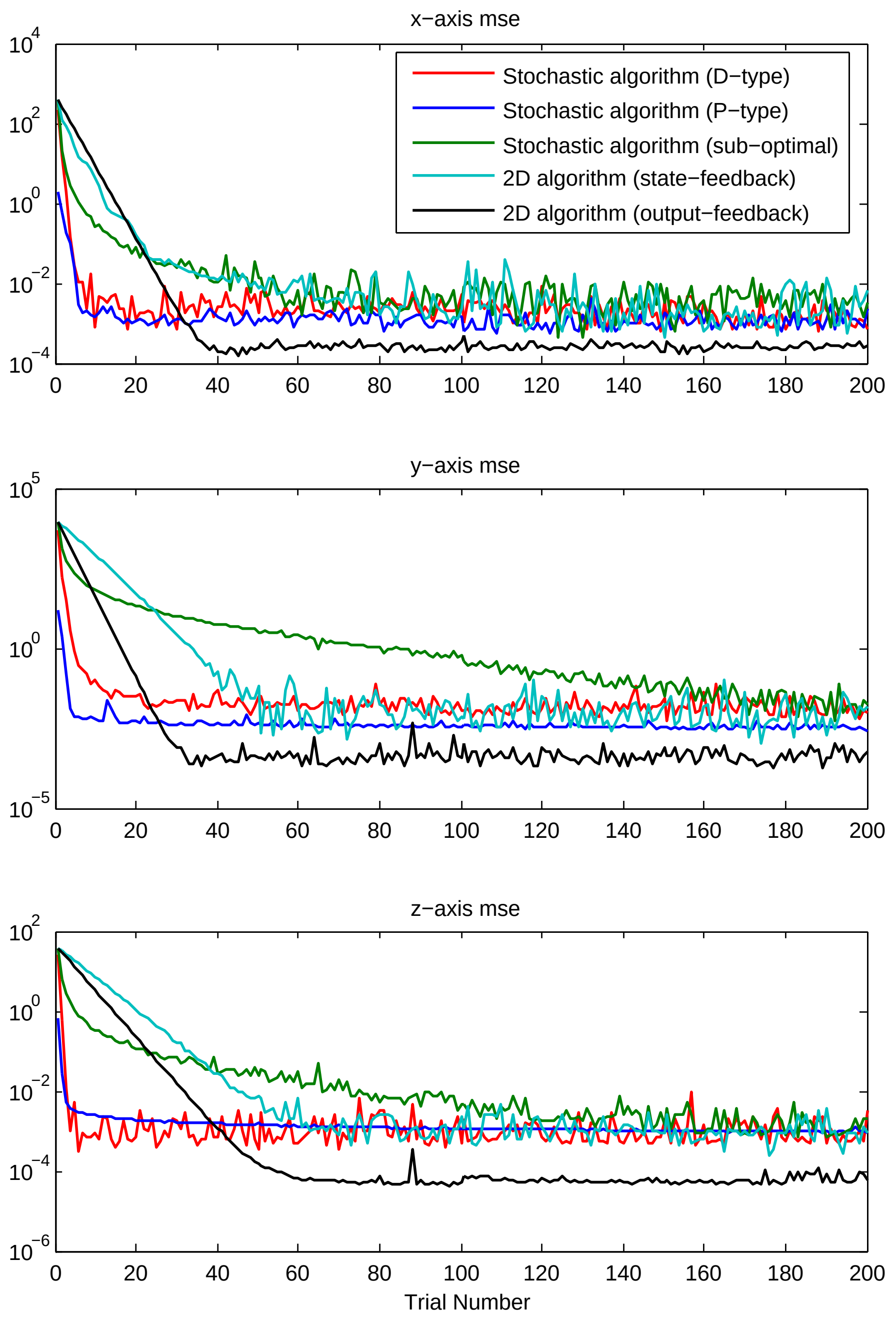
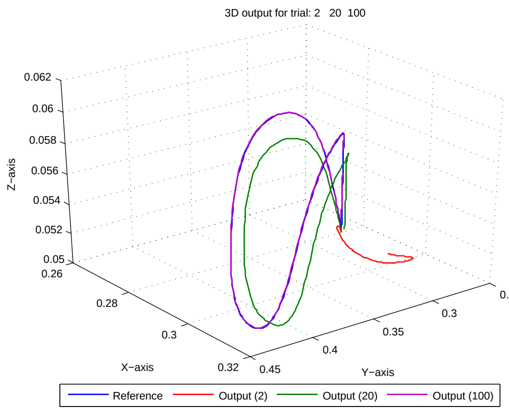
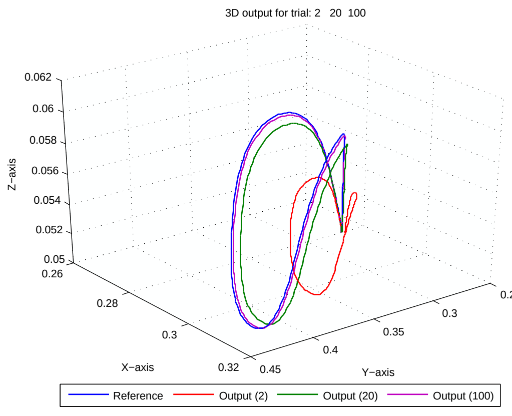


FIGURE 6.3: Comparison of the algorithms implemented on a multi-axes gantry robot.

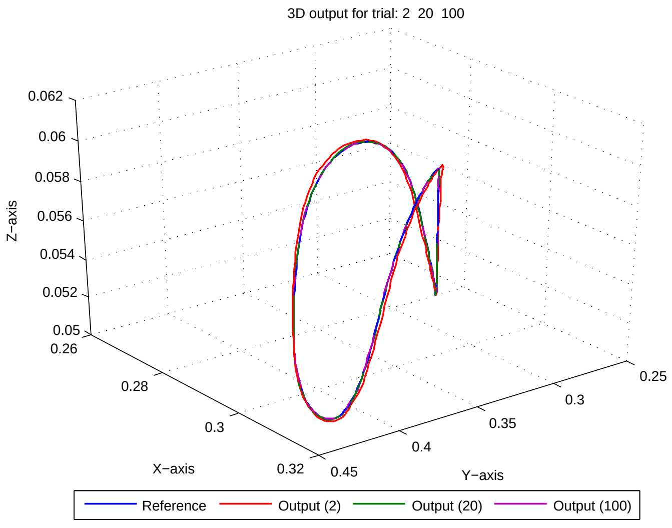
of selecting filters. Here, individual trial performance will be presented in a 3-dimension plot showing the output trajectory along with the reference trajectory. Figure 6.4(a), Figure 6.4(b) and Figure 6.4(c) show the output tracking performance for the 2D algorithm (state-feedback design), the stochastic algorithm (sub-optimal) and the stochastic algorithm (P-type) respectively. It is obvious that the stochastic algorithm with P-type ILC performs better in initial trials. The main reason for this is that there is a relatively high-gain PID feedback controller used in conjunction with the ILC controller in the algorithm setup. The algorithm based on 2D theory performs badly in the first few trials but the latter trial performance is better than the others.



(a) 2D algorithm (state-feedback)



(b) Stochastic algorithm (sub-optimal)



(c) Stochastic algorithm (P-type)

FIGURE 6.4: Output trajectories using different ILC algorithms

A separate comparison is carried out to show the difference between these three control designs more clearly. Figure 6.5 shows the output for these three algorithms over the second trial. As mentioned before, the stochastic algorithm (P-type) is used with a PID feedback controller which improves the initial trial performance, while the 2D algorithm performs badly as it has not yet got enough information to learn effectively. Figure 6.6 and Figure 6.7 provide a comparison over the 20th trial and the 100th trial output trajectories for these three algorithms. All three algorithms achieve very good tracking performance at the 100th trial. It is not easy to evaluate the algorithm using this kind of plot at this stage. Mean squared error plots are able to facilitate this. However, the 3D output plot is still useful, especially for examining the initial stages of learning as sometimes a good understanding of the movement is needed (e.g. robot manipulators need to move around without colliding with obstacles, and this can be shown using a 3D output plot).

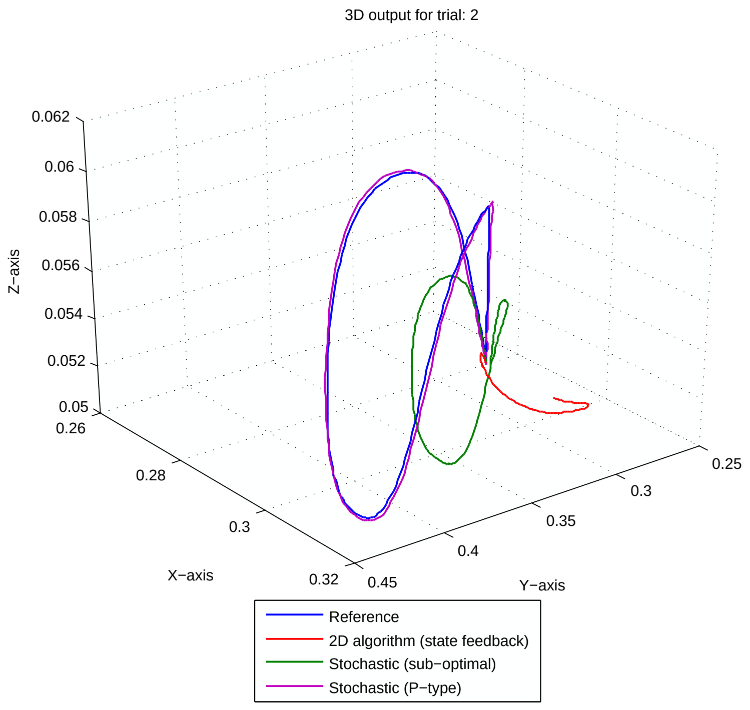


FIGURE 6.5: Comparison of output trajectories for three ILC algorithm (Trial:2)

6.4 Algorithm Efficiency

Complexity theory deals with the relative computational difficulty of computing functions and solving other problems numerically. The time complexity of a problem is the number of steps that it takes to solve an instance of the problem as a function of the size of the input (usually measured in bits), using the most efficient algorithm. The space complexity of a problem is a related concept, that measures the amount of space, or memory required by the algorithm. Here the time and space complexity are not generated by using mathematical analysis but directly recorded or read from the computer.

6.4.1 Timing Complexity

Here time complexity considers only the processing time of computation within each sample and the time of computation between every two which are all recorded and averaged to express the time complexity of each algorithm. Table 6.3 shows the recorded processing time of five ILC algorithms implemented on the gantry robot.

6.4.2 Computing Complexity

The space complexity here is evaluated by examining the measured memory required by the algorithm at the running status. The following algorithms to be compared are

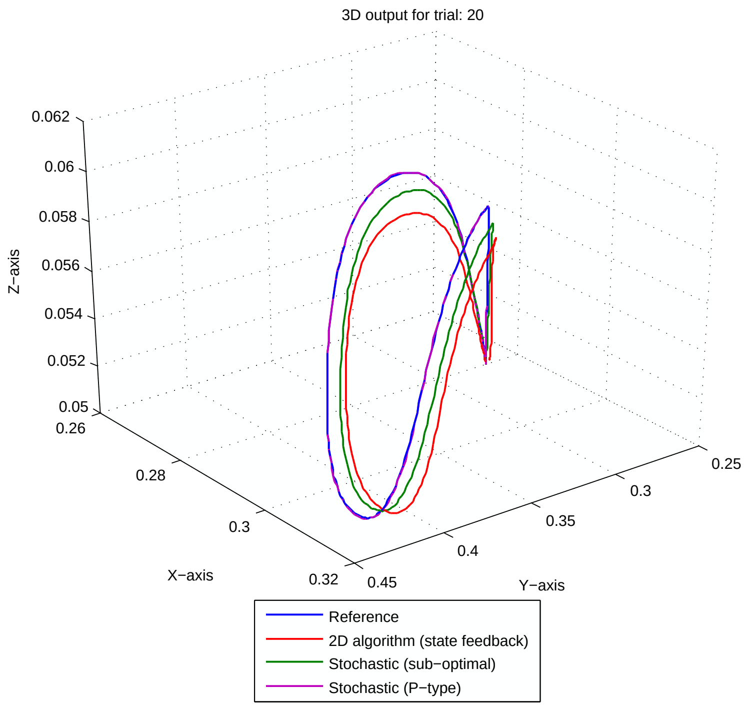


Figure 6.6: Comparison of output trajectories for three ILC algorithms ($k = 20$)

Algorithm	Time in sample (ms)	Time between trials (ms)
Stochastic algorithm (D-type)	0.095	1.653
Stochastic algorithm (P-type)	0.087	1.586
Stochastic algorithm (sub-optimal)	0.051	0.952
2D algorithm (state-feedback)	0.122	0.424
2D algorithm (output-feedback)	0.208	0.395

Table 6.3: Recorded processing time of ILC algorithms

written in the same computer language and are run on the same system platform. The reading of required memory is taken when the robot is moving. More than ten readings are taken and then averaged to give the following results (see Table 6.4 for details).

Algorithm	Memory required (KB)
Stochastic algorithm (D-type)	19,563
Stochastic algorithm (P-type)	19,286
Stochastic algorithm (sub-optimal)	16,952
2D algorithm (state-feedback)	17,424
2D algorithm (output-feedback)	15,395

Table 6.4: Recorded required memory of ILC algorithms

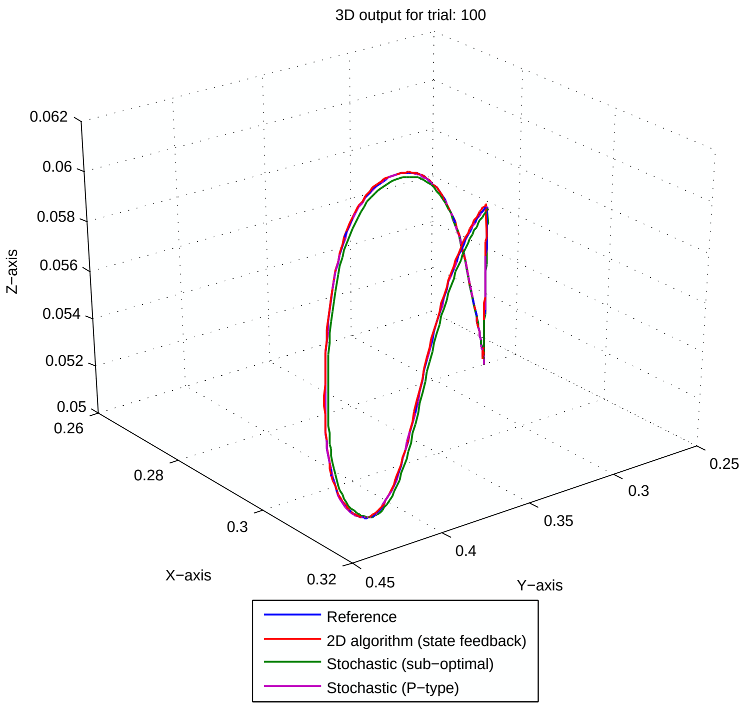


FIGURE 6.7: Comparison of output trajectories for three ILC algorithms ($k = 100$)

6.5 Algorithm Accuracy

The accuracy of an algorithm can be evaluated using the tracking error recorded over each trial. The MSE plots show the performance of each algorithm and provide a clear overview of each approach. However, the comparisons can represent only the performance variation. Is it possible for the algorithm being used in industry or has the experimental results met the standard in industry? Another question is motivated by this example: a certain project needs a robot of providing a tracking error (mean squared error) less than $10^{-4}m^2$. But the experimental data for an algorithm on the gantry robot facility produced a mean squared error of $10^{-3}m^2$. Does it mean the algorithm has not got sufficient ability for the given application? The answer is negative simply because the measured error signals are based on devices used on the facility, and the devices (e.g. an encoder) will have a specific accuracy in measuring. Sometimes, the algorithm works well but the encoder may have limited accuracy. It is not likely to produce better results. However, if one algorithm is able to produced more accurate results while another one is not, then even if using a better measuring device, it is not possible to give superior results.

6.6 Problems in Practical Implementation

There have been several problems encountered when implementing algorithms on the experimental facilities. Some of them are predictable while others are not. An analysis of the experimental results is likely to provide good evidence and support modification to the algorithms. In general the major problems encountered in this research project are problems related to measurement and computation.

6.6.1 Measurement Problems

As mentioned in previous sections, there will exist a problem of measurement in the practical implementation as the devices using in the experiments have got a specific accuracy. It is not possible to provide results beyond this level of accuracy. This sometimes will disturb existing learning that has been achieved already in a learning control process. Take the gantry robot as an example: the encoders have got an minimum step which is 0.000032m. When the learning process has been running for a large number of trials, sometimes the value of the reference signals at some sample points is not be exactly divisible by this minimum step. So that no matter how good a learning controller is designed, there will exist certain tracking errors (see Figure 6.8 for details). And in some cases, the errors will be magnified to produce the control input which potentially disturbs the control performance or even makes the system unstable.

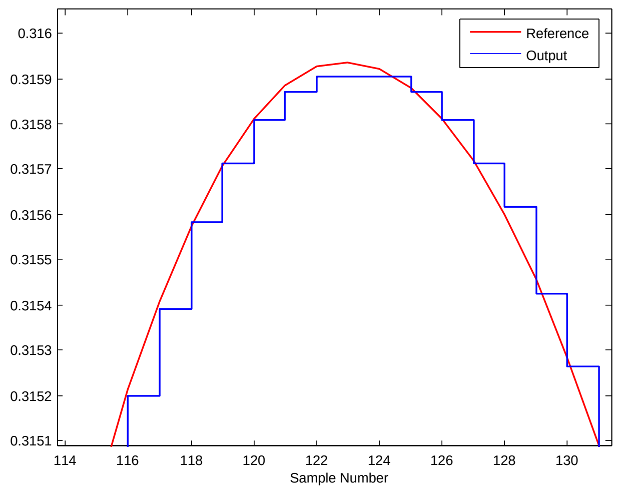


FIGURE 6.8: Output signal and ideal reference signal.

These differences between the output and the ideal reference signal very often result in producing the appearance of high frequency noise in the later part of a learning control process as the measured data samples are usually on both sides of the ideal reference. Then the error signal will look like the shape of a sawtooth. When the error signals are magnified by the learning gains, it produces a high frequency noise signal. This might be tiny and invisible in the process, and sometimes it takes very long time to appear (e.g. after thousands of trials). As discussed in the previous chapters, high frequency noise was observed in the X-axis of the gantry robot after around 100 trials of learning. That is mainly because of the resonance in the X-axis at certain frequencies. It is basically due to the fact of system dynamic. Therefore, using a zero-phase low pass filter is one way of removing the unexpected frequency parts and maintaining the performance of the whole system. Usually the frequency of the resonance will be lower than the frequency of noise produced by measurement error. So a low pass filter is able to filter out the high frequency noise produced by measurement error as well.

6.6.2 Computation Problems

With the development of computer technology, a single processor unit can complete millions of calculations in one second. Also, a major reason ILC has attracted so many researchers is the possibility of simple algorithm which provide a high level of performance. However, in order to improve ILC performance, many complex algorithms have been developed. Most of them require the plant model and need a series of computations to be executed between samples or trials. Usually, a lot of computation takes the form of matrix multiplications. The more details wanted in the controller, the larger the matrices will be. Therefore, computation problems may become more and more common. For example, when a system is running at a 1000 Hz sampling frequency, all computation of the next control input for the next sample must be completed within 1ms which for some algorithms on certain platforms, is not feasible. Furthermore, in most applications in industry microprocessor chips are used which might be less powerful than a normal computer. One reason for this is that manufacturers make a balance between the cost of hardware and the performance.

Besides the problems of computation speed, there are also some issues in computation accuracy. The programming platform used in the control process sometimes needs a very high computational accuracy to allow better performance. Some of the compilers just use approximation of the result. However, after a series of multiplication, the difference in the approximation sometimes would be scaled significantly which results in large error. Take the following case as an example, as described in Chapter 4; the learning gain matrix for the stochastic learning algorithm can be computed by the following equations:

$$
\begin{aligned}
K_k &= \mathbf{P}_{u,k}(CB)^T \cdot [(CB)\mathbf{P}_{u,k}(CB)^T + (C - CA) \\
&\quad \mathbf{P}_{x,k}(C - CA)^T + C\mathbf{Q}_t C^T + \mathbf{R}_t]^{-1} \\
\mathbf{P}_{u,k+1} &= (I - K_k CB)\mathbf{P}_{u,k} \\
\mathbf{P}_{x,k+1} &= A\mathbf{P}_{x,k}A^T + B\mathbf{P}_{u,k}B^T + \mathbf{Q}_t
\end{aligned}
$$

The stochastic learning algorithm repeatedly deals with the model matrices A, B, C and it takes a long time to compute the adaptive gain K_k between every trial. Practically, it is not feasible to implement the algorithm for the sampling time $T_s = 0.001$s (sampling frequency: 1000Hz) because computation for K_k contains too many matrix operations which are not accurate and reliable enough using C programming. Figure 6.9 shows that a comparison of K_k evolution progression using C++ and Matlab (Matlab gives much more computational accuracy). It is clear that the values for 100Hz are almost the same while for 1KHz, the results using C++ programming are not reliable after about 5 trials.

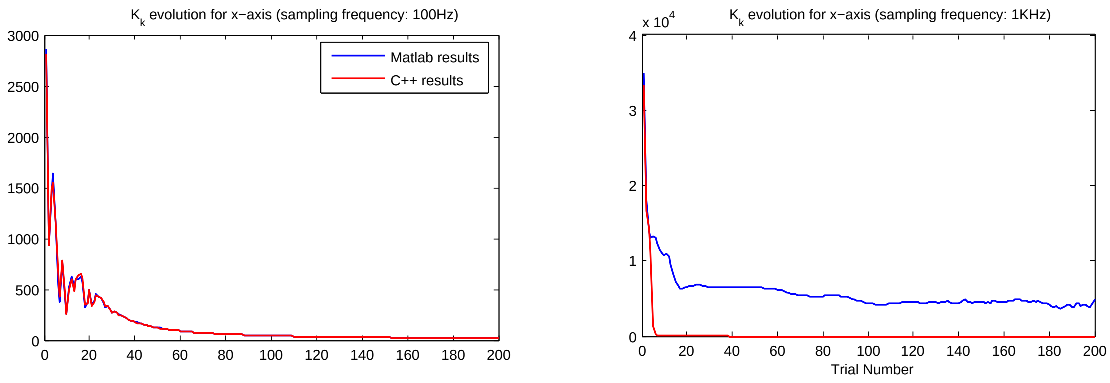


FIGURE 6.9: Adaptive gain K_k evolution of sampling frequency 100Hz and 1KHz using different computational tools.

However, the problem can be solved by computing the tuning parameters in batch before the iterative loop starts since all the parameters appearing in the algorithms are not related to what is measured during each trial. This will greatly reduce the computation time between successive trials. Also, it is possible to compute all the parameters before implementation using a more accurate mathematical software environment, for example Matlab. However this can not be applied to processes without operational gaps. Another solution to this issue is to introduce a more accurate computation package in the development platform. But this will then face the previous problem of computation speed. Actually, all these issues can be tuned as tuning parameters in an optimisation problem. A balanced setting up is more likely to give the best results in practice.

6.6.3 Problems Related to Discrete Systems

With the development of digital components, most controllers are now designed using the assumption of discrete underlying system. In both the controller design stage and the implementation, the sampling rate is an important parameter. Sometimes, the physical plant is only able to operate over a certain range of sampling rate (e.g. 100Hz to 1KHz). A sampling rate too low or too high will make some of the components in a system work improperly and produce erroneous results. However, for some designs of controllers, the sampling rate will be prohibitive constraint. Sometimes, it is not possible to find a solution in a controller design problem using some particular sampling rates. This will make the controller design not possible to be implemented (see for example the case of the output feedback controller design for the 2D ILC algorithm in Chapter 5).

Possible solutions have been provided in the last chapter: one is to use a multi-sampling technique which uses more than one controller in the control loop. The other method is by using linear interpolation to fill the gaps in sampled data. A comparison has been given already and the multi-sampling technique produced better results as it does use more information than the linear interpolation method.

Similarly, using a higher sampling frequency is likely to produce superior results. Figure 6.10 compares two different sampling rates $T_s = 0.01$s (100Hz) and $T_s = 0.005$s (200Hz). It is clearly that using 200Hz is much better in the latter stages as previously mentioned it does have more available information than the controller of 100Hz. Further increasing the sampling rate is not helpful as it might introduce higher frequency noise in the process and will make performance even worse as a controller of lower sampling rate itself can act as a low-pass filter.

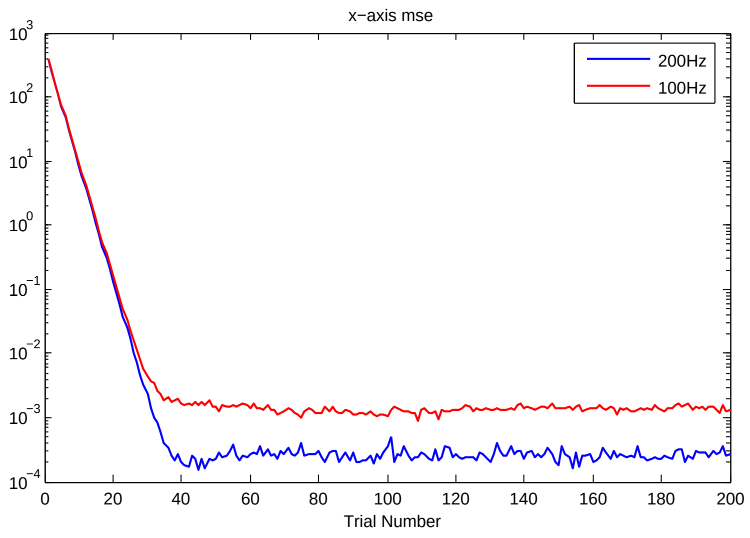


FIGURE 6.10: Comparison of using different sampling rate

6.7 Summary

In this chapter, some issues and problems related to iterative learning control in practical implementations have been discussed including algorithm differences, comparison of results and some practical problems such as measurements and computation. To some particular problems, solutions have been provided but still there are many problems which are waiting for good or improved solutions. Therefore, there is still much further work to conduct in practical ILC research and the next chapter will provide some conclusions resulting from the work in this thesis and some further plans are given in this area.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

The main thrust of this thesis has been experimental benchmarking of ILC schemes with algorithm development and enhancement where appropriate. The experimental work has been completed on two experimental facilities in the form of a non-minimum phase electro-mechanical system and a gantry robot respectively. The former is based on rotary action and uniquely enables the effects of non-minimum phase dynamics on ILC control laws. The task for the gantry robot is to place objects on a moving conveyor belt under synchronisation. Both of these facilities have already been used in experimental testing of some ILC algorithms and this forms the basis for the work reported in this thesis.

For the non-minimum phase case, a previously developed ILC algorithm (the phase lead algorithm) was reported to be a simple and efficient algorithm which achieved a high level of performance. However, two major problems were still left open; how to select the phase lead value and the low level of initial performance along the trial. A new reference shift algorithm that overcomes these problems has been developed. The algorithm computes the shift value automatically by minimising the difference between the output data and shifted reference signal after each trial to make the shift parameter adaptive. Moreover the algorithm does not need any information about the plant which makes implementation much simpler. Experimental results have shown that the performance of the reference shift algorithm produces superior performance compared with the phase-lead algorithm carried out using that plant. The reference shift algorithm achieved faster convergence speed and is able to maintain a constant final error. The reference shift algorithm produces errors which are bounded below a small value and it achieves similar convergence with less fluctuation in tracking error when compared to the norm-optimal algorithm which is highly complex in computation. Some problems and issues have been identified and they have been already scheduled into a further research plan.

On the multi-axis gantry robot system, two main categories of algorithms have been tested: stochastic learning algorithms and ILC algorithms based on repetitive processes control theory including five sub-algorithms which are: the D-type stochastic learning algorithm, the P-type stochastic learning algorithm, the sub-optimal stochastic learning algorithm, the 2D state feedback ILC algorithm and the 2D output feedback ILC algorithm. Because these two classes of algorithm are both model-based techniques, the plant models of all three axes are required prior to the simulation and experimental work. All three axes of the gantry robot were modelled by means of frequency response tests. A number of sinewave signals of over a hundred different frequencies were fed into each axis and the corresponding outputs were measured and stored to produce the Bode plots. A fitted model for each axis then was computed for algorithm design and simulation and experimental work.

Some applications of ILC will require operation in the presence of levels of noise which it is not possible to assume negligible for the purposes of control law design. This has led to the emergence of some design algorithms in the literature but no attempts to experimentally verify their performance. Therefore the implementation here comprises a verification plus a source of useful information for further development. A large number of experiments were carried out on the gantry robot using various parameters. Results have been compared to show the effects of tuning different parameters. For the sub-optimal algorithm, the optimisation of parameters was carried out and verified using experimental results. A zero-phase low pass filter has been used throughout all experiments and a new filter arrangement was used which is experimentally shown to produce superior results. Compared to the performance of other algorithms carried out using the robot, the stochastic algorithm appeared to be reasonable good. It is only eclipsed by the far more computationally intensive norm-optimal ILC algorithm.

2-dimensional (2D) system theory (in particular, the theory of discrete linear repetitive processes) has been considered in developing a class of ILC controllers as ILC is fundamentally a 2D system. The stability theory for linear repetitive processes has been used in the ILC controller design to guarantee the stability along the trial as well as the stability from trial-to-trial. This is a novel approach which not only achieved stability of both dimensions in ILC but also improved the along the trial performance. The controller was initially designed based on a state feedback controller. Experimental results show that the performance of the state-feedback algorithm are comparable to the previously implemented algorithms (e.g. P-type ILC, inverse model ILC, norm-optimal ILC) on the test facility. Furthermore, an output-feedback control scheme was developed to overcome the case when the states of the system are neither measurable nor observable. Output-feedback design does not require any component of the system state, it therefore also avoids the problem of inaccurate state estimates often encountered when using an observer. Experimental results show that the performance is superior to all the other results produced on the gantry robot. Some practical problems have been detected in the course of the experimental work such as the sampling rate problem and the numerical computation problem. A multi-sampling technique was found to be effective in solving the problem that exists when a lower sampling frequency can not be applied to the experimental facility, by introducing more than one controllers in the algorithm implementation. The effects of parameters tuning were also studied and the results were compared.

Finally, an analysis of the experimental results was then carried considering all relevant aspects. The analysis compared all the ILC algorithms which were implemented in this thesis taking into account different issues (e.g. the mathematical formulation, implementation program, time and space complexity and associated experimental results). Also some practical problems encountered in the experiments have been summarised which are useful for further investigation as part of future research work.

7.2 Future Work

The results reported in this thesis significantly advance the knowledge supporting the development of ILC towards eventual industrial use. It has also highlighted the need for further research in a number of key areas that are now discussed.

The first of these is the computational complexity of the algorithms involved and the trade-off between the more complex of these and improved performance relative to simpler structure algorithms. There will clearly be cases when simple structure algorithms, such as the P-type, will be all that is required and others where powerful state-space based algorithms must be used. Equally, there will be cases where the latter here do offer improvement but this may not be significant enough over simpler structure alternatives to merit the costs etc involved in switching over. There is productive research to be done on attempting to reduce the complexity of more advanced algorithms by, for example, optimising how the computations can be done. Some preliminary research in this direction was undertaken in Ratcliffe (2005) for norm-optimal ILC leading to an implementation known as fast norm-optimal with supporting experimental evidence. This, however, did not include issues such as robust control law design or any work on the 2D repetitive process and stochastic algorithms. One approach here could be to formulate the overall problem in an optimization setting.

In terms of current practical applications a number of problems have arisen in the work here that require further investigation. For example, in Chapter 3, a number of time shift values are able to guarantee the convergence criteria after fixed cut-off frequency filtering, so an adaptive filtering design is possible to further improve its performance. Further research will be carried out for the problem of different sampling rate in algorithm design and practical implementation. Also for current multi-sampling techniques, theoretical analysis and support will be needed. Additionally, the averaging algorithm is considered as a filtering process for application to signals appearing in ILC. This may be an effective solution for noise reduction.

Moreover, implementation of repetitive control is also another task for future practical research. Repetitive control techniques are remarkably similar to iterative learning control techniques, except that the algorithm must be computed in real-time as the test proceeds: there is no stoppage time between repetitions and the data cannot be processed in batches. The terminal conditions at the end of each repetition are the initial conditions for the next repetition; there is no resetting between trials. These features are therefore well suited to the conveyor situated beneath the gantry robot. The combination of the conveyor with the gantry will be another control task needed to extend this practical research, as synchronisation is another major problem to face. Also successful results will provide the industry with useful reference methods for use in designing and construction of systems.

In terms of current practical applications a number of problems have arisen in the work here that require further investigation. These are summarised next together with areas that require more algorithm development and then experimental testing.

- The stochastic ILC results developed in this thesis are based on a state-space model description. There is also much further development needed here. For example, a transfer-function based analysis has recently been published (Bristow, 2008) and work is currently under way on evaluating this approach. There is also a repetitive process based setting that can be used here.
- The results developed in this thesis have shown that the repetitive process setting can be extended from the original 2D Roesser state-space model based trial-to-trial convergence analysis to produce algorithms for trading-off trial-to-trial error convergence against along the trial performance. The starting point for a robust control treatment that would allow, unlike current approaches, for along the trial dynamics has also been started. There is much more work to be done in this area to fully exploit this approach. Also the work on robust control reported in the literature so far assumes a discrete plant model whereas in many cases the along the trial dynamics are governed by a linear differential equation. The theory of differential linear repetitive processes provides an alternative design by emulation possibility and this also requires further work.
- Extend the results to investigate the effects of different reference signals. Currently the reference trajectories are designed specifically for the gantry robot to achieve only a pick and place movement. More complicated trajectories might be used in real industrial applications, therefore the evaluation of ILC algorithms is required to be carried out on more complicated trajectories which contain higher frequency or higher speed movements, higher accelerations or decelerations.

- Developing a better solution for the high frequency noise build up encountered in implementations of ILC algorithms.
- Analysis of the multi-sampling technique.
- Develop and experimentally test averaging algorithms for use in processing of the learning signals in ILC to establish if there is any improvement.
- Use a high-accuracy library in the program development and try to minimise the influence of numerical problems in computation.
- Extend and enhance the duality theory for RC and ILC.

In the non-minimum phase case, An objective-driven ILC scheme is introduced in Freeman et al. (2009). The key departure from the standard ILC framework that will be considered is to permit the reference to change between trials. It is shown that this approach leads to a simple method which has the ability to speed up learning when the intention is not to track a fixed reference, but instead wish to perform a specified point-to-point movement task. This approach is implemented on the non-minimum phase plant and some initial results are produced (Freeman et al., 2009). More investigation are required and extra experiments are being carried out on the plant.

Appendix A

Additional Figures

A.1 Frequency response tests for the gantry robot

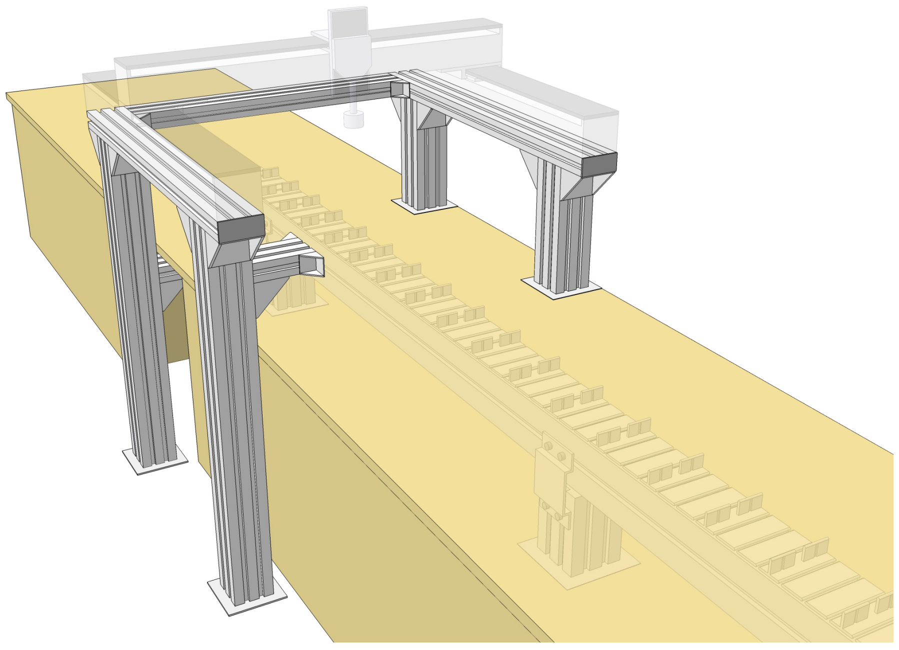


Figure A.1: The base framework of the gantry robot.

Note: Frequency responses of the system depend on the physical structure of the plant. The biggest influence on the structure for the gantry robot is the base framework and the work bench. Here, the framework has been firmly grounded to the work bench by bolted feet, beams and other components.

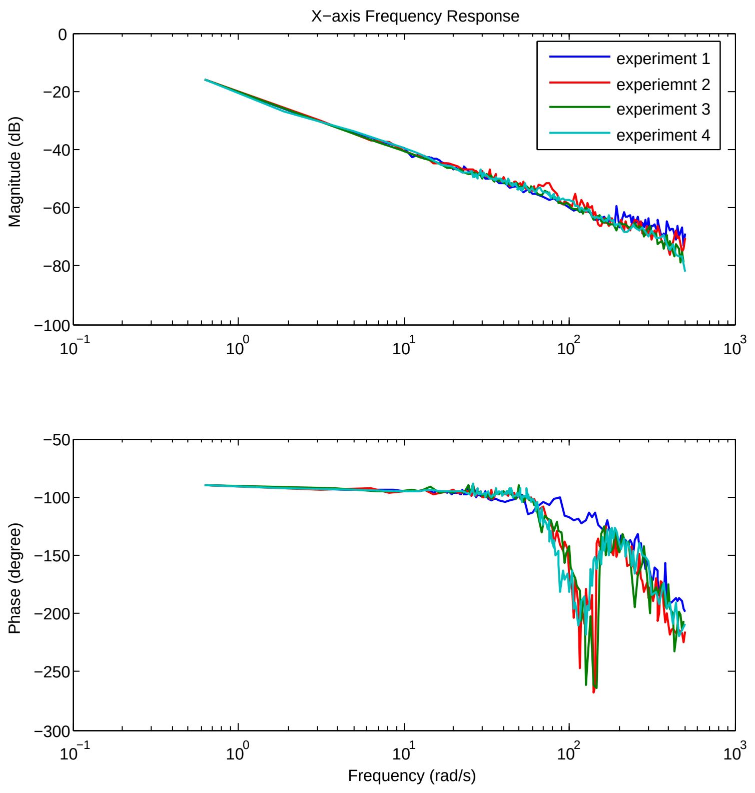


FIGURE A.2: X-axis frequency response (Experiment 1 result has not been used as it has got too much difference with others)

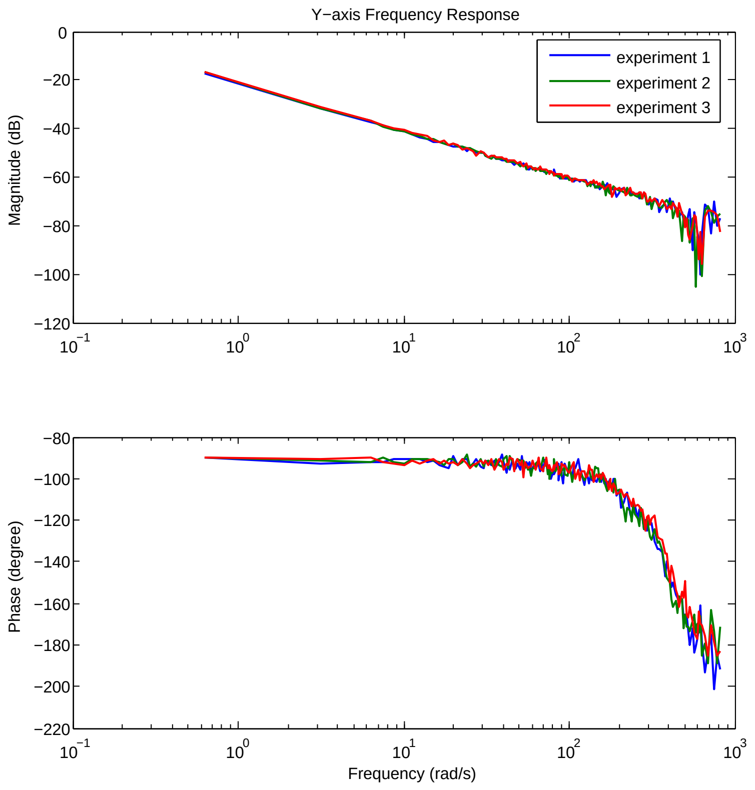


FIGURE A.3: Y-axis frequency responses

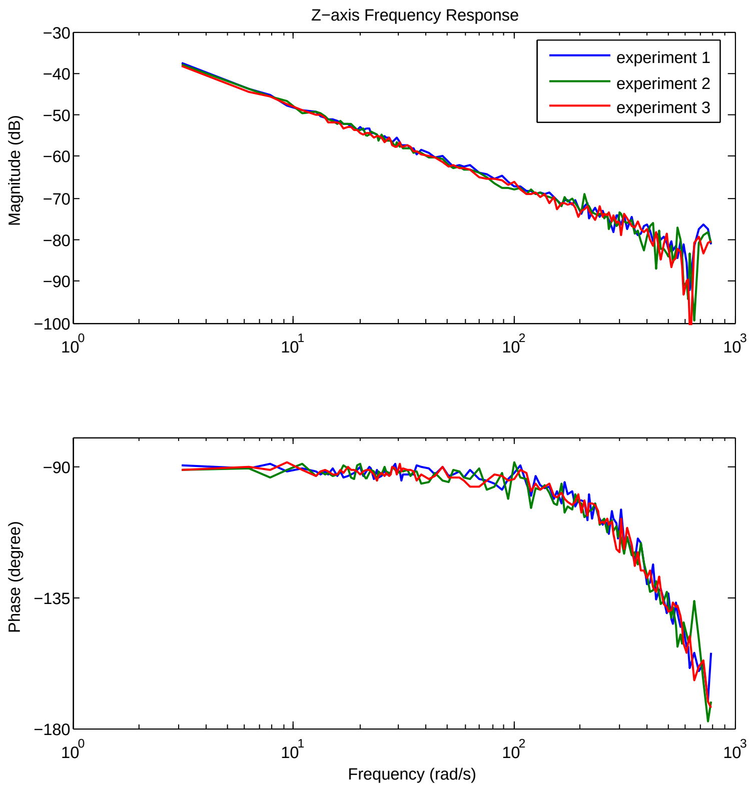


FIGURE A.4: Z-axis frequency response

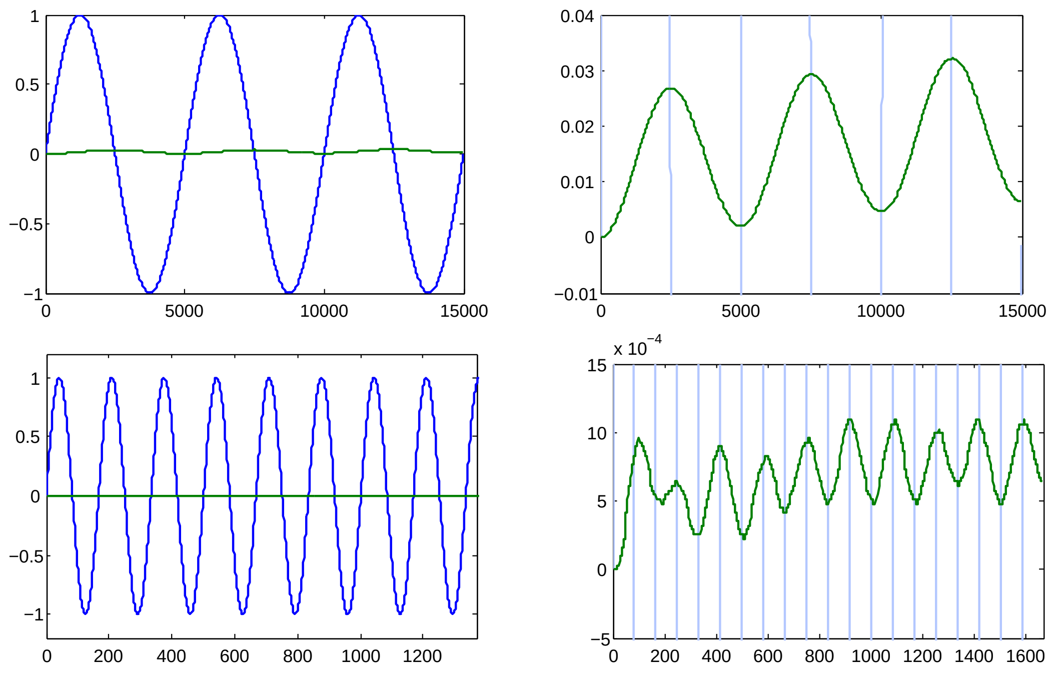


FIGURE A.5: Detailed frequency responses for X-axis at 1Hz and 30Hz

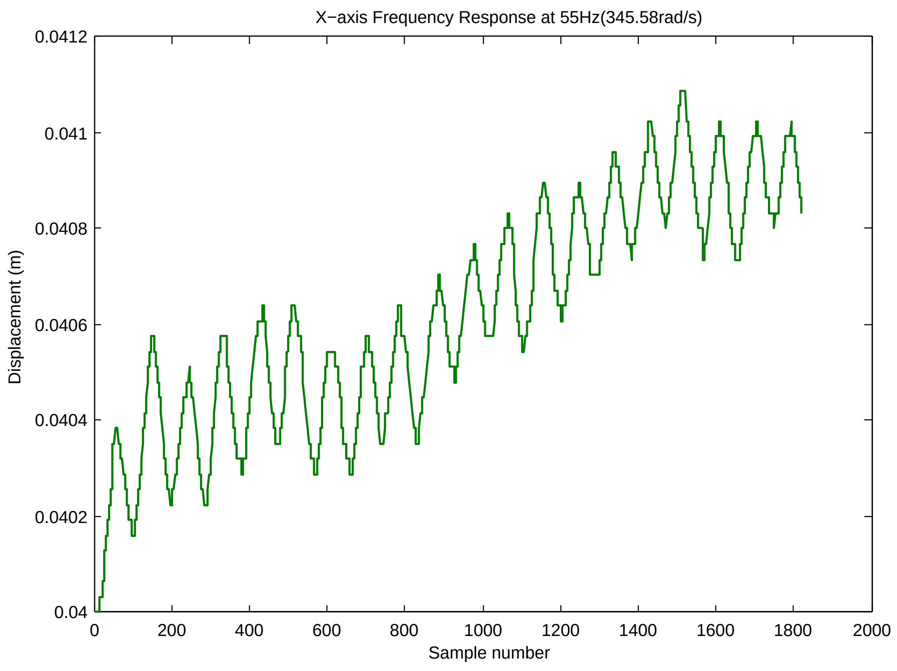


FIGURE A.6: The drift effect in frequency response tests

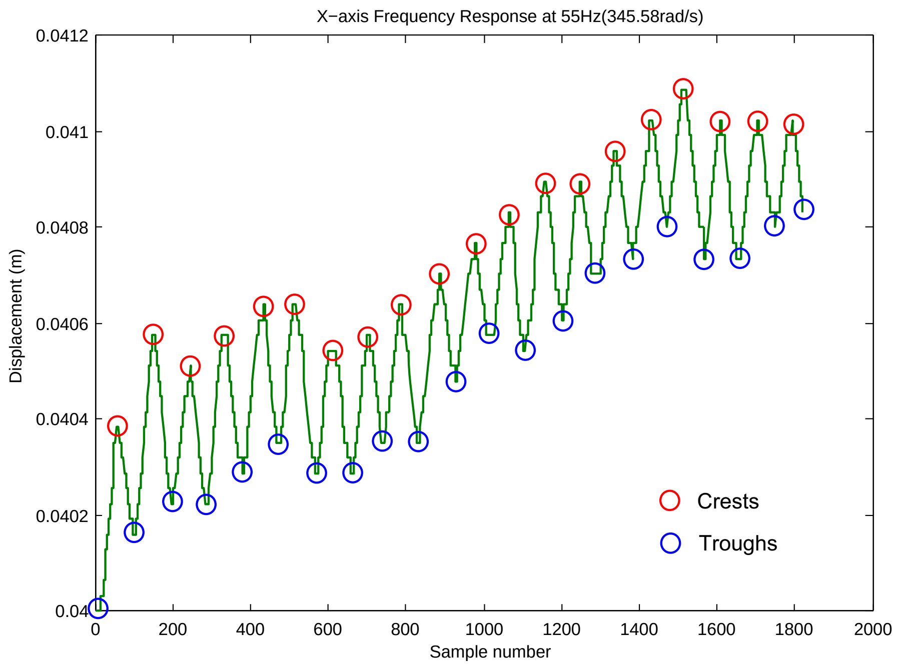


FIGURE A.7: Solution to the drift effect in frequency response tests

Note: The drift effect can be eliminated by computing the average value of all the crests and troughs in the response. The amplitude for the response sine wave is the difference between these two averages.

A.2 State Space Models

A.2.1 Continuous Time Models

X-axis

$$A_X = \begin{bmatrix} -0.03078364485761 \times 10^3 & 0.20298644689641 \times 10^3 & -0.04709572089683 \times 10^3 \\ -0.05074661172410 \times 10^3 & -0.03078364485761 \times 10^3 & 0.08171086612447 \times 10^3 \\ 0 & 0 & -0.11395406895744 \times 10^3 \\ 0 & 0 & -0.20853149459178 \times 10^3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix.$$

$$\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.20853149459178 \times 10^3 & 0.29947455699632 \times 10^3 & 0 & 0 \\ -0.11395406895744 \times 10^3 & -0.00074639754604 \times 10^3 & 0 & 0 \\ 0 & -0.23303901856095 \times 10^3 & 1.49654118008048 \times 10^3 & 1.19790008827019 \times 10^3 \\ 0 & -0.37413529502012 \times 10^3 & -0.23303901856095 \times 10^3 & 0 \\ 0 & 0 & 0 & \end{matrix} \Bigg]$$

$$B_X = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0.22071770344408 \end{bmatrix}^T$$
$$C_X = \begin{bmatrix} 0.07331633313581 & 0 & 0.11035902310463 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Y-axis

$$\begin{aligned} A_Y &= \begin{bmatrix} -2.13370875659248 \times 10^2 & 3.58972168348815 \times 10^2 & 0.12653295628928 \\ -3.58972168348815 \times 10^2 & -2.13370875659248 \times 10^2 & 0.15786804930489 \\ 0 & 0 & 0 \end{bmatrix} \\ B_Y &= \begin{bmatrix} 0 & 0 & 1.21797985624413 \end{bmatrix}^T \\ C_Y &= \begin{bmatrix} 1.54012664142245 & 0 & 0 \end{bmatrix} \end{aligned}$$

Z-axis

$$\begin{aligned} A_Z &= \begin{bmatrix} -3.53805615717504 \times 10^2 & 9.22069362175882 \times 10^2 & 8.44676545460854 \\ -2.30517340543970 \times 10^2 & -3.53805615717504 \times 10^2 & 4.54856964975879 \\ 0 & 0 & 0 \end{bmatrix} \\ B_Z &= \begin{bmatrix} 0 & 0 & 0.12455748002748 \end{bmatrix}^T \\ C_Z &= \begin{bmatrix} 0.15100083012077 & 0 & 0 \end{bmatrix} \end{aligned}$$

A.2.2 Discrete Time Models ($T_s = 0.01$s)

X-axis

$$
\begin{aligned}
A_X &= \begin{bmatrix}
0.38786170479844 & 1.00000000000000 & 0.21377417070825 & 0 \\
-0.38984051991243 & 0.38786170479844 & 0.17438624181313 & 0 \\
0 & 0 & -0.15746031623301 & 0.25000000000000 \\
0 & 0 & -0.31033783641033 & -0.15746031623301 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix. \\
&\qquad \left. \begin{matrix}
0.10406878797622 & 0 & 0.08315257924196 \\
0.08489409532075 & 0 & 0.06783170175813 \\
-0.20061655240719 & 0 & -0.16029574376420 \\
-0.05553532410202 & 0 & -0.04437358720057 \\
0.03528556803047 & 0.50000000000000 & 0.28087745569161 \\
-0.01642801323473 & 0.03528556803047 & -0.27571648446142 \\
0 & 0 & 1.00000000000000
\end{matrix} \right] \\
B_X &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0.09099004806844 \end{bmatrix}^T \\
C_X &= \begin{bmatrix} 0.03905666166991 & 0 & 0.01462020349057 & 0 & 0.00711735590969 & 0 & 0.00568687800428 \end{bmatrix}
\end{aligned}
$$

Y-axis

$$
\begin{aligned}
A_Y &= \begin{bmatrix}
-0.10670673614842 & 0.12500000000000 & 0.07772242425923 \\
-0.02105288038893 & -0.10670673614842 & 0.10157425099610 \\
0 & 0 & 1.00000000000000
\end{bmatrix} \\
B_Y &= \begin{bmatrix} 0 & 0 & 0.02858653106809 \end{bmatrix}^T \\
C_Y &= \begin{bmatrix} 0.03602025169204 & 0 & 0.02858653106809 \end{bmatrix}
\end{aligned}
$$

Z-axis

$$
\begin{aligned}
A_Z &= \begin{bmatrix}
-0.00296119813090 & 0.06250000000000 & 0.07577148130355 \\
-0.01338053396753 & -0.00296119813090 & 0.06370302042894 \\
0 & 0 & 1.00000000000000
\end{bmatrix} \\
B_Z &= \begin{bmatrix} 0 & 0 & 0.01910017507687 \end{bmatrix}^T \\
C_Z &= \begin{bmatrix} 0.02318792188715 & 0 & 0.01910017507687 \end{bmatrix}
\end{aligned}
$$

A.2.3 Discrete Time Models ($T_s = 0.05\text{s}$)

X-axis

$$\begin{aligned}
A_X &= \left[\begin{matrix} 0.07603878804090 & 0.25000000000000 & -0.01291677943914 & 0 \\ -0.16101035153141 & 0.07603878804090 & 0.06188804933559 & 0 \\ 0 & 0 & -0.00180691599037 & 0.01562500000000 \\ 0 & 0 & -0.00051085310399 & -0.00180691599037 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}\right. \\
&\qquad \left.\begin{matrix} -0.00076085893609 & 0 & -0.03156726985626 \\ 0.00364549658807 & 0 & 0.15124797659199 \\ -0.00246559474334 & 0 & -0.10229503910296 \\ 0.00055196958846 & 0 & 0.02290066150881 \\ 0.00000834960583 & 0.00024414062500 & 0.02410277922525 \\ -0.00000002461658 & 0.00000834960583 & -0.00022057464204 \\ 0 & 0 & 1.00000000000000 \end{matrix}\right] \\
B_X &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0.12948437388010 \end{bmatrix}^T \\
C_X &= \begin{bmatrix} 0.02551198085987 & 0 & 0.01324569012334 & 0 & 0.00078023331919 & 0 & 0.03237109347002 \end{bmatrix}
\end{aligned}$$

Y-axis

$$\begin{aligned}
A_Y &= \begin{bmatrix} 0.00001444472309 & 0.00390625000000 & 0.55207760786942 \\ -0.00000008515284 & 0.00001444472309 & 0.00615014490692 \\ 0 & 0 & 1.00000000000000 \end{bmatrix} \\
B_Y &= \begin{bmatrix} 0 & 0 & 0.06645205446018 \end{bmatrix}^T \\
C_Y &= \begin{bmatrix} 0.00229291820620 & 0 & 0.06645205446018 \end{bmatrix}
\end{aligned}$$

Z-axis

$$\begin{aligned}
A_Z &= \begin{bmatrix} -0.00000001013794 & 0.00012207031250 & 0.54740161698385 \\ -0.00000000000269 & -0.00000001013794 & 0.00012935089288 \\ 0 & 0 & 1.00000000000000 \end{bmatrix} \\
B_Z &= \begin{bmatrix} 0 & 0 & 0.04430837974574 \end{bmatrix}^T \\
C_Z &= \begin{bmatrix} 0.00151590491992 & 0 & 0.04430837974574 \end{bmatrix}
\end{aligned}$$

Appendix B

Additional Results

B.1 Reference Shift Algorithm

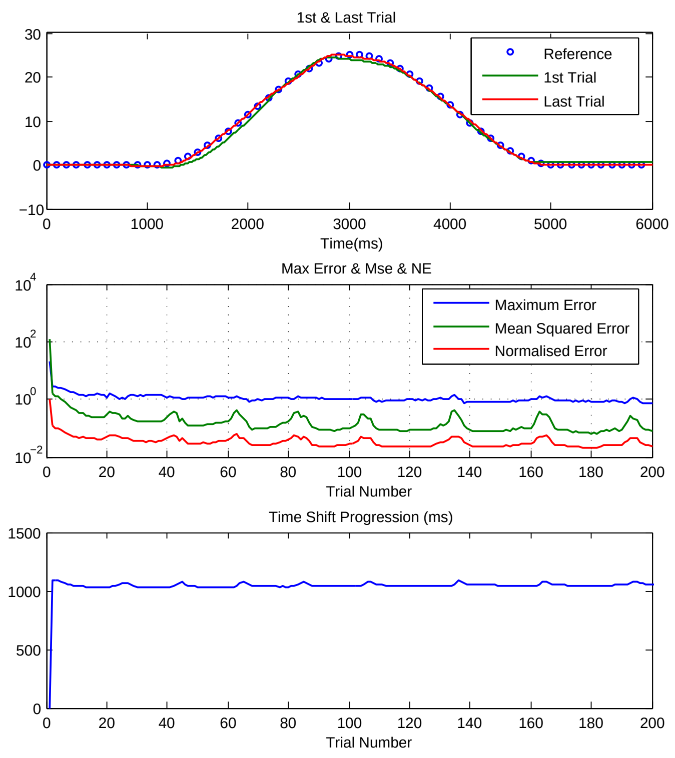


Figure B.1: Additional experimental results for the Reference Shift algorithm (Parameters: $K_p = 0.105, K_i = 0.005, K_d = 0.003, L = 0.1$).

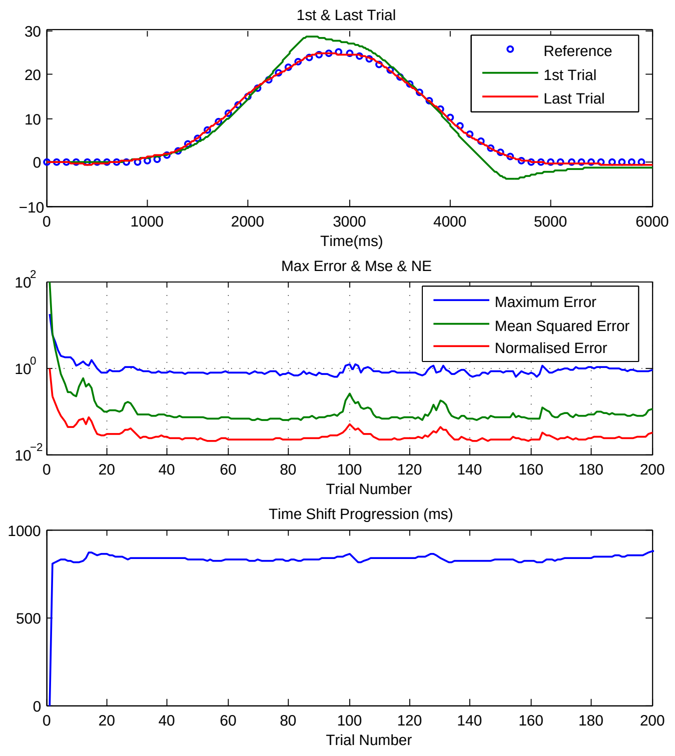


FIGURE B.2: Additional experimental results for the Reference Shift algorithm (Parameters: $K_p = 0.138, K_i = 0.005, K_d = 0.003, L = 0.3$).

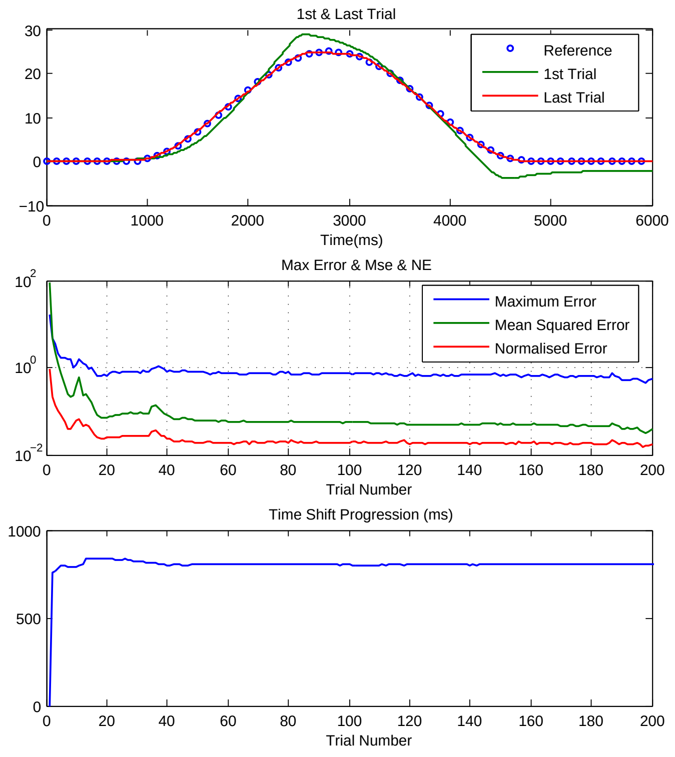


FIGURE B.3: Additional experimental results for the Reference Shift algorithm (Parameters: $K_p = 0.138, K_i = 0.005, K_d = 0.003, L = 0.3$).

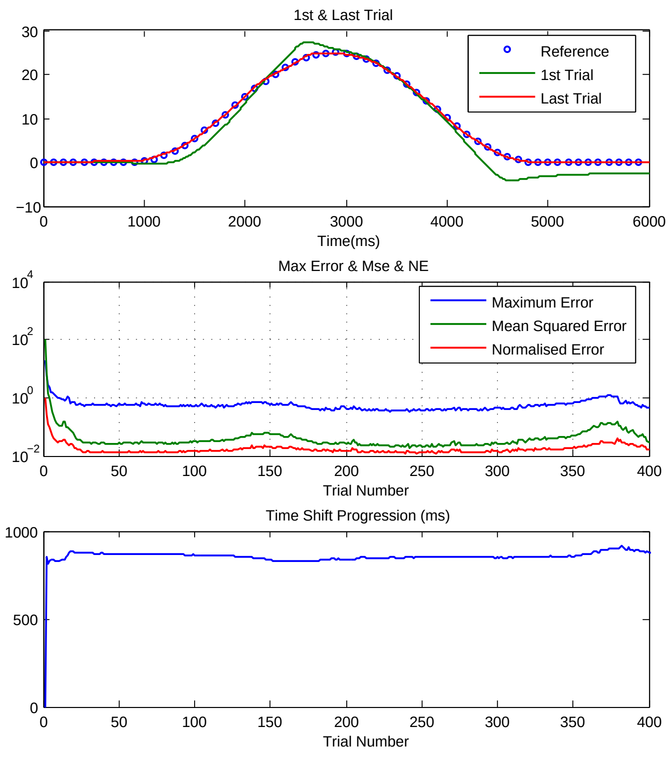


FIGURE B.4: Additional experimental results for the Reference Shift algorithm. (Parameters: $K_p = 0.138, K_i = 0.005, K_d = 0.003, L = 0.3$, Filter: zero-phase Chebyshev low pass filter with cut off frequency of 1Hz).

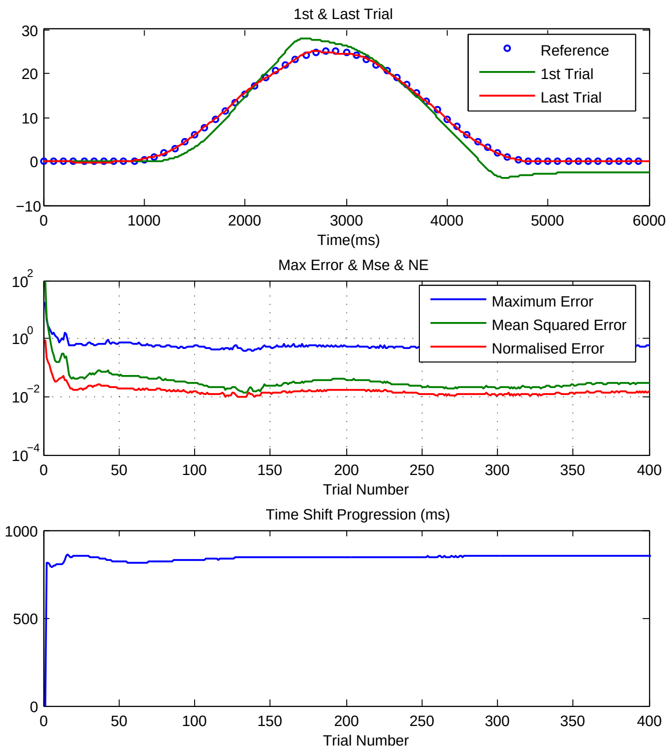


FIGURE B.5: Additional experimental results for the Reference Shift algorithm. (Parameters: $K_p = 0.138, K_i = 0.005, K_d = 0.003, L = 0.3$, Filter: zero-phase Chebyshev low pass filter with cut off frequency of 0.85Hz).

B.2 Stochastic Learning Algorithms

B.2.1 Simulation of the D-type stochastic algorithm

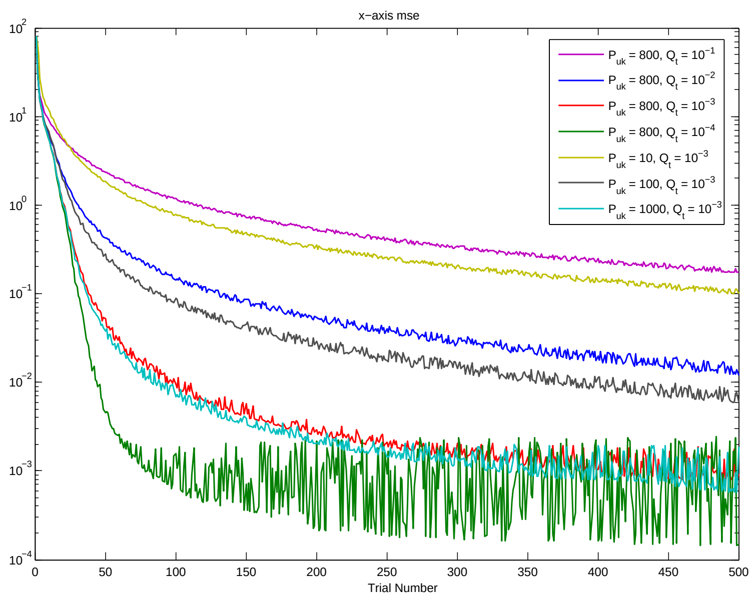


FIGURE B.6: Parameter tuning effects for the D-type stochastic learning algorithm

B.2.2 Simulation of the P-type stochastic algorithm

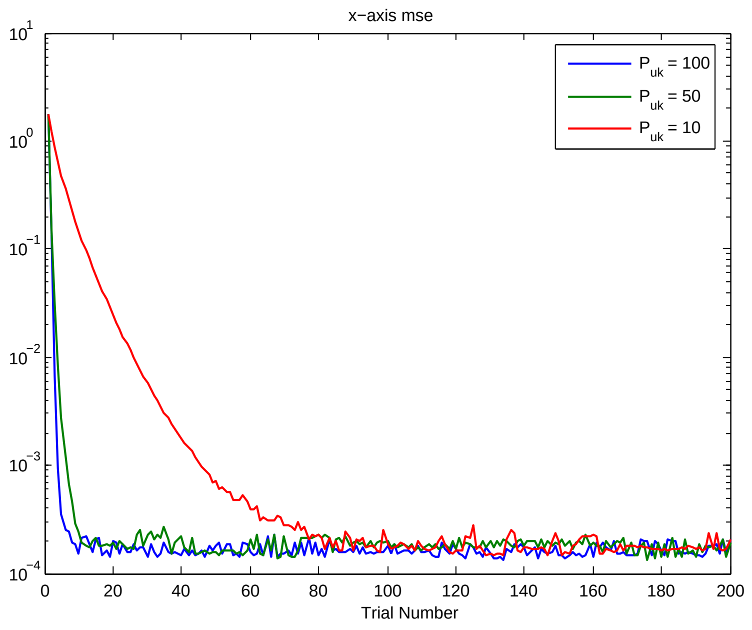


FIGURE B.7: Parameter tuning ($P_{u,0}$) effects for the D-type stochastic learning algorithm

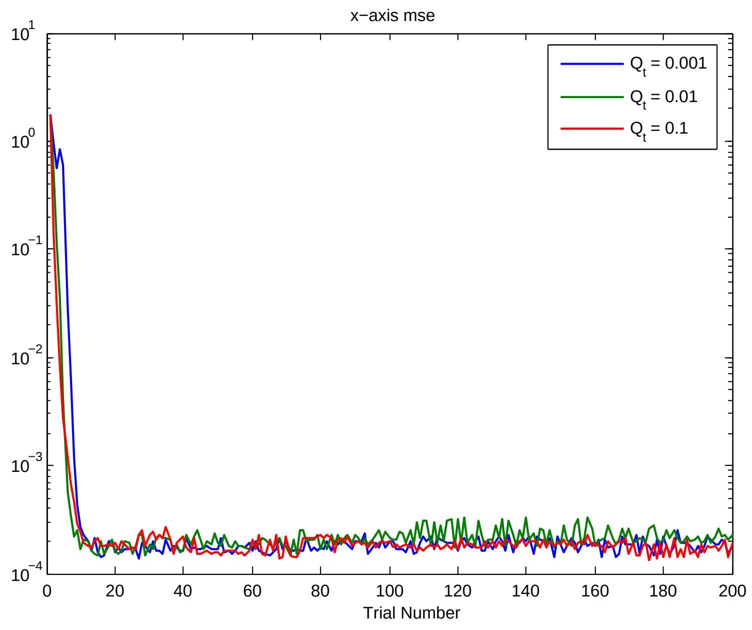


FIGURE B.8: Parameter tuning (Q_t) effects for the D-type stochastic learning algorithm

B.2.3 Verification of filter arrangement

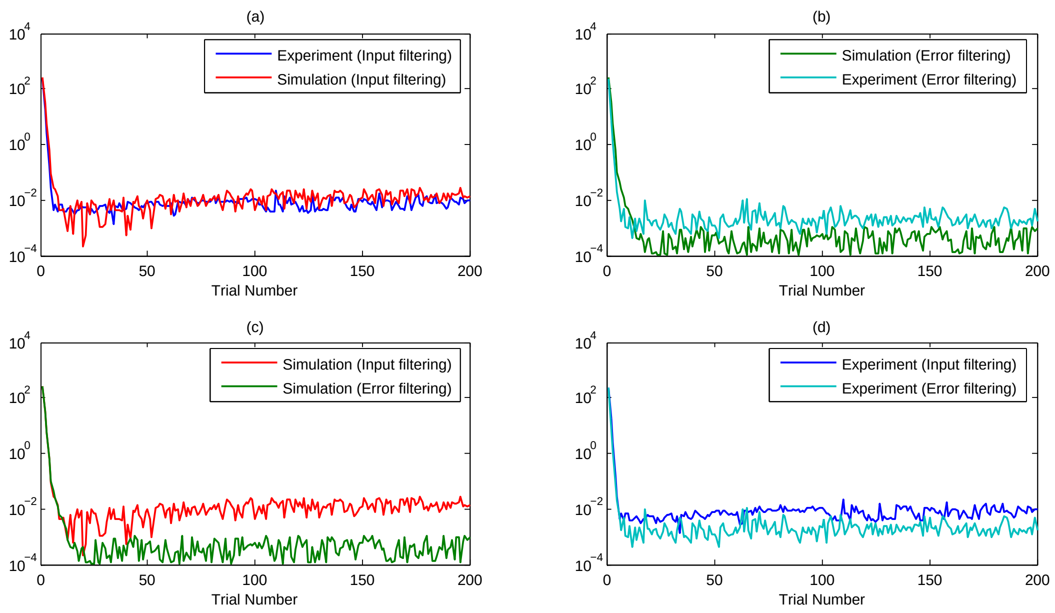
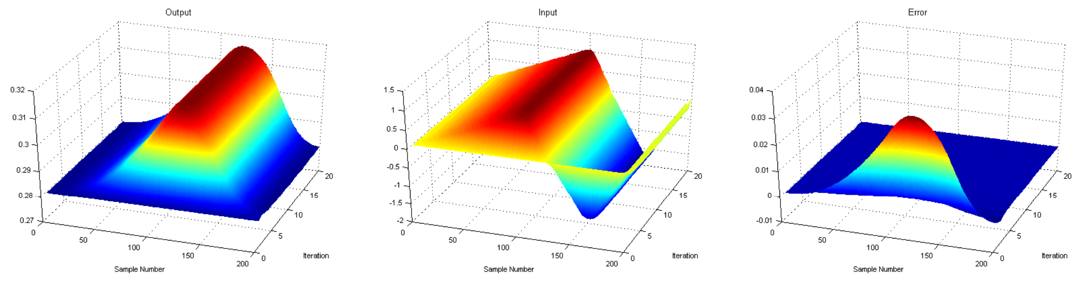


FIGURE B.9: Comparison of effort of filter arrangement in simulation and experiment.

B.2.4 Additional experimental results

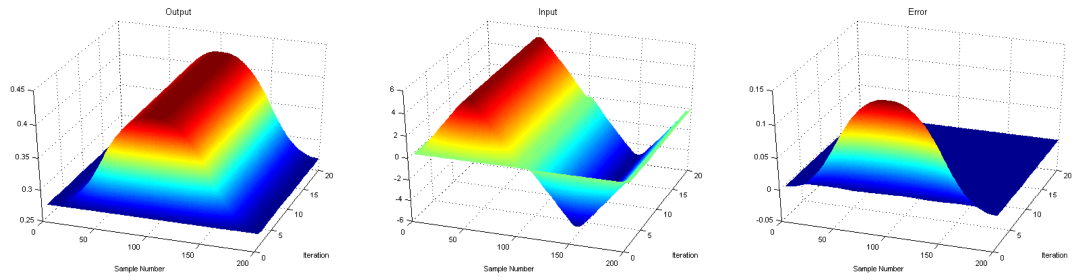
D-type stochastic ILC

Parameters: $P_{u,0} = 100, Q_t = 0.001, P_{x,0} = 0.1, PID = \{0, 0, 0\}$



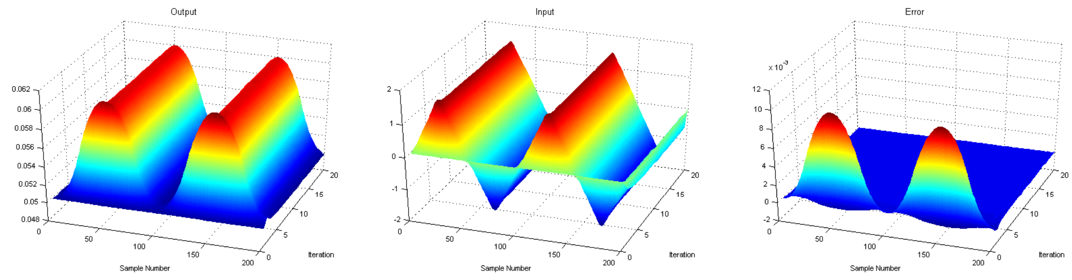
(a) (b) (c)

FIGURE B.10: The D-type stochastic learning algorithm experimental results (X-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics



(a) (b) (c)

FIGURE B.11: The D-type stochastic learning algorithm experimental results (Y-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics

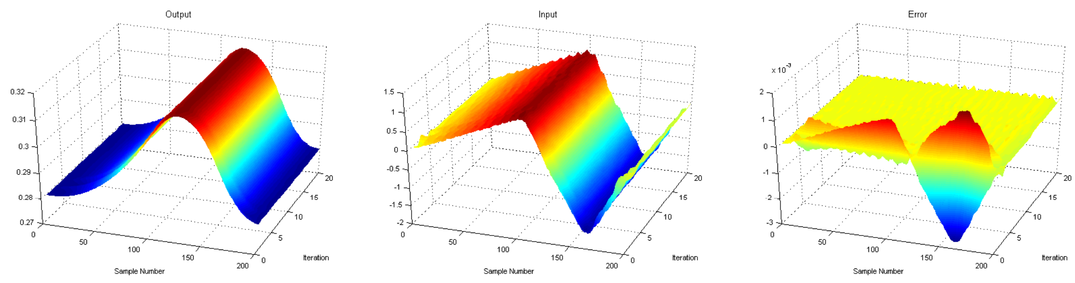


(a) (b) (c)

FIGURE B.12: The D-type stochastic learning algorithm experimental results (Z-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics

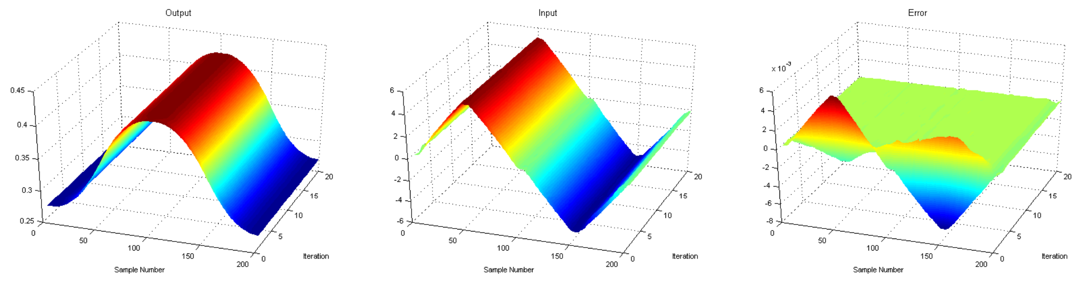
P-type stochastic ILC

Parameters: $P_{u,0} = 50, Q_t = 0.01, P_{x,0} = 0.1, PID(x) = \{600, 300, 0.2\}, PID(y) = \{800, 300, 0.2\}, PID(z) = \{1100, 300, 0.2\}$



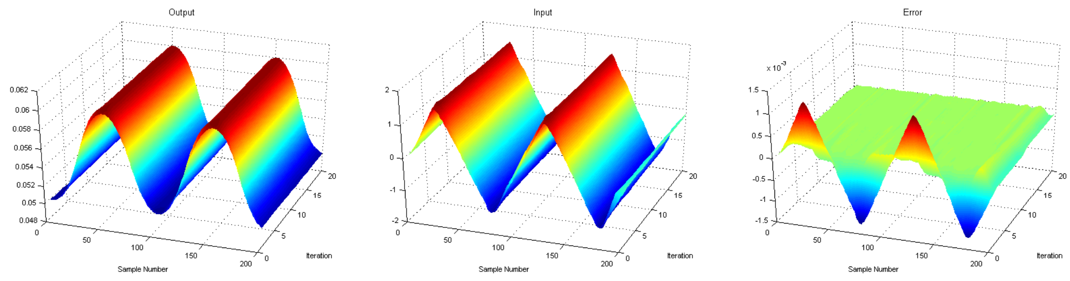
(a) (b) (c)

FIGURE B.13: The P-type stochastic learning algorithm experimental results (X-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics



(a) (b) (c)

FIGURE B.14: The P-type stochastic learning algorithm experimental results (Y-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics

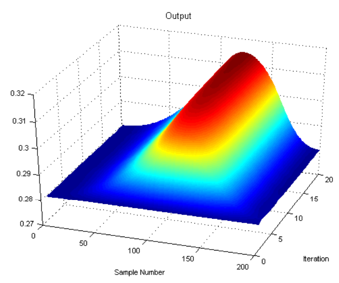


(a) (b) (c)

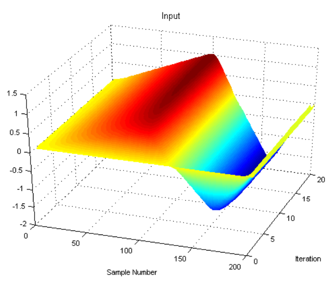
FIGURE B.15: The P-type stochastic learning algorithm experimental results (Z-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics

Sub-optimal stochastic ILC

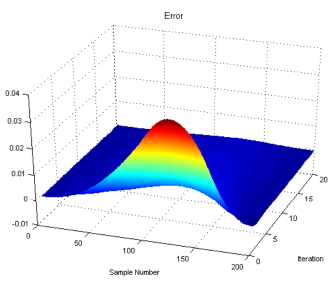
Parameters: $P_{u,0} = 100, Q_t = 0.01$



(a)

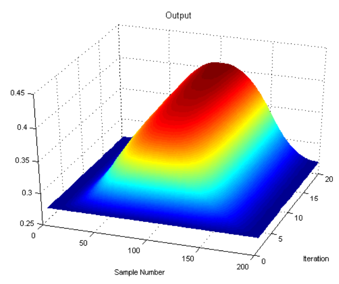


(b)

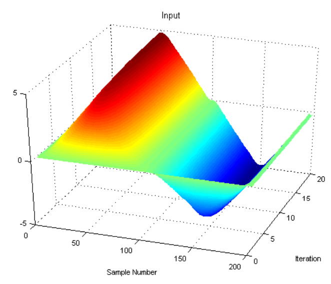


(c)

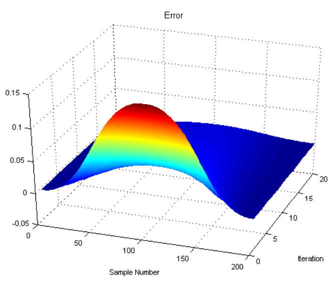
FIGURE B.16: The sub-optimal stochastic learning algorithm experimental results (X-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics



(a)

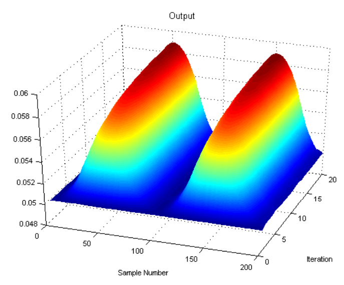


(b)

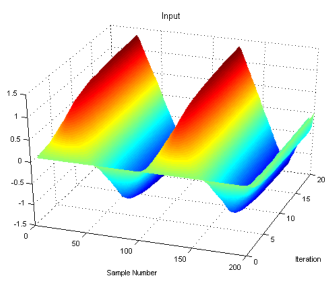


(c)

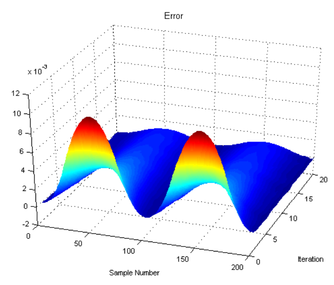
FIGURE B.17: The sub-optimal stochastic learning algorithm experimental results (Y-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics



(a)



(b)



(c)

FIGURE B.18: The sub-optimal stochastic learning algorithm experimental results (Z-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics

Long term stability test

Parameters: $P_{u,0} = 100, Q_t = 0.01, P_{x,0} = 0.1, PID(x) = \{600, 300, 0.2\}, PID(y) = \{800, 300, 0.2\}, PID(z) = \{1100, 300, 0.2\}$

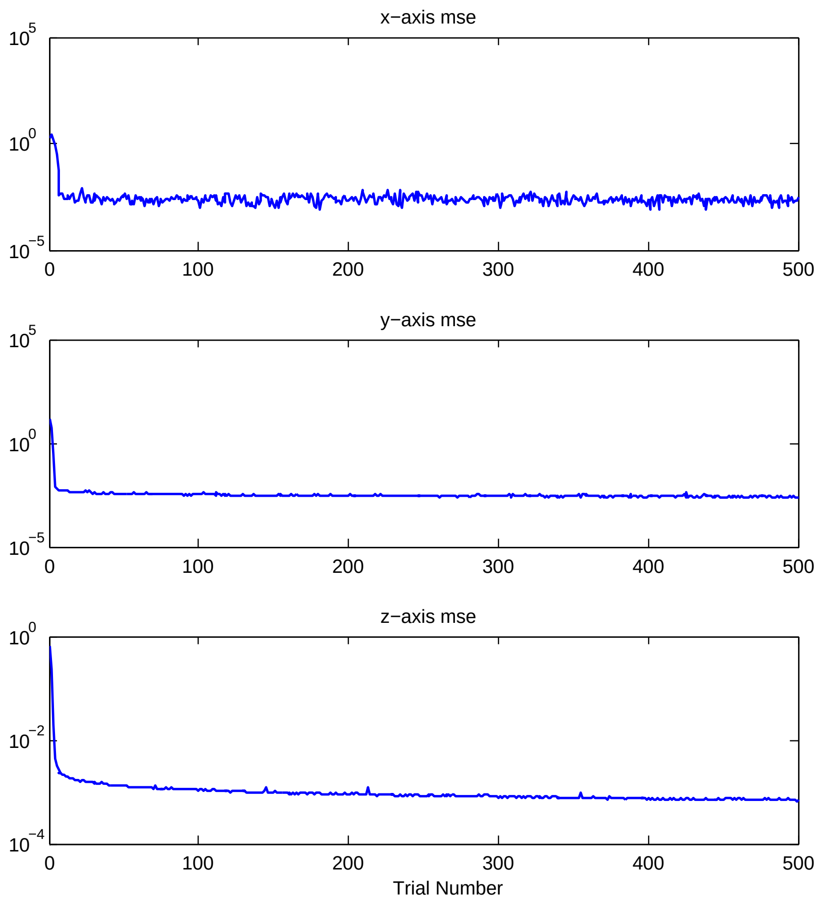
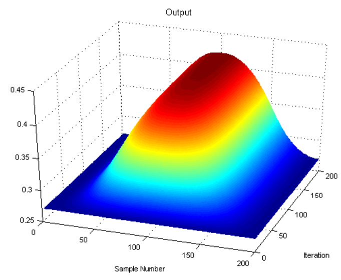


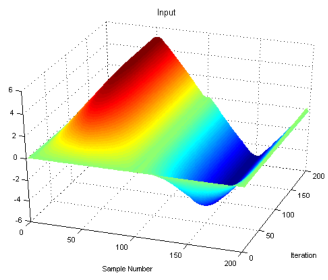
FIGURE B.19: Long term stability testing for the P-type stochastic learning algorithm.

B.3 2D ILC Algorithms

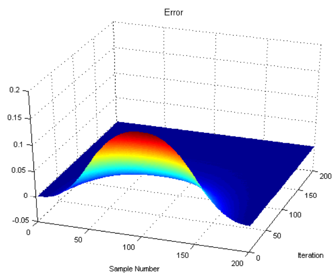
B.3.1 Y-axis results (State-feedback controller)



(a)



(b)



(c)

FIGURE B.20: The state-feedback 2D ILC algorithm experimental results (Y-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics

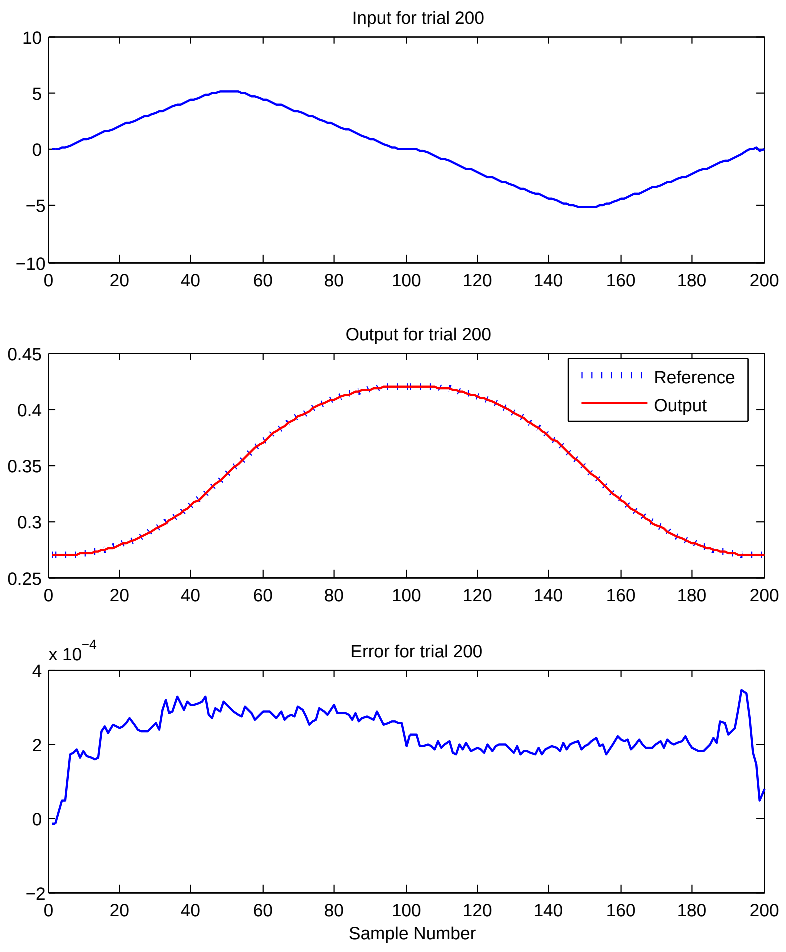
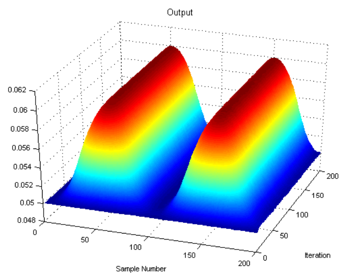
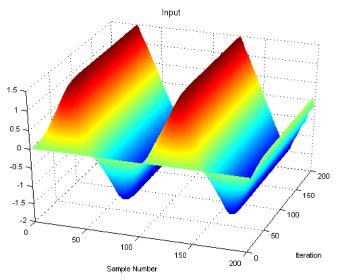


FIGURE B.21: Along the trial performance for the state-feedback 2D ILC algorithm (Y-axis).

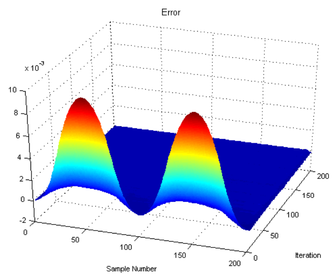
B.3.2 Z-axis results (State-feedback controller)



(a)



(b)



(c)

FIGURE B.22: The state-feedback 2D ILC algorithm experimental results (Z-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics

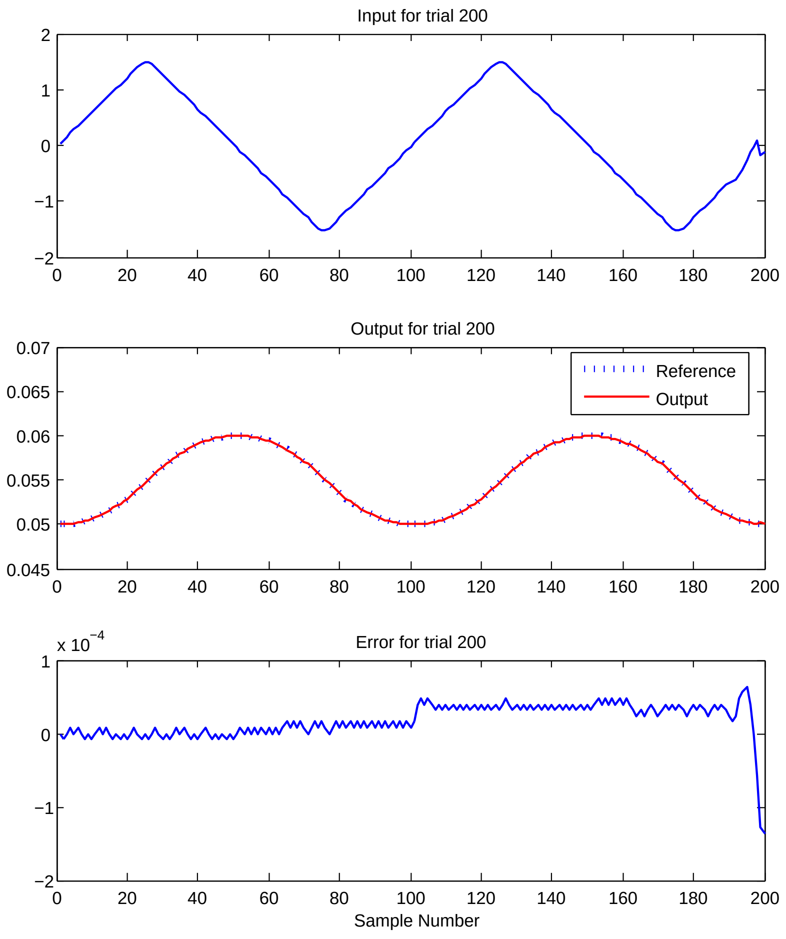
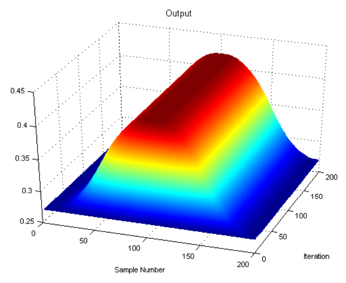
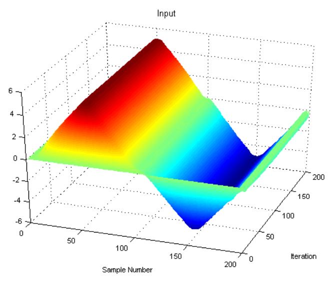


FIGURE B.23: Along the trial performance for the state-feedback 2D ILC algorithm (Z-axis).

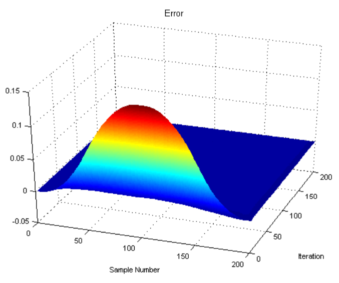
B.3.3 Y-axis results (Output-feedback controller)



(a)



(b)



(c)

FIGURE B.24: The output-feedback 2D ILC algorithm experimental results (Y-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics

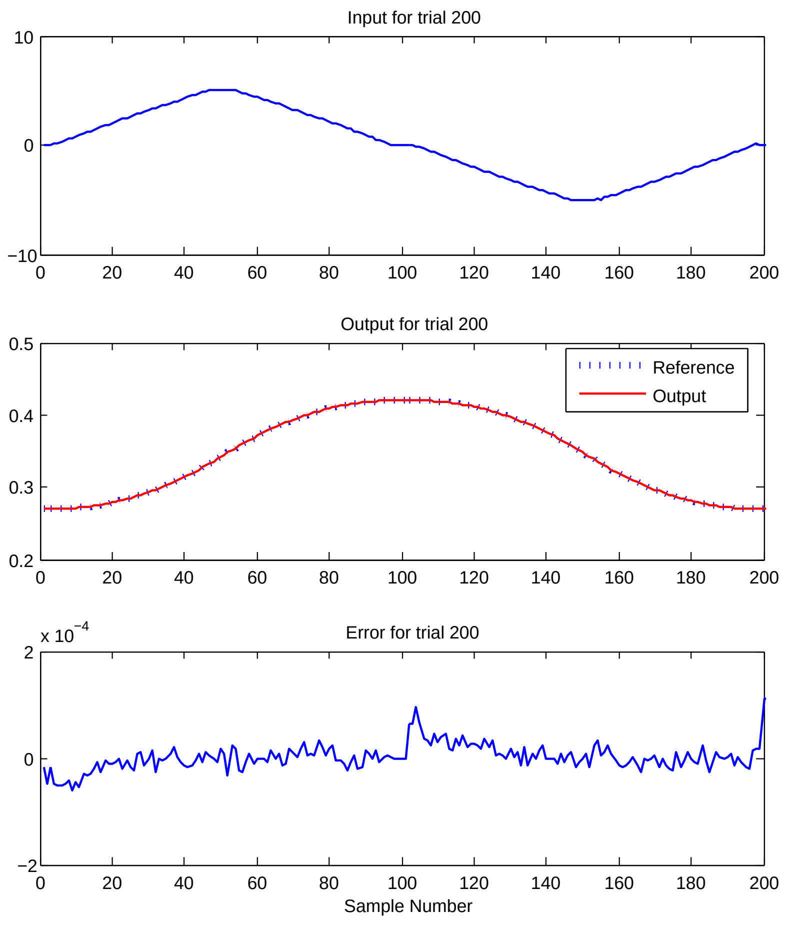
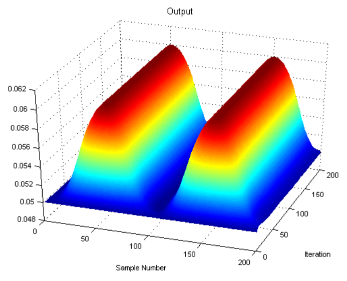
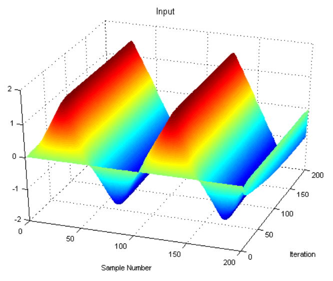


FIGURE B.25: Along the trial performance for the output-feedback 2D ILC algorithm (Y-axis).

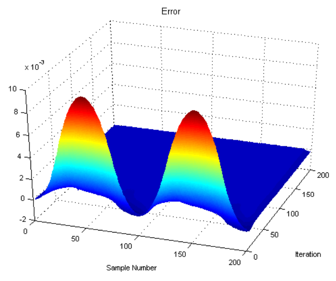
B.3.4 Z-axis results (Output-feedback controller)



(a)



(b)



(c)

FIGURE B.26: The output-feedback 2D ILC algorithm experimental results (Z-axis): (a) Evolution of the output, (b) Evolution of the control input, (c) Evolution of the error dynamics

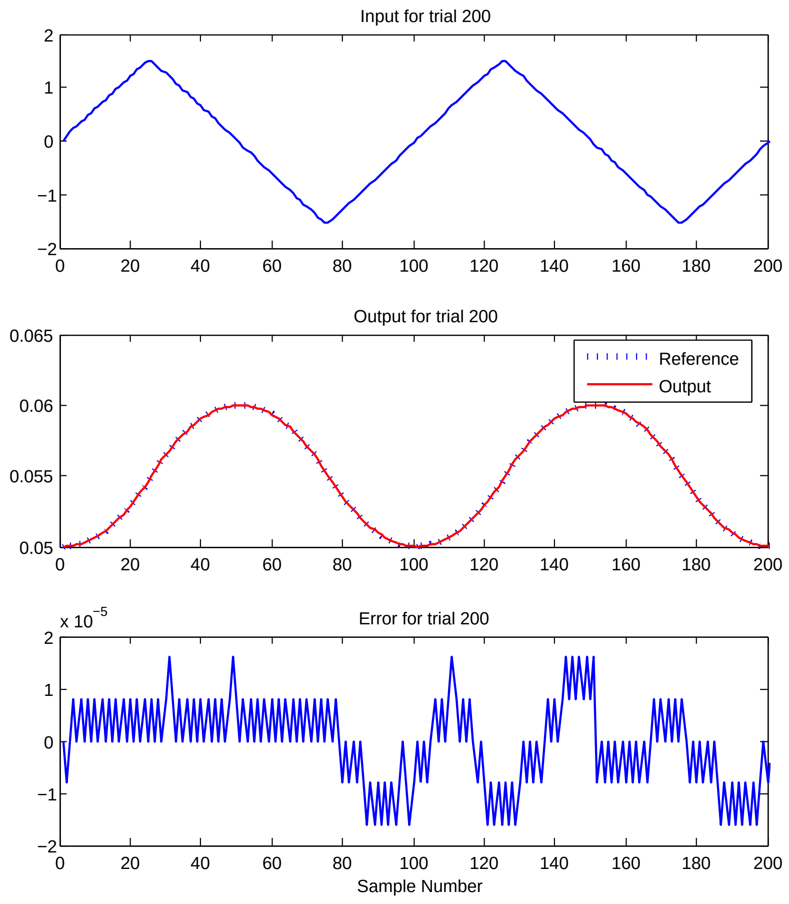


FIGURE B.27: Along the trial performance for the output-feedback 2D ILC algorithm (Z-axis).

Bibliography

H. Ahn, C. Choi, and K. Kim. Iterative learning control for a class of nonlinear systems. *Automatica*, 29:1575–1578, 1993.

H-S. Ahn, Y. Chen, and K. L. Moore. Iterative learning control: brief survey and categorization 1998–2004. *IEEE Transactions on Systems Man and Cybernetics Part C*, 37(6):1099–1121, November 2007.

N. Amann and D. H. Owens. non-minimum phase plants in iterative learning control. In *Proceedings of 2nd international conference on Intelligent systems engineering*, pages 107–112, Hamburg-Harbourg, Germany, 1994.

N. Amann, D. H. Owens, and E. Rogers. Iterative learning control for discrete-time systems with exponential rate of convergence. *Control Theory and Applications, IEE Proceedings*, 143(2):217–224, March 1996a.

N. Amann, D. H. Owens, and E. Rogers. Iterative learning control using optimal feedback and feedforward actions. *International Journal of Control*, 65:277–293, 1996b.

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

S. Arimoto, F. Miyazaki, and S. Kawamura. Bettering operation of dynamical systems by learning: a new control theory for servomechanism or mechatronics systems. In *Proceedings of the 23rd Conference on Decision and Control*, pages 1064–1069, Las Vegas, Nevada, 1984b.

S. Arimoto, F. Miyazaki, S. Kawamura, and S. Tamaki. Learning control theory for dynamical systems. In *Proceedings of the 24th Conference on Decision and Control*, pages 1375–1380, Fort Lauderdale, Florida, 1985.

A. Barton, P. L. Lewin, and D. Brown. Practical implementation of a real-time iterative learning position controller. *International Journal of Control*, 73(10):992–999, 2000.

Z. Bien and K. Huh. Higher-order iterative learning control algorithm. In *Proceedings of the Institution of Electrical Engineers, Control Theory and Applications*, volume 136, pages 105–112, 1989.

P. Bondi, G. Casalino, and L. Gambardella. On the iterative learning control theory for robotic manipulators. *IEEE Journal of Robotics and Automation*, 4(1):14–22, February 1988.

D. A. Bristow. Frequency domain analysis and design of iterative learning control for systems with stochastic disturbances. In *Proceedings of the American Control Conference*, pages 3901–3907, Seattle, Washington, USA, June 2008.

D. A. Bristow, M. Tharayil, and A. G. Alleyne. A survey of iterative learning control a learning-based method for high-performance tracking control. *IEEE Control Systems Magazine*, 26(3):96–114, 2006.

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

Z. Cai, C. T. Freeman, E. Rogers, and P. L. Lewin. Reference shift iterative learning control for a non-minimum phase plant. In *Proceddings of 2007 American Control Conference*, pages 558–563, New York, NY, USA, July 2007.

H-F. Chen and H-T. Fang. Output tracking for nonlinear stochastic systems by iterative learning control. *IEEE Transactions on Automatic Control*, 49(4):583–588, 2004.

K. Chen and R.W. Longman. Stability issues using FIR filtering in repetitive control. *Advances in the Astronautical Sciences*, 112(2):1321–1340, 2002.

Y. Chen, C. Wen, Z. Gong, and M. Sun. An iterative learning controller with initial state learning. *IEEE Transactions on Automatic Control*, 44 (2):371–376, 1999.

Y. Chen, C. Wen, J-X. Xu, and M. Sun. An initial state learning method for iterative learning control of uncertain time-varying systems. In *Proceedings of the 35th IEEE Conference on Decision and Control*, pages 3996–4001, Kobe, Japan, 1996.

K. K. Chew and M. Tomizuka. Digital control of repetitive errors in disk-drive systems. *IEEE Control Systems Magazine*, 10:16–20, January 1990.

J-W. Choi, H-G. Choi, K-S. Lee, and W-H Lee. Control of ethanol concentration in a fed-batch cultivation of acinetobactoer calcoaceticus rag-1 using a feedback-assisted iterative learning algorithm. *Journal of Biotechnology*, 46:29–43, 1996.

B. G. Dijkstra and O. H. Bosgra. Convergence design considerations of low order !-ilc for closed loop systems, implemented on a high precision wafer stage. In *Proceedings of the 41th IEEE Conference on Decision and Control*, pages 2494–2499, Las Vegas, Nevada USA, December 2002a.

B. G. Dijkstra and O. H. Bosgra. Extrapolation of optimal lifted system ilc solution, with application to a waferstage. In *Proceedings of the 2002 American Control Conference*, pages 2595–260, Anchorage, AK, USA, May 2002b.

T. Donkers, J. van de Wijdeven, and O. Bosgra. Robustness against model uncertainties of norm optimal iterative learning control. In *Proceedings of the 2008 American Control Conference*, pages 4561–4566, Seattle, WA, USA, 11-13 June 2008.

K. Dutton, S. Thompson, and B. Barraclough. *The art of control engineering*. Addison-Wesley, 1998. ISBN 0-201-17545-2.

H. Elci, R.W. Longman, M. Phan, J-N. Juang, and R. Ugoletti. Discrete frequency based learning control for precision motion control. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2767–2773, San Antonio, TX, 1994.

E. Fornasini and G. Marchesini. Doubly-indexed dynamical systems: state space models and structural properties. *Mathematical Systems Theory*, 12: 59–72, 1978.

C. Freeman, A-M. Hughes, J. Burridge, P. Chappell, P. L. Lewin, and E. Rogers. Iterative learning control of FES applied to the upper extremity for rehabilitation. *Control Engineering Practice*, 2008a. In Press.

C. T. Freeman. *Experimental Evaluation of Iterative Learning Control on a Non-minimum Phase Plant*. PhD thesis, School of Electronics and Computer Science, University of Southampton, 2004.

C. T. Freeman, Z. Cai, E. Rogers, and P. L. Lewin. Objective-driven ILC for point-to-point movement tasks. In *2009 American Control Conference*, June 2009. In Press.

C. T. Freeman, A-M. Hughes, J. Burridge, P. Chappell, P. L. Lewin, and E. Rogers. A model of the upper extremity using FES for stroke rehabilitation. *ASME Journal of Biomechanical Engineering*, 2008b. In Press.

C. T. Freeman, P. L. Lewin, and E. Rogers. Discrete predictive optimal ilc implemented on a non-minimum phase experimental test-bed. In *Proceedings of 2005 American Control Conference*, pages 282–287, Portland, OR, 2005a.

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

Mark French. Robust stability of iterative learning control schemes. *International Journal of Robust and Nonlinear Control*, 18(10):1018–1033, 2008.

J. A. Frueh and M. Q. Phan. Linear quadratic optimal learning control. *International Journal of Control*, 73(10):832–839, 2000.

F. R. Gao, Y. Yang, and C. Shao. Robust iterative learning control with applications to injection molding process. *Chemical Engineering Science*, 56(24):7025–7034, 2002.

Z. Geng, R. Carroll, and J. Xie. Two-dimensional model and algorithm analysis for a class of iterative learning control systems. *International Journal of Control*, 52(4):833–862, 1990.

J. Ghosh and B. Paden. Pseudo-inverse based iterative learning control for nonlinear plants with disturbances. In *Proceedings of the 38th IEEE Conference on Decision and Control*, pages 5206–5212, Phoenix, Arizona, USA, December 1999.

J. Ghosh and B. Paden. Iterative learning control for nonlinear nonminimum phase plants. *Journal of Dynamic Systems, Measurement, and Control*, 123:21–30, 2001.

J. Ghosh and B. Paden. A pseudoinverse-based iterative learning control. *IEEE Transactions on Automatic Control*, 47(5):831C836, May 2002.

S. Gunnarsson and M. Norrlöf. Some aspects of an optimization approach to iterative learning control. In *Proceedings of the 38th IEEE Conference on Decision and Control*, pages 1581–1586, Phoenix, Arizona, December 1999.

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

H. Havlicsek and A. Alleyne. Nonlinear control of an electrohydraulic injection molding machine via iterative adaptive learning. *IEEE-ASME Transaction on Mechatronics*, 4(3):312–323, 1999.

G. Heinzinger, D. Fenwick, B. Paden, and F. Miyazaki. Robust learning control. In *Proceedings of the IEEE Conference on Decision and Control Including The Symposium on Adaptive Processes*, volume 1, pages 436–440, 1989.

L. Hladowski, Z. Cai, K. Galkowski, E. Rogers, C. T. Freeman, and P. L. Lewin. Using 2D systems theory to design output signal based iterative learning control laws with experimental verification. In *Proceddings of 47th IEEE Conference on Decision and Control*, pages 3026–3031, Cancun, Mexico, December 2008a.

L. Hladowski, Z. Cai, K. Galkowski, E. Rogers, C. T. Freeman, and P. L. Lewin. A novel 2D systems approach to iterative learning control with applications to electromechanical systems. In *Proceddings of European Control Conference 2009*, Budapest, Hungary, August 2009. Submitted.

L. Hladowski, K. Galkowski, Z. Cai, E. Rogers, C. T. Freeman, and P. L. Lewin. A 2D systems approach to iterative learning control with experimental validation. In *Proceddings of 17th IFAC World Congress*, pages 2832–2837, Seoul, Korea, July 2008b.

Y-C. Huang and R. W. Longman. The source of the often observed property of initial convergence followed by divergence in learning and repetitive control. *Advances in the Astronautical Sciences*, 90:555–572, 1996.

D. H. Hwang, B. K. Kim, and Z. Bien. Decentralized iterative learning control methods for large scale linear dynamic systems. *International Journal of Systems Science*, 24(12):2239–2254, 1993.

S. Kawamura, F. Miyazaki, and S. Arimoto. Applications of learning method for dynamic control of robot manipulators. In *Proceedings of the 24th IEEE Conference on Decision and Control*, pages 1381–1386, Ft. Lauderdale, FL, December 1985.

S. Kawamura, F. Miyazaki, and S. Arimoto. Realization of robot motion based on a learning method. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):126–133, 1988.

S. Kichhoff, C. Schmidt, G. Lichtenberg, and H. Werner. An iterative learning algorithm for control of an accelerator based free electron laser. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 3032–3037, Cancun, Mexico, December 2008.

D-I. Kim and S. Kim. An iterative learning control method with application for cnc machine tools. *IEEE transaction on Industry Applications*, 32: 66–72, 1996.

J. E. Kurek and M. B. Zaremba. Iterative learning control synthesis based on 2-D system theory. *IEEE Transactions on Automatic Control*, 38(1): 121–125, 1993.

H-S. Lee and Z. Bien. Study on robustness of iterative learning control with non-zero initial error. *International Journal of Control*, 64(3):345–359, 1996.

J. H. Lee, K. S. Lee, and W. C. Kim. Model-based iterative learning control with a quadratic criterion for time-varying linear systems. *Automatica*, 36: 641–657, 2000.

P. L. Lewin. Iterative learning control of repetitive processes. In *Proceedings of the 2nd International Conference on Climbing and Walking Robots*, pages 295–303, Portsmouth, UK, 1999.

Y-J. Liang and D. P. Looze. Performance and robustness issues in iterative learning control. In *Proceedings of the 32nd IEEE Conference on Decision and Control*, pages 1990–1995, San Antonio, Texas, USA, December 1993.

R. W. Longman. Iterative learning control and repetitive control for engineering practice. *International Journal of Control*, 73:930–954, 2000.

A. De Luca and S. Panzieri. An iterative scheme for learning gravity compensation in flexible robot arms. *Automatica*, 30(6):993–1002, 1994.

O. Markusson, H. Hjalmarsson, and M. Norrlof. Iterative learning control of nonlinear non-minimum phase systems and its application to system and model inversion. In *Proceedings of the 40th Conference on Decision and Control*, pages 4481–4482, Orlando, Florida, 2001.

M. Mezghani, G. Roux, M. Cabassud, B. Dahhou, M. V. Le Lann, and G. Casamatta. Robust iterative learning control of an exothermic semi-batch chemical reactor. *Mathematics and Computers in Simulation*, 57 (6):367–385, 2001.

M. Mezghani, G. Roux, M. Cabassud, M.V. Le Lann, B. Dahhou, and G. Casamatta. Application of iterative learning control to an exothermic semibatch chemical reactor. *IEEE Transactions on Control Systems Technology*, 10(6):822–834, November 2002.

T. Mita and E. Kato. Iterative control and its application to motion control of robot arm – a direct approach to servo-problems. In *Proceedings of the 24th IEEE Conference on Decision and Control*, pages 1393–1398, Ft. Lauderdale, FL, December 1985.

J. Moon, M. Lee, M. Chung, S. Y. Jung, and D. H. Shin. Track-following control for optical disk drives using an iterative learning scheme. *IEEE Transactions on Consumer Electronics*, 42(2):192–198, May 1996.

S. Oh, Z. Bien, and I. H. Suh. An iterative learning control method with application for the robot manipulator. *IEEE Journal of Robotics and Automation*, 4(5):508–514, October 1988.

Y. Onuki and H. Ishioka. Compensation for repeatable tracking errors in hard drives using discrete-time repetitive controllers. *IEEE/ASME Transactions on Mechatronics*, 6(2):132–136, 2001.

D. H. Owens, N. Amann, E. Rogers, and M. French. Analysis of linear iterative learning control schemes - a 2D systems/repetitive processes approach. *Multidimensional Systems and Signal Processing*, 11(1-2):125–177, 2000.

F. Padieu and R. Su. An H_∞ approach to learning control systems. *International Journal of Adaptive Control*, 4:465–474, 1990.

M. Q. Phan and J. A. Frueh. Learning control for trajectory tracking using basis functions. In *Proceedings of the 35th IEEE Conference on Decision and Control*, pages 2490–2492, Kobe, Japan, December 1996.

M. Q. Phan and J. A. Frueh. Model reference adaptive learning control with basis functions. In *Proceedings of the 38th IEEE Conference on Decision and Control*, pages 251–257, Phoenix, Arizona, USA, December 1999.

A. M. Plotnik and R. W. Longman. Subtitles in the use of zero-phase low-pass filtering and cliff filtering in learning control. *Advances in the Astronautical Sciences*, 103:673–692, 1999.

J. D. Ratcliffe. *Iterative Learning Control Implemented on a Multi-axis System.* PhD thesis, School of Electronics and Computer Science, University of Southampton, 2005.

J. D. Ratcliffe, T. J. Harte, J. J. Hätönen, P. L. Lewin, E. Rogers, and D. H. Owens. Practical implementation of a model inverse optimal iterative learning controller on a gantry robot. In *Proceedings of IFAC Workshop on Adaptive and Learning in Control and Signal Processing and the IFAC Workshop on Periodic Control Systems*, pages 687–692, Yokohama, Japan, 2004.

J. D. Ratcliffe, J. J. Hätönen, P. L. Lewin, E. Rogers, J. Harte, and D. H. Owens. P-type iterative learning control for systems that contain resonance. *International Journal of Adaptive Control and Signal Processing*, 19(10):769–796, 2005a.

J. D. Ratcliffe, J. J. Hätönen, P. L. Lewin, E. Rogers, and D. H. Owens. Robustness analysis of an adjoint optimal iterative learning controller with experimental verification. *International Journal of Robust and Nonlinear Control*, 18(10):1089–1113, July 2008.

J. D. Ratcliffe, P. L. Lewin, and E. Rogers. Fast norm-optimal iterative learning control for industrial applications. In *Proceedings of the 2005 American Control Conference*, pages 1951–1956, Portland, OR, USA, June 2005b.

J. D. Ratcliffe, P. L. Lewin, and E. Rogers. Comparing the performance of two iterative learning controllers with optimal feedback control. In *Proceedings of the 2006 IEEE International Symposium on Intelligent Control*, pages 838–843, Munich, Germany, October 2006a.

J. D. Ratcliffe, P. L. Lewin, E. Rogers, J. J. Hätönen, and D. H. Owens. Norm-optimal iterative learning control applied to gantry robots for automation applications. *IEEE Transactions On Robotics*, 22(6):1303–1307, December 2006b.

R. Roesser. A discrete state-space model for linear image processing. *IEEE Transactions on Automatic Control*, AC-20:1–10, 1975.

E. Rogers, K. Galkowski, and D. H. Owens. *Control Systems Theory and Applications for Linear Repetitive Processes (Lecture Notes in Control and Information Sciences Series)*, volume 349. Springer Verlag, 2007.

D. De Roover. Sysnthesis of a robust iterative learning controller using an h_∞ approach. In *Proceedings of the 35th IEEE Conference on Decision and Control*, pages 3044–3049, Kobe, Japan, December 1996.

D. De Roover and O. H. Bosgra. Dualization of the internal model principle in compensator and observer theory with application to repetitive and learning control. In *Proceedings of the 1997 American Control Conference*, pages 3902–3906, 1997.

D. De Roover and O. H. Bosgra. Synthesis of robust multivariable iterative learning controllers with application to a wafer stage motion system. *International Journal of Control*, 73(10):968–979, 2000.

S. S. Saab. A discrete-time learning control algorithm. In *Proceedings of the American Control Conference*, volume 1, pages 749–753, Baltimore, Maryland, June 1994.

S. S. Saab. A discrete-time learning control algorithm for a class of linear time-invariant systems. *IEEE Transactions On Automatic Control*, 40(6): 1138–1142, June 1995.

S. S. Saab. A discrete-time stochastic learning control algorithm. *IEEE Transactions On Automatic Control*, 46(6):877–887, June 2001a.

S. S. Saab. On a discrete-time stochastic learning control algorithm. *IEEE Transactions On Automatic Control*, 46(8):1333–1335, August 2001b.

S. S. Saab. Stochastic P-type/D-type iterative learning control algorithms. *International Journal of Control*, 76(2):139–148, 2003.

D. D. Siljak and D. M. Stipanovic. Robust stabilisation of nonlinear systems: The LMI approach. *Mathematical Problems in Engineering*, 6:461–493, 2000.

T. Sogo, K. Kinoshita, and N. Adachi. Iterative learning control using adjoint systems for nonlinear non-minimum phase systems. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 3445–3446, Sydney, Australia, December 2000.

M. Steinbuch. Repetitive control for systems with uncertain period-time. *Automatica*, 38(12):2103–2109, 2002.

M. Steinbuch and R. van de Molengraft. Iterative learning control of industrial motion systems. In R. Isermann, editor, *1st IFAC conference on Mechatronic Systems*, pages 967–972, Darmstadt, Germany, 2000.

A. Tayebi and M. B. Zaremba. Robust iterative learning control design is straightforward for uncertain lti systems satisfying the robust performance condition. *IEEE Transactions on Automatic Control*, 48(1):101–106, January 2003.

A. Tayebia and S. Islamb. Adaptive iterative learning control for robot manipulators: Experimental results. *Control Engineering Practice*, 14(7): 843–851, 2006.

M. Togai and O. Yamano. Analysis and design of an optimal learning control scheme for industrial robots: a discrete system approach. In *In Proceedings of the 24th IEEE Conference on Decision and Control*, pages 1399–1404, Ft. Lauderdale, FL, December 1985.

R. L. Tousain and D. van Casteren. Iterative learning control in a mass product : Light on demand in dlp projection systems. In *Proceedings of the 2007 American Control Conference*, pages 5478–5483, New York City, NY, USA, July 11-13 2007.

M. Uchyama. Formulation of high-speed motion pattern of a mechanical arm by trial. *Transactions of the Society for Instrumentation and Control Engineers*, 88(4):569–587, 1978.

D. Wang and C.C. Cheah. An iterative learning scheme for impedance control of robot manipulators. *International Journal of Robotics Research*, 17 (10):1091–1104, October 1998.

Y. Wang and R. W. Longman. Use of non-causal digital signal processing in learning and repetitive control. *Advances in the Astronautical Sciences*, 90(1):649–668, 1996.

M. B. Groot Wassink, O. H. Bosgra, and S. H. Koekebakker. Enhancing inkjet printhead performance by mimo iterative learning control using implementation based basis functions. In *Proceedings of the 2007 American Control Conference*, pages 5472–5477, New York City, NY, USA, July 11-13 2007.

J. Van De Wijdeven and O. H. Bosgra. Residual vibration suppression using hankel iterative learning control. *International Journal of Robust and Nonlinear Control*, 18:1034–1051, 2008.

Z. H. Xiong and J. Zhang. Product quality trajectory tracking in batch processes using iterative learning control based on time-varying perturbation models. *Industrial and Engineering Chemistry Research*, 42(26): 6802–6814, 2003.

J. X. Xu, Y. Q. Chen, T. H. Lee, and S. Yamamoto. Terminal iterative learning control with an application to RTPCVD thickness control. *Automatica*, 35(9):1535–1542, 1999.

J. X. Xu, Q. P. Hu, T. H. Lee, and S. Yamamoto. Iterative learning control with smith time delay compensator for batch processes. *Journal of Process Control*, 11(3):321–328, 2001.

J. X. Xu, T. H. Lee, and Y. Tan. Enhancing trajectory tracking for a class of process control problems using iterative learning. *Engineering Applications of Artificial Intelligence*, 15(1):53–64, 2002.

Y. Zhou, M. Steinbuch, and G. Leenknegt. A cost-effective scheme to improve radial tracking performance for high speed optical disk drives. *Journal of Vibration and Control*, 10:795–810, 2004.