Development of Robotic Test-bed and Algorithm Implementation for Iterative Learning Control

Naveen Parthasarathy

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Master of Engineering

2010

Acknowledgements

I would like to thank my supervisor, Dr. Danwei Wang, for his precious advice, guidance, constant support and encouragement throughout my project. His comprehensive knowledge, insights and suggestions helped me to shape my research skills and carried me on through hard times. It is a true honor to be associated with him.

I would like to take this opportunity to thank my supervisors at Electroglas Inc, Singapore, Mr. Uday Nayak and Mr. Ya Lei Sun for their support and direction. Association with Electroglas Inc. through JIP provided me much needed industrial exposure. It was a great and valuable experience to work with Electroglas team.

I wish to express my appreciation to all the staff members and research students in Intelligent Robotics Lab, NTU especially Mr. Chia and Mr. Neo for always providing help with a smile.

I would like to acknowledge the financial support given by Nanyang Technological University, Electroglas Inc. and the Economic Development board of Singapore.

Finally, I am deeply indebted to my parents and families for their love, care and support throughout my research.

Summary

Robotic manipulators are powerful and complex *mechanical systems* often used in industrial and practical applications to perform intricate and compound operations. A robot system consists of a mechanical manipulator and computer or a microprocessor that controls the motion of the manipulator.

The basic concept of *Iterative Learning Control* (ILC) is based on the fact that most robotic manipulators perform a same task *repeatedly*. By using information from previous repetitions, a suitable control action can be found iteratively. Applications of ILC add more significance to the field of ILC research.

In this project, a *robotic test bed* is developed in order to implement, test and analyze various types of ILC algorithms. It consists of SEIKO D-TRAN TT 3000 SCARA robot, dSPACE Control kit, MATLAB/SIMULINK assembly and Intel PC. It offers stable and highly flexible system platform for real-time implementation of various ILC algorithms.

The Seiko robot has four degrees of freedom with one prismatic and three revolute joints. Characteristics such as non-linearity, coupling gravity and friction in the dynamics are exhibited by Seiko robot. The robotic test bed is equipped with sensors such as optical encoders for joint angle measurements and the custom made interface boards which act as the bridge between different hardware and software interfaces.

Interactive and user friendly robotic test bed system is designed so as to be *flexible* and simple to use. It also provides a platform to implement different types of sophisticated algorithms. System architecture of the robotic test bed offers a good potential for extension in future.

Different types of ILC algorithms are designed and implemented on two of the three major axes of the robot and performance is evaluated. SCARA robot consists of one prismatic joint (Z- axis) and three revolute joints. Three revolute joints are named as joint 2(T1-axis), joint 3(T2-axis) and joint 4(A-axis). Experiments are conducted on T1 and T2 axes. Design and analysis of ILC is performed in frequency domain because it is simple and practical. Some of the ILC algorithms like P-type (Proportional type) and A-type (Anticipatory type) ILC are designed and implemented on the robotic test bed.

Multi-Input Multi-Output (MIMO) systems with *cross-couplings* are considered in this project and suitable ILC algorithms are developed. These interactions make it challenging to control a MIMO system. *Decoupling technique* is developed to compensate for the effect of undesired cross-couplings in MIMO systems. The decoupling technique with ILC is demonstrated in this project.

System identification of continuous time systems using ILC is a potential area of research to acquire accurate plant model. System identification of two different axes of SCARA robot using iterative identification scheme *via* projection is illustrated in this project. Robustness and accuracy of this type of ILC based identification technique is examined.

Table of contents

List of figures

Chapter 1

Introduction

1.1 Background

The main objective of a control system is to *force* output response to follow a desired trajectory. Over the decades, researchers have developed many control strategies for robotic control. Good controllers should be *robust* to system parameter uncertainties and environmental disturbances. Presence of uncertainties and unmodelled dynamics in robotic systems has shown the deficiency of conventional feedback techniques to achieve high tracking accuracy.

Control theories such as adaptive control, robust control, learning control, neural and fuzzy control system, etc have been extensively studied. Iterative learning control is an interesting topic of great potential in robotic control research. ILC offers vital advantages like *robustness* against uncertainties in dynamic parameters, convergence of tracking error along iteration axis, ease of design and implementation, robustness against non linearity in robotic systems, etc.

Most industrial systems and machines perform their tasks in *repetitive* manner. These systems are bound to execute tasks with highest degree of precision and high speeds. Systems are also prone to model variations, disturbances and repeatability imperfections. In industrial applications, higher system efficiency and accuracy will enhance product quality, increase productivity, improve efficiency, earn greater profits and reduce costs.

Performance of systems with higher accuracy and efficiency is highly dependent on the *control* in action. Iterative learning control (ILC) is one such control strategy that aims to *improve* control system performance against the lack in knowledge of system model, disturbances and initial offsets.

Each repetitive operation executed by a system is termed as *iteration or trial*. The basic concept of ILC is that, it uses information such as error data, output data and input data from previous iterations to construct a *feedforward* control input for the next iteration such that output follows a desired trajectory. It aims to reduce the *tracking error* along the *iteration* axis. A typical iterative learning control law [18] is of the form,

$$u_{k+1} = u_k + k_p e_k \tag{1.1}$$

where u_k is the input to system during the k^{th} repetition, e_k is the tracking error during the k^{th} repetition and k_p is a learning gain parameter used in ILC.

ILC schemes can be applied to MIMO systems. Inputs and outputs are *interacting* in the case of MIMO systems. *Decoupling technique* is a technique which eliminates the effect of undesired *cross-couplings* and improves control system performance. In decoupling technique, *decoupling block* is designed by using *simplified* decoupling technique and *generalized* decoupling technique and later integrated to the MIMO system. ILC algorithm is *implemented* on the decoupled system. Generalized decoupling uses a diagonal decoupling block and simplified decoupling uses simpler decoupling block.

ILC based *system identification* approach demonstrated in this project uses projection of continuous-time error, input and output data onto to a finite dimensional sub-space. This approach offers *good accuracy* and *convergence* in learning of identified parameters. System identification can be performed in open loop as well as in closed loop.

1.2 Motivation of research

Robotics is a field of technology which offers *wide range* practical applications for ***iterative learning control*** **(ILC)**. The importance of *experimentation* in ILC study is vital. Some of the theoretical concepts on ILC can be verified by *simulations*. But these simulations don't *guarantee* the performance of ILC in the presence of disturbances, uncertainties and variations of system parameters. ILC algorithms should be tested in *practical situations* and should be able to meet the *industrial* requirements. A robotic test bed is good for implementation of ILC laws.

This robotic test bed should highly *flexible*, easy to operate and also have a provision of *extension* for future development. Robotic test bed should be able to *integrate* other control strategies such as the feedback control. Hence a highly interactive and simple to use robotic test bed is desirable. This objective can be achieved by *structuring* software architecture with different modules, interfaces between modules and/or hardware and an interactive graphical user interface.

Most real-world systems are MIMO systems consisting of multiple *interacting* variables. Due to these *interactions*, designing a controller for MIMO system is a challenge. Techniques to overcome *undesirable* cross-couplings in MIMO system can improve the control system performance. ILC with decoupling technique can be designed and implemented on MIMO systems.

One important aspect in control system design is the *accuracy* of a system model. ILC based *system identification* offers a new way to achieve model accuracy. Hence ILC based system identification is performed on the robotic test bed.

1.3 Literature review

Iterative learning control (ILC) is a *feedforward* control scheme, which uses iterative method to compensate for repetitive errors. Numerous efforts have been made towards defining and analyzing learning control algorithms since Uchiyama [1] presented the concept of ILC for the first time in 1978. In 1984, rigorous formulation and research on ILC was exercised by Arimoto *et al.* [2], Casalino *et al.* [3], Craig [4] and Middleton *et al.* [5]. In [6], term "***iterative learning control***" was first introduced Kawamura *et al.* After their significant contributions in the field ILC research, ILC gained importance and attention.

In model based learning scheme [7], data from estimated system parameters is used to calculate the input corresponding to the desired and actual trajectories. Resulting *input error* is supplied to the learning controller. Arimoto *et al.* [2,6,8] proposed a learning control scheme which uses the *output error* directly. Gu and Loh[9] proposed a ILC scheme where the desired trajectory is fed in the form of generalized momentum and the scheme is termed as multi-step learning control.

ILC can be considered in time-domain, frequency domain and time-frequency domain. In time-domain, *convergence analysis* is performed by applying α-norm or λ-norm [59, 60]. Many researchers have considered learning control in *frequency domain* [10,12,18,24,25,32]. Norrlöf and Gunnarsson presented discussion on time and frequency domain *convergence properties* in ILC in [11]. Goh *et al.* [32] explained the limitations of time domain analysis. Frequency domain analysis of ILC uses every frequency component of the error and aims to drive each frequency component error to decay along iteration axis. Analysis and design in frequency domain is much simpler and practical.

Both time and frequency domain analysis offers their share of advantages and disadvantages in learning. To bridge designs in time and frequency domains, *time-frequency* domain analysis is considered in [22].

Some key issues regarding ILC are the *robustness* in the presence of initial state errors and bad learning transients. The issue regarding presence of *initial state errors* in learning has been in [33, 34, 35, 36] and the developed approach tackles the problem of initial errors effectively. In [19, 23], Wang, Ye and Zhang analyzed cut-off *frequency tuning* of ILC and the use of low pass filters to guarantee good learning behavior. Many schemes of ILC have been developed for continuous-time dynamic systems. Some of the basic schemes are P-type, D-type, PD-type and A-type. Many literatures address the attributes and issues of P-type and D-type ILC schemes [37, 38, 39]. The anticipatory (A-type) ILC scheme was introduced in [40].

A particular type of ILC scheme which widens the *learnable bandwidth* of ILC called the multi-channel type of ILC scheme was developed and implemented in [24, 25]. Bien and Huh [14] proposed a *higher order* learning algorithm which utilizes more historical data storage for a class of nonlinear dynamic systems.

Most real world systems are multi-input and multi-output (MIMO) systems. Tracking accuracy of MIMO systems can also be improved by ILC. Repeated operations of MIMO systems allow control signal to be adjusted between cycles using trajectory error information. In [41], necessary and sufficient condition for the stability of MIMO learning control systems was analyzed.

In [42], Xu and Tan proposed a new ILC scheme called the Newton-type ILC, for nonlinear non-affine dynamic MIMO systems based on functional approximation approaches. They discussed the importance of learning performance indices convergence factor (Q-factor) and convergence order (Q-order) in ILC.

In [43], Tan *et al.* developed a necessary and sufficient condition for convergence of ILC for MIMO linear time-varying systems. In [44], Wang and Ye discussed about two non-causal filtering based ILC approaches for MIMO time-invariant systems. The analysis and design are carried out in frequency domain.

In [45], convergence condition of a feedback-based iterative learning control (ILC) system for an uncertain linear MIMO plant was proposed. Firstly, convergence condition for the known part of the uncertain plant is obtained and later convergence condition for the ILC system including the plant uncertainty is derived and implemented.

In [46], ILC is developed for a class of MIMO systems with less model knowledge. It consisted of a Nussbaum-type gain selector and a refined compensator learned through repetitive tracking.

In [47-48], ILC design for MIMO-LTI systems is discussed. Two particular ILC schemes are considered and analyzed in both frequency and time domains. The first scheme uses only the tracking error from the current iteration, while the second scheme uses the tracking errors from the current and/or the previous iterations. This literature presented discussions regarding convergence, implementation, robustness with respect to reinitialization errors, positive realness issues and disturbances.

In [49], analysis of feedback based ILC approach for MIMO-LTI systems was discussed. This paper shows that ILC combined with feedback controller is more effective to reduce the tracking error than only feedback controller.

System identification using ILC techniques is a potential area for research and development. Numerous methods of system identification techniques like the Least Square method, Predictive error method, recursive method, Instrumental variable method, etc. have been developed. Sugie and Sakai [50] proposed a new ILC scheme, which accomplishes *high-precision* tracking for uncertain linear continuous-time systems in the presence of heavy measurement noise. This approach achieves robustness against measurement noise through projection of continuous-time I/O signals onto a finite dimensional parameter space. The integral operation in this noise tolerant learning law was performed using error data of all previous iterations.

Based on this control method, an effective approach for *identification* of continuous-time systems directly from the sampled I/O data was developed. In [51-52], Campi and Sugie *et al.*, presented a *novel* approach for identification of linear continuous-time systems. The *robustness* against measurement noise is achieved through projection of continuous-time I/O signals onto a finite dimensional parameter space and Kalman filter type noise reduction. The efficiency of this approach was demonstrated on a non-minimum phase plant. In [53], closed-loop identification of linear continuous-time systems directly from the sampled I/O data, based on ILC was discussed. It demonstrated that this type of identification technique performs efficiently without any knowledge of controllers used in the loop. In [56], Kim and Sugie presented an identification approach based on ILC for a class of linear MIMO continuous-time systems with unknown but fixed input disturbances and measurement noise.

1.4 Thesis Contributions

1. A *robotic test bed* is set-up. This test bed can be used to implement several advanced control algorithms and carry out experimental research on robotic systems. Software architecture of the test bed is constructed so that test bed offers high level of flexibility and user-friendly programming environment.
2. *Exhaustive documentation* on the test bed is presented. This accounts for *future extension* and development. Provisions are made in the software set up, in the form of independent modules so that additional piece of code can be developed and implemented.
3. A class of ILC for *trajectory control* is implemented. Two different classes of iterative learning controllers are studied. The advantages of learning controllers to overcome the uncertainties of the model and parameters are displayed.
4. Development of ILC for a class of MIMO systems is exercised. An ILC scheme is developed for MIMO systems in conjunction with *decoupling technique*. Two different schemes of decoupling technique are discussed.
5. *System identification* using ILC is performed on T1 and T2 axes of the SCARA robot. Continuous-time system identification using ILC *via* projection-type approach is demonstrated. Accuracy of the model is examined through model validation.

1.5 Organization of thesis

The remaining part of the thesis is organized as follows,

In chapter 2, the focus is on the *fundamentals of iterative learning control*. The basic concepts, principles, objectives and important issues of iterative learning control are discussed in detail.

In chapter 3, we discuss the hardware, software and system structure of the experimental *robotic test bed.* The construction, working principle and operating principle of test bed are explained in detail.

In chapter 4, the development and implementation of a class of ILC algorithms on robotic test bed are explained in detail. The performance of ILC algorithm is evaluated and compared.

In chapter 5, design and implementation of ILC algorithms for a class of MIMO systems and the use of *decoupling technique* are presented.

In chapter 6, we elucidate continuous-time system identification approach using ILC. *ILC based identification* of T1 and T2 axis of SCARA robot and model validation procedures are discussed in this chapter.

In chapter 7, conclusions and future recommendations of the thesis are presented.

Chapter 2

Fundamentals of iterative learning control

2.1 Iterative learning control: an introduction

ILC is based on the fact that *tracking error* can provide rich information for tuning of input actions and lead to substantial improvements in tracking performance. ILC is designed to exploit *repetitiveness* in systems. Below given are the quotes on ILC by experts in this area.

"Learning Control concept stands for the repeatability of operating a given objective system and the possibility of improving the control input on the basis of previous actual operation data." - **Arimoto *et al.* [65]**

"Learning Control is a name attributed to a class of self-tuning processes whereby the systems performance of a specified task improves, based on the previous performance of identical tasks" - **Heinzinger *et al.* [66]**

"Learning Control is a technique in which the input signal required to achieve a given behavior as the output of a dynamical system is built iteratively from successive experiments." **– de Luca *et al.* [67]**

"Learning control aims to produce zero tracking error during the whole period of a process operation, including transient part using minimum knowledge of system, which is accomplished by using past experience with the same task to improve performance in future operations"- **Danwei Wang [24]**

"Term iterative indicates a kind of action that requires the dynamic process be repeatable, i.e., the dynamic system is deterministic and the tracking control tasks are repeatable over a finite tracking interval. When a control task is performed repeatedly, we gain extra information from a new source: past control input and tracking error profiles, which can be viewed as a kind of experience."- **Jian-Xin Xu [68]**

2.2 Basic idea of iterative learning control

"Errors are repeated when trajectories are repeated". For a time-invariant system with/without feedback control, each time the system repeats a same task, it will produce the same overshoot, rise time, settling time, and steady-state error. ILC has an in-built capability of learning through iterations and is a *feedforward* control technique based on tracking errors. Basic idea is explained in this subsection.

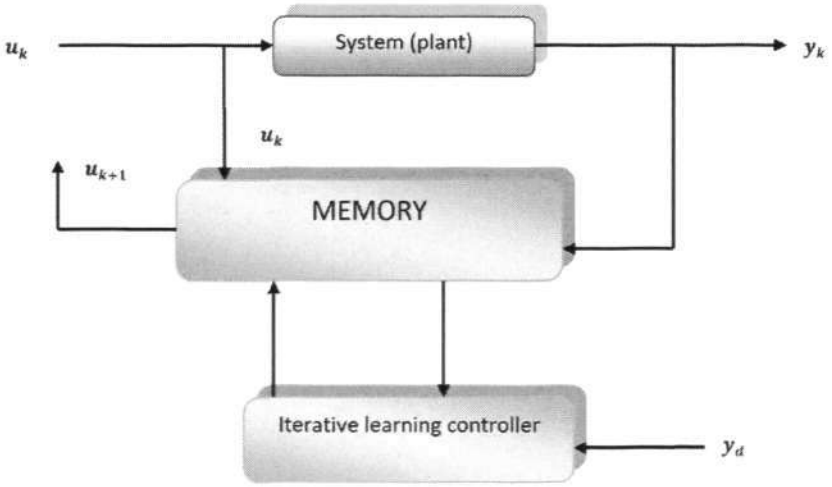


Figure 2.1 : Basic block diagram of ILC

A repetitive action performed by the system is termed as "*trial*" *or* "*repetition*" and is denoted by the subscript k. $\boldsymbol{y_d}$ is the reference trajectory. Figure 2.1 shows the basic diagram of ILC operation. y_k and u_k are the output and *feedforward* input of the system, respectively, in the current iteration. u_{k+1} is the updated input(by learning controller) to the system for the next iteration. Learning controller uses memory unit to store the previous iteration's input and output values. Learning Controller with some *priori* knowledge about the system interacts with the memory unit to calculate updated input signal for the next trial, so that tracking error converges to zero as trials go to infinity. In other words, updated control input in the current trial is designed to produce smaller tracking error than control input signal in the previous trial.

2.3 Phases of ILC

2.3.1 Phase 1: Command Execution

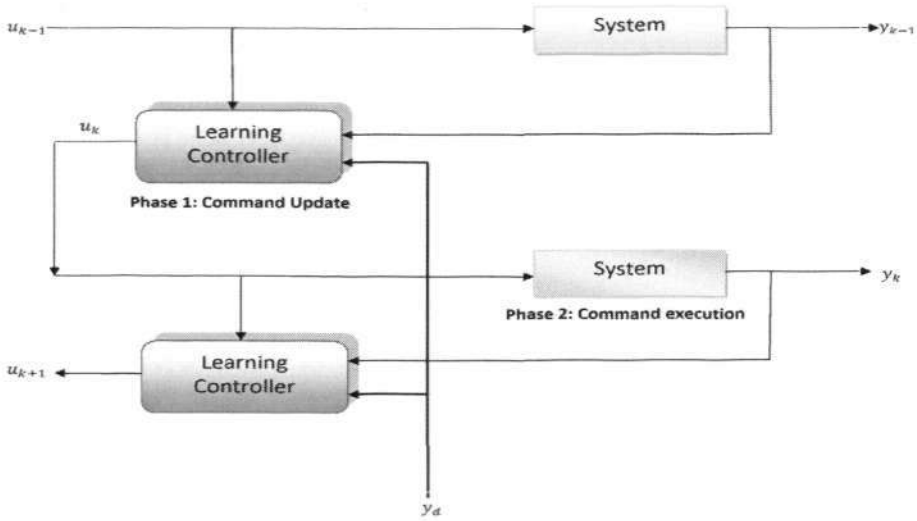


Figure 2.2 : Phases of ILC

where k represents current trial, u_k is control input for current trial, y_k is output of current trial and y_d is desired trajectory. Command execution is *online phase* of ILC. It is basically the real-time implementation of a *feedforward* control input to system.

2.3.2 Phase 2: Command Update

Command Update is *off-line* phase of ILC and calculates control input for the next trial. Figure 2.2 shows the interaction between two phases of ILC. Phase 1 executes current iteration based on *feedforward* control input calculated by the learning controller. During this process, phase 2 updates the control input for the next iteration.

2.4 Objective of ILC

The main objective of ILC design is to calculate recursively,

$$u_{k+1}(t) = L\big(u_k(t), \Delta y_k(t)\big) \tag{2.1}$$

$$\Delta y_k(t) = y_d(t) - y_k(t) \tag{2.2}$$

such that output, $y_k(t) \to y_d(t)$ and $u_{k+1}(t) \to u_d(t)$ as $k \to \infty$

2.5 Attributes of iterative learning control

2.5.1 Design issues

ILC has to be designed to guarantee robust, stable and efficient performance. An ILC controller is often associated with two important issues, namely *performance and stability*. Each system is unique. Systems can be linear or non-linear, MIMO or SISO, dynamic or non-dynamic, time-varying or time-invariant. Some systems possess specific characteristics, such as time delays, uncertainties, arbitrary relative degree, initial error, and so on. Based on the system, a suitable learning law has to be adopted and implemented. Important design issue is to choose an appropriate learning law based on the system. For an effective learning process, the design criterion should be based on efficient derivation of learning parameters, such as learning gain.

2.5.2 Stability issues

Stability is a general and very important property for an ILC control system. ILC based control system is stable, if the system remains unaffected with variation of parameters, presence of known/unknown disturbances and system uncertainties *for all iterations*. Stability of an ILC system can be explained with respect to *tracking error* along the *iteration axis*. ILC system is stable if its tracking error converges to zero or bounded value, as iterations tend to infinity. Stability of ILC based control system can also be explained with reference to BIBO stability, uniform stability and classical stability as follows.

BIBO stability: "*A linear iterative system is BIBO stable if a bounded input,* $\|u\| < \infty$, *generates a bounded output,* $\|y_k\| < \infty$, *for all k", where k is the iteration number.*

BIBO stability [57] for the linear iterative system implies that the corresponding ILC algorithm gives a stable ILC system.

Uniform (Exponential) stability: "*A linear iterative system is uniformly stable if*

$$\|y_k\| \leq \gamma \|y_o\|, \text{ for all } k \tag{2.3}$$

where γ *is positive constant". Uniform exponential stability* [57] *is achieved if*

$$\|y_k\| \leq \gamma\, \lambda^k \|y_o\| \text{ where } k>0 \text{ and for } 0 < \lambda \leq 1 \tag{2.4}$$

Classical stability of ILC in frequency domain:

The system in G(s) controlled using the ILC algorithm is stable if

$$\|1 - L(j\omega)G(j\omega)\| < 1 \tag{2.5}$$

where L(s) is the learning gain matrix"

2.5.3 Convergence

The basic purpose of an ILC algorithm is to *reduce tracking error* as iteration grows. This implies that, as iterations increase, tracking error has to decrease upon implementation of a good ILC algorithm. Convergence is an essential property of ILC which guarantees *stability of the system* along iteration axis. The rate of convergence along the iteration axis decides the performance of an ILC based control system. The convergence is broadly classified as *monotonic convergence* and *asymptotic convergence*. Monotonic convergence is convergence condition that determines whether or not ILC converges as the iterations go to infinity. Asymptotic convergence is related to the decrease in error in the sense of some norm as the iterations increases. From appendix A, we can establish that *asymptotic stability* in repetition domain guarantees tracking error *convergence* in time-domain. Convergence is also defined based on the desired and output trajectory in many ways as stated below

$$\lim_{k\to\infty} sup\, \|y_d(t) - y_k(t)\| = 0 \text{ (Asymptotic convergence)} \tag{2.6}$$

$$\lim_{k\to\infty} sup\, \|y_d(t) - y_k(t)\| = \varepsilon \text{ (ε-convergence)} \tag{2.7}$$

$$\forall \alpha, \beta, k > 0\,, \|y_d(t) - y_k(t)\| \leq \alpha \|y_d(t) - y_0(t)\| e^{-\beta k} \text{ (Exponential convergence)}$$

Convergence condition can also defined based on state space parameters of system as

$$\|I - CBL\| < 1 \tag{2.8}$$

where C and B are matrices from state domain representation of system, L is learning gain matrix. The convergence condition does not take plant matrix A into account. This condition does not require complete knowledge about the plant in design of ILC. But this can be a demerit since *priori* knowledge about the system can be extremely useful in effective ILC design. The convergence analysis can be carried out using norm analysis such as α-norm, λ-norm, sup norm, H_∞ norm or H_2 norm. Newton's and secant's [42] methods are developed for fast convergence.

2.5.4 Robustness

Robustness is an important issue of ILC that deals with *uncertainties*. Convergence of ILC is proved based on the nominal mathematical model of plant. Since the plant model mostly has model uncertainties such as modeling errors, dynamic variation of parameters and unstructured disturbances, ILC designed may defy the convergence conditions. A robust system is *susceptible* to sacrifice system performance and stability. Hence, a robust ILC has to be designed with a *trade-off* between performance and robustness.

2.5.5 Learnable band

The range of frequencies that the convergence condition holds good is defined as *learnable frequency band*. Wider learnable frequency band ensures better tracking performance. Error components outside this frequency range should not be used in learning and has to be cut-off to ensure convergence of tracking error. If the learning frequency bandwidth is not wide enough and the dynamic system requires higher bandwidth, Multi-channel ILC's [24, 25] can be used to widen the learnable bandwidth.

2.5.6 Learning transients

The learning process in ILC is theoretically guaranteed by convergence and robustness conditions. But due to the presence of uncertainties and different type's noise components, ILC may exhibit bad learning behavior. In simpler words, ILC shows a divergence in learning and this property is termed *bad learning transient*. It can be rectified by *cut-off frequency tuning* of ILC [19]. The basic objective of frequency tuning is to find the actual learnable bandwidth of system and to maximize effective performance of ILC over the learnable bandwidth. This can be achieved by using simple low pass filters in convergence and robustness conditions [23].

2.5.7 Order of ILC

First order ILC:

A first-order ILC algorithm is one that uses measurements from only the previous iteration

$$u_{k+1}(t) = u_k(t) + L(.)e_k(t) \tag{2.9}$$

$$e_k(t) = y_d(t) - y_k(t)$$

Higher order ILC:

The update law of *high-order ILC* uses error signals from more than one previous iteration. A general linear, time-invariant high-order ILC is illustrated to be

$$u_{k+1}(t) = \sum_{j=k-N+1}^{k} \left(Q_{k-j+1}(.)\left(u_j(t) + L_{k-j+1}(.)e_j(t)\right)\right) \tag{2.10}$$

where $Q_j(.)$ and $L_j(.)$ represent linear transfer operators. The control signal for next trial is calculated from control signals and errors in N previous iterations. This represents an Nth-order ILC algorithm.

2.6 Types of ILC

ILC is classified based on the learning law used in update of control input.

2.6.1 P-type ILC

The basic form of P-type ILC [18, 58] is given below. It uses tracking error from previous iteration to calculate the control input of next iteration. The update and error equations are defined as

$$u_{k+1}(t) = u_k(t) + \varphi\big(y_d(t) - y_k(t)\big) \tag{2.11}$$

$$e_k(t) = y_d(t) - y_k(t)$$

where $u_k(t), y_d(t), y_k(t)$ are control input, desired output and measured output of the current iteration, respectively; φ is the learning gain and k is the iteration number.

General convergence of P-type for linear time invariant system is given by the norm,

$$\|e_{k+1}(t)\| < \rho\|e_k\| \qquad 0 \le \rho < 1 \tag{2.12}$$

Therefore, $\|e_k\| < \rho^k\|e_0\| \to 0 \quad as\ k \to \infty$

The convergence condition can also be derived in frequency domain as follows,

From Eq 2.11, $$U_{k+1}(s) = U_k(s) + \varphi E_k(s) \tag{2.13}$$

System in Laplace domain, $$Y_k(s) = G_p(s)U_k(s) \tag{2.14}$$

$$Y_{k+1}(s) - Y_k(s) = G_p(s)\big(U_{k+1}(s) - U_k(s)\big) \tag{2.15}$$

From Eq. 2.13, $$Y_{k+1}(s) - Y_k(s) = G_p(s)\varphi E_k(s) \tag{2.16}$$

Since, $$Y_{k+1}(s) - Y_k(s) = -(E_{k+1}(s) - E_k(s)) \tag{2.17}$$

We deduce that, $$E_{k+1}(s) = \Big(1 - \varphi G_p(s)\Big)E_k(s) \tag{2.18}$$

In order to ensure convergence in frequency domain,

$$\big|1 - \varphi G_p(j\omega)\big| < 1 \tag{2.19}$$

P-type ILC is the simplest type of ILC and does not require *highest order* derivative signals from system dynamics. This eliminates the use of derivatives in system and is less noisy. P-type law is not a *causal pair* of input action and output produced. P-type learning control as mentioned above will work only in the absence of uncertainties and it is not robust. In the presence of initial errors the law may result in absurd calculations. The concept of forgetting factor was thus introduced to provide robustness to P-type ILC. Modified update equation with forgetting factor is given as,

$$u_{k+1}(t) = (1-\gamma)u_k(t) + \gamma u_0(\mathrm{t}) + \phi\big(y_d(t) - y_k(t)\big) \tag{2.20}$$

γ is forgetting factor and $\gamma u_0(\mathrm{t})$ is bias term used to keep input within the bounds. To ensure that ILC law is robust, law is designed with taking into account tracking error bounds and strict assumptions.

2.6.2 D-type ILC

D-type ILC [2, 18] uses differential of tracking error from previous iteration to calculate control input of next iteration.

$$u_{k+1} = u_k + L\big(\dot{y}_d(t) - \dot{y}_k(t)\big) \tag{2.21}$$

$$e_k(t) = y_d(t) - y_k(t)$$

where $u_k(t), y_d(t), y_k(t)$ are control input, desired output and measured output of the current iteration, respectively; L is the learning gain and k is the iteration number. General convergence of D-type law is governed by the equation

$$\|\dot{e}_{k+1}(t)\| < \rho\|\dot{e}_k\| \qquad 0 \le \rho < 1 \tag{2.22}$$

Hence, $$\dot{e}_k(t) \to 0 \; as \; k \to \infty \tag{2.23}$$

and by integration, $$e_k(t) \to 0 \;\; as \; k \to \infty \tag{2.24}$$

Convergence condition in frequency domain is derived in the same way as derived for the P-type learning law. D-type convergence condition is given as,

$$\left|1 - (j\omega)\varphi G_p(j\omega)\right| < 1 \tag{2.25}$$

D-type law is a *causal pair* of input action and the output produced. Hence input action applied and the output produced take place at same time *t*. The disadvantage of D-type control law is that it requires *highest order derivatives* of the system. Most of the systems have only position measurements. Since the highest order derivatives are usually not measureable, numerical differentiation of position signals is inevitable and this process is very *noisy*. High noise levels reduce performance of the system. Use of carefully designed low pass filters can reduce the noise to some extent.

2.6.3 PD-type ILC

PD-type updating law is a combination of P-type and D-type.

$$u_{k+1}(t) = u_k(t) + \Gamma\dot{e}_k(t) + Re_k(t) \tag{2.26}$$

where (2.27)

$$\dot{e}_k(t) = \dot{y}_d(t) - \dot{y}_k(t) \tag{2.28}$$

$$e_k(t) = y_d(t) - y_k(t) \tag{2.29}$$

where $u_k(t), y_d(t), y_k(t)$ are control input, desired output and measured output of the current trial, respectively; Γ and R are the learning gains and *k* is the iteration no.

Convergence of PD-type of learning law can be proved to be

$$\left|1 - \Gamma.j\omega.G_p(j\omega) + RG_p(j\omega)\right| < 1 \tag{2.30}$$

PD type of learning law is a combination of P-type and D-type laws. It compensates for the disadvantages of the P-type learning law as the causal pair of input action and output are used. It has a disadvantage of additional noises due to the differential of error used.

2.6.4 A-type ILC

Anticipatory (A-type) ILC [18,27,40] is an update law with *linear phase* compensation. It uses a *time shift* in output errors, which incorporates anticipatory characteristics of the updating law. A-type scheme is based on the fact that an input *u(t)* at time *t* to a dynamic system is causally paired with its output *y*(*t*+Δ) at time (*t*+Δ). Here, Δ is the time shift introduced in output.

The learning law of A-type ILC is given as

$$u_{k+1} = u_k + L\ e(t+\Delta) \tag{2.31}$$

$$e(t+\Delta) = y_d(t+\Delta) - y_k(t+\Delta)$$

where $u_k(t), y_d(t), y_k(t)$ are control input, desired output and measured output of the current trial, respectively; L is learning gain, Δ is lead time and k is the iteration no.

A-type ILC has an anticipative nature and does not require *differentiation* of signals like error or output. Hence *low noise levels* are achieved by this control law. Tracking error convergence results can be established in the time domain and frequency domain under the presence of uncertainties, disturbances, and measurement noise.

Convergence of A-type ILC can be derived in frequency domain to be

$$\left|1 - e^{j\Delta\omega} L(j\omega) G_p(j\omega)\right| < 1 \tag{2.32}$$

$G_p(s)$ is the plant transfer function.

A-type can capture the direction and trend data from recorded errors. It also avoids the use of *highest order derivatives* of dynamical system and therefore achieves *low noise levels*. A-type ILC system is easy for design and simple in implementation. Hence, A-type ILC can compensate for disadvantages of P-type ILC and D-type ILC.

2.7 Applications of ILC

ILC is used for systems which execute same operations again and again. Hence ILC has a lot of potential applications in industries, robotics, artificial intelligence and other research areas. Some of the industrial applications of ILC are multi-axis chain conveyor system, pnuematic actuated X-Y Table, hard disc drive industry, CNC machine tools, batch processing, Wafer test probers, gas-metal arc welding and signal processing.

2.8 Advantages of ILC

- Great *potential* for systems with naturally repetitive action where transfer of data between iterations can lead to substantial improvements in tracking performance.
- ILC achieves *perfect* tracking by iteratively updating input from trial to trail. Actual error measured during iteration is used instead of model data, because of which finally calculated *feedforward* input results in better tracking accuracy.
- This method isn't limited by any *model boundaries* whatsoever because the feed-forward input is stored in a time-indexed vector.
- ILC does not require an *accurate model* of the system in order to work.
- The ILC design can be paired with most *stabilizing controllers*
- The tracking error from previous iterations is used as the correction factor for the next control action. Hence better *convergence*
- ILC provides *robustness* to system. The convergence of output error by the ILC is guaranteed even in presence of modeling errors, unknown external disturbances and measurement noise.
- The high-order ILC scheme may improve transient characteristics of system

Chapter 3

Development of robotic test bed

This chapter presents detailed discussion on the *system architecture* and *hardware* platform used in construction and design of robotic test bed. It describes the system configuration set-up, development environment using MATLAB /SIMULINK, Real time workshop (RTW), dSPACE controller and interface libraries.

Figure 3.1 illustrates the block diagram of the robotic test bed.

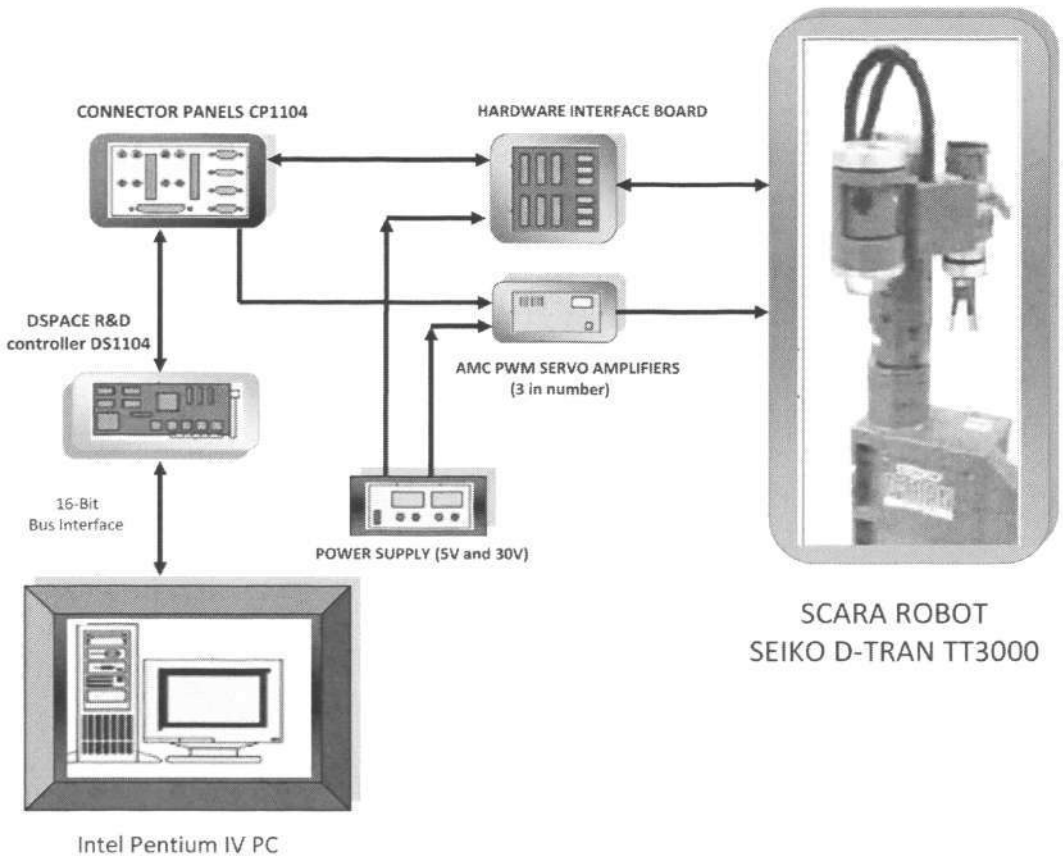


Figure 3.1 : Robotic test bed

DS1104 DSP Controller Board, which is a PC *add-in* board, is housed in a 16-bit ISA slot on PC motherboard. The Connector Panel CP1104 provides connections between the DS1104 DSP Controller Board and interface board. The hardware interface board acts as an interface between connector panel CP1104 and PWM power amplifier drivers. It includes connections for control signals for the robot's axes, encoder output signal and power supply. Three separate PWM (Pulse Width Modulation) DC servomotor amplifiers are used to control the three major axes of the robot. There is also external power supply which supplies the 30V DC voltage required by PWM DC servomotor amplifier and 5V DC voltage required by hardware interface board.

3.1 Components of Robotic Test bed

3.1.1 SCARA Robot SEIKO DTRAN TT-3000

3.1.1 a Description

The SEIKO D-TRAN 3000 Series robot is a 4-axis, closed-loop DC servo SCARA (Selectively Compliance Assembly Robot Arm) manipulator. Each of the 4 axes provides a different motion and contributes to one degree of freedom of the robot arm (figure 3.2).The main characteristics of the TT-3000 SCARA robot are its high precision, repeatability and speed.

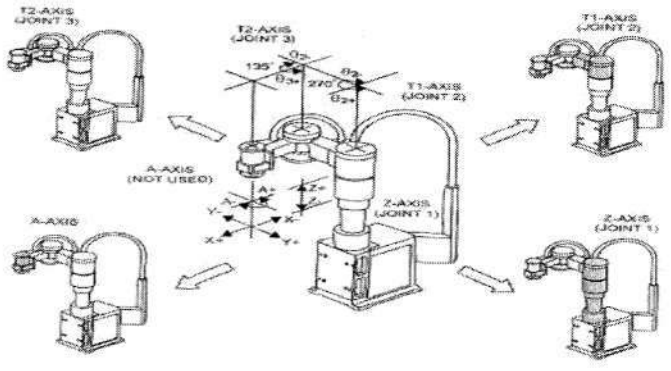


Figure 3.2 : SCARA robot Seiko Dtran TT3000

SCARA robot consists of three revolute joints (T1, T2 and A-Axis) and one prismatic joint (Z-Axis) in such a way that all axes of motion are parallel (figure 3.3). The SCARA ROBOT offers stiffness to *vertical loads* and compliance to *horizontal loads*. SCARA robot is an effective manipulator used in high speeds and high precision because gravitational load, coriolis and centrifugal forces do not stress the structure as much as they would, if the axes were horizontal. Third revolute joint (A-Axis) of the TT-3000 at end-effector provides addition *flexibility* to manipulator task programming. Figure 3.4 shows the joint motion of the robot, when T1 and T2 axes move simultaneously.

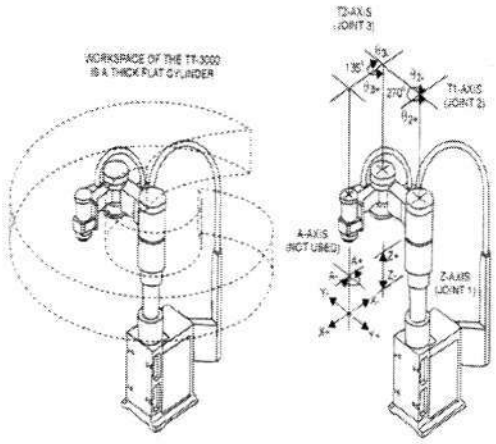


Figure 3.3 : Robot work envelope

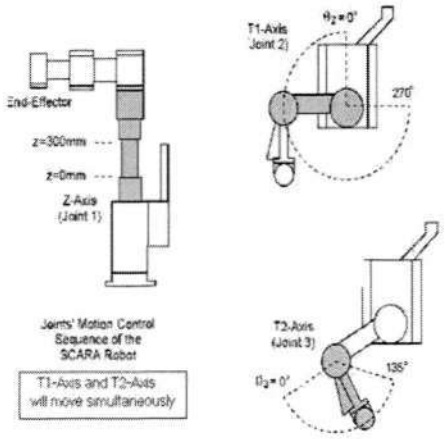


Figure 3.4 : Robot geometry

3.1.1 b Robot Specifications

Property	Details
Repeatability	+/- 0.050mm
Speed (Vertical Stroke)	300 mm/second
Combined Max. Speed	2240 mm/second
Payload	2.5kg
Power	0.5 KVA
Voltage	Single Phase 240 VAC 50 Hz

3.1.1 c The Robot dynamic model:

The dynamics of a robot arm with *n* joints is governed by the differential equation:

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) + F_v(\dot{\theta}) + F_c = \tau \qquad (3.1)$$

where $M(\theta)$ is n×n inertia matrix of the manipulator, $C(\theta,\dot{\theta})$ is n×n matrix of centrifugal and Coriolis terms, $G(\theta)$ is n×1 vector of gravity terms, $F_v(\dot{\theta})$ is n×1 vector of viscous friction terms, F_c is n×1 vector of coulomb terms and τ is n×1 vector of input torque (generated by the joint motor). The terms $\theta, \dot{\theta}$ *and* $\ddot{\theta}$ are nx1 vectors of output link position, velocity and acceleration, respectively.

3.1.2 PC interface

The PC interface is a Intel(R) Core(TM) 2 CPU 6300, 1.86 GHz, 1 GB RAM processor system, operating with Windows XP operating system. MATLAB version 7.1 and dSPACE version 3.1 are installed. This interface provides an *Integrated Development Environment* (IDE) for MATLAB/SIMULINK and dSPACE controller.

3.1.3 DS1104 R & D Controller board

DS1104 R&D Controller Board [71], which is manufactured by dSPACE Inc., Germany, is a real-time hardware based on PowerPC technology and its set of I/O interfaces makes the board an ideal solution for developing controllers in various industrial applications. DS1104 Controller board is plugged into PCI (Peripheral Component Interconnect) bus slot of PC. The DS1104 is specifically designed for the development of high speed multi-variable digital controllers and real-time simulations. For advanced purposes, the board includes a slave DSP processor TMS320F240 (DSP microcontroller). Slave DSP processor can be used for applications such as PWM signal generation, serial peripheral communication and development of RS232 interface. Figure 3.5 illustrates the block diagram of DS1104 R&D Controller board.

The ***hardware package*** contains

- One PCI-slot board with a bracket including a 100-pin I/O connector
- One adapter cable with 50-pin sub-D connectors.
- CP1104 connector panel with adapter cable to I/O board

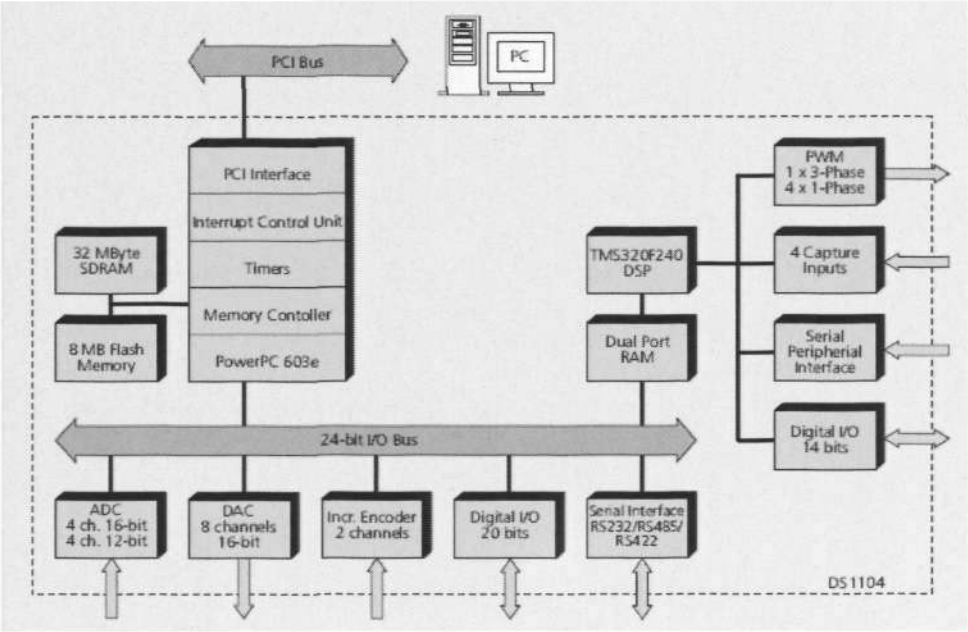


Figure 3.5 : Block diagram of R&D controller board DS1104 [71]

dSPACE **software package** includes implementation and experimentation software such as the RTI, ControlDesk, python, instrument panels and CAN.

3.1.4 PWM Servo Amplifiers

The power amplifiers used in robotic test bed are 25A8 series amplifiers manufactured by *Advanced Motion Controls*. The 25A Series PWM servo amplifiers are designed to drive brush type DC motors at a high switching frequency. It is fully protected against over-voltage, over-current, over-heating and short-circuits across motor, ground and power leads. This model of servo amplifiers can be interfaced with digital controllers or can be used as stand-alone drives. The robotic test bed houses 3 PWM amplifiers, which act as drives for three major axes motors (T1, T2 and Z axes) of robot.

3.1.5 Connector panel

Connector panel (CLP 1104 or CP1104) is used for establishing *connections* between the controller board (DS1104) and peripheral devices. Connector panel has connections for ADC and DAC I/O, Digital I/O, Slave I/O PWM Connector, Incremental encoder interface connectors and UART RS232 connectors.

3.1.6 Interface board

Interface board is a *custom made* board which acts as an interface between drive components (PWM Amplifiers) and controller. Interface board was originally designed to suit the DSP board DS1102. But many modules of interface board are not compatible for DS1104. An up-graded interface board was designed. Figure 3.6 shows hardware architecture of upgraded interface board.

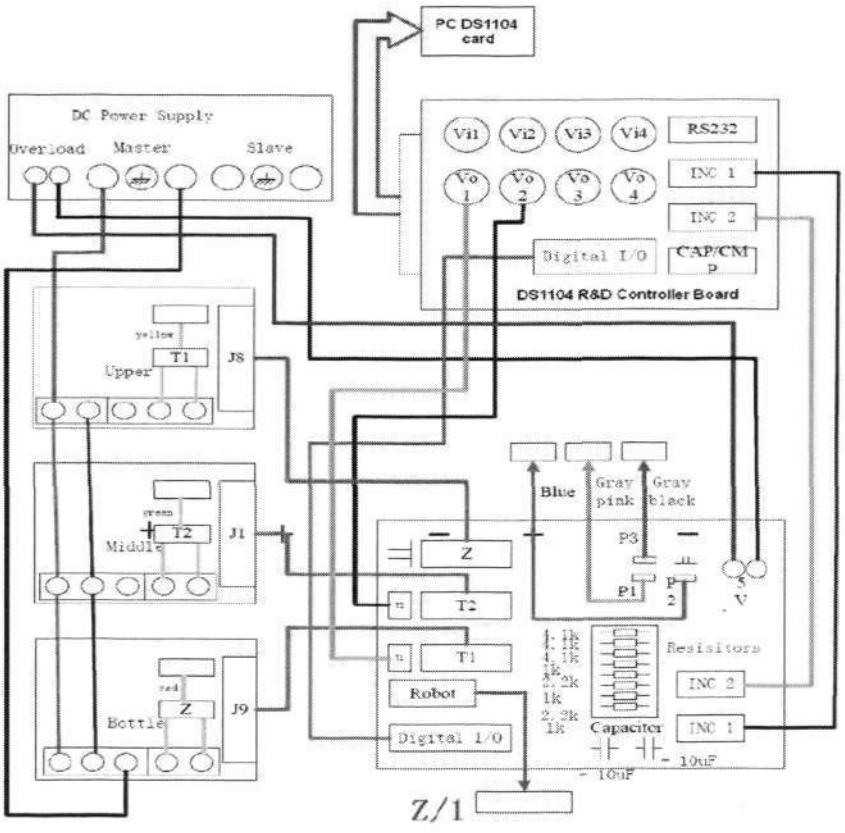


Figure 3.6 : Interface Board

3.1.7 Software architecture

The software architecture governing robotic test bed is *hierarchical* in nature. It consists of two hierarchies', namely *low level* and *high level*. ILC focuses on control input update algorithms and differs from most existing control methods in the sense that it exploits every possibility to incorporate past control information, such as tracking errors and control input into construction of present control input. The software architecture is developed such that test bed meets all the requirements of an efficient ILC based control system. ILC has two different phases, namely *command update* and *command execution*. Command execution is performed by low level hierarchy and command update is performed by high level hierarchy. Hence hierarchical architecture successfully isolates two phases offering higher flexibility in testing ILC laws. Figure 3.7 describes hierarchies that constitute the software system.

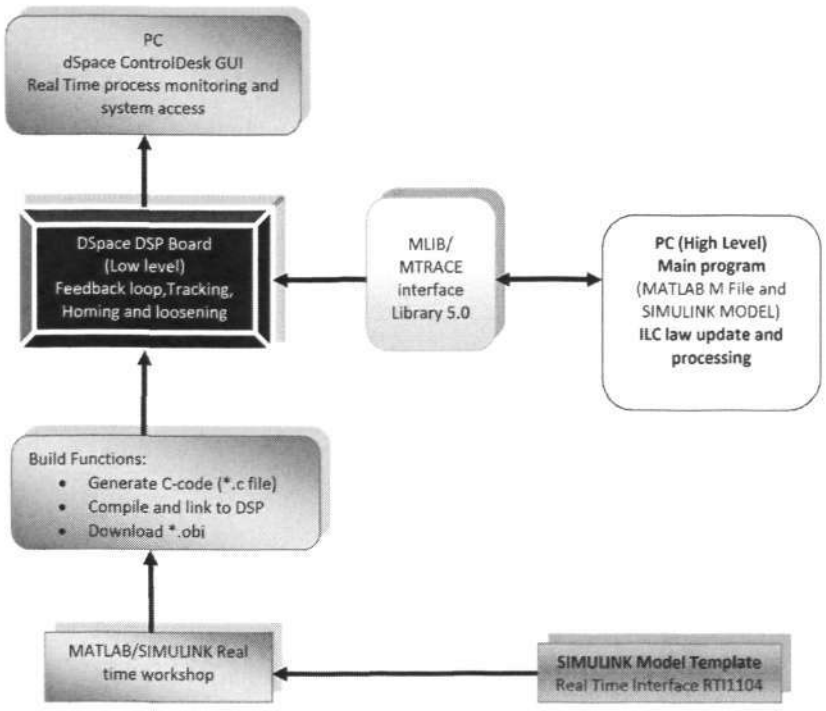


Figure 3.7 : Software architecture of robotic test bed

3.1.7a Low level hierarchy

Programs that reside on DSP Board signify *low level* hierarchy of software architecture. MATLAB Real Time Workshop compiles SIMULINK template and generates a *model_name.obj* file, which is downloaded on to the DSP board using Real Time Workshop. Hence low level software hierarchy is created by SIMULINK and MATLAB Real-Time Workshop.

3.1.7b High level hierarchy

ILC algorithms and error processing sub-routines signify *high level* hierarchy of software architecture. ILC algorithm calculates control input for next iteration based on outputs of previous and/or present iteration. These programs dynamically influence low level module by transferring data such as system parameters and command signals to DSP board through MLIB/TRACE interface library. High level module is offline in test bed. Basically, MATLAB *.m files are used here to develop high level hierarchy.

3.1.8 System upgrade

In this section, we discuss the system *up-gradation* from dSPACE DS1102 to DS1104. The upgrade constitutes hardware and software enhancements. The existing system consists of dSPACE board DS1104 and MATLAB 7 interface. The system was upgraded from dSPACE board DS1102/MATLAB 3 interface. The following table shows the comparison.

Property	Old System	Up-graded system
DSP Board	DS1102	DS1104
No. of Pins	62	100
MATLAB compatibility	Version 5 or lower	Version 6 or higher
dSPACE version	3.0	5.1
No. of DAC's and ADC's	4	8
Incremental encoders	2	2
Connector panel	CP1102	CP1104
Power amplifier	AMC 20A14E	AMC25A8M
DAC range	+5 V/-5 V	+10 V/-10 V

3.2 Hardware and Software configuration

3.2.1 Hardware Configuration

Hardware	Details
PC	Intel(R) Core(TM)2 CPU 6300 ,1.86 GHz ,1 GB RAM
Controller board	DS 1104 DSP R&D Controller Board
Hardware driver	Pulse Width Modulation (PWM) DC Servomotor Amplifiers(AMC 25A8)
Plant(Robot)	Seiko TT-3000 SCARA Robot
Power supply	DC Power Supply (30V & 5V)

3.2.2 Software Configuration

Software	Details
Operating System	Windows NT or Windows XP
Texas Instruments C Compiler	Version 4.70, 32-bit executables
MATLAB	Version 7.3.0.267(R2006B)
SIMULINK	Version 6.5
Real-Time Workshop	Version 6.5
dSPACE Real-Time Kernel	RTK
dSPACE ControlDesk	Version 3.1
dSPACE Real-time Interface	Version 5.5
dSPACE MLIB/MTRACE	Version 3.1.1

3.3 Software Development for Robotic test bed

The software development of robotic the test bed is based on *hierarchical* software architecture. Software development procedure is illustrated in detail in figure 3.8.

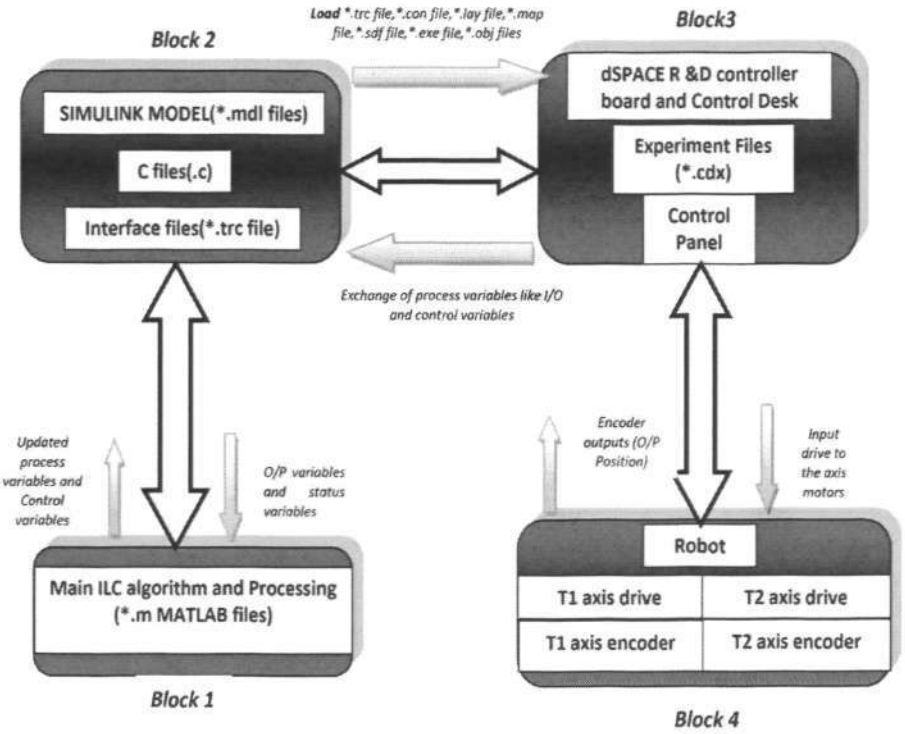


Figure 3.8 : Software development process

Software interface (figure 3.8) is divided into four blocks.

Block 1: consists of main ILC algorithms and data processing algorithms in the form of MATLAB M-files. It is used for update of control input variables.

Block 2: consists of SIMULINK template, MATLAB files and C Files. It forms the backbone of robotic test bed.

Block 3: provides dSPACE Interface.

Block 4: consists of drives and feedback encoder interface.

Files and data that are exchanged between these blocks are depicted in figure 3.8.

3.3.1 SIMULINK Interface

SIMULINK interface provides tools for system modeling, simulation, and validation. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or hybrid of the two. Multi-rate systems, i.e., systems having different parts that are sampled or updated at different rates can also be realized in SIMULINK. SIMULINK has an Integrated Development Environment (IDE) for building models as block diagrams. This is very efficient and flexible, when compared to previous simulation packages that require you to formulate differential equations and difference equations in a language or program. SIMULINK has collection of comprehensive library, which includes built-in tool box, sources, sinks, connectors, switches, etc. Third party interface libraries can also be added to SIMULINK library. Physical Interpretation of SIMULINK blocks are realized in the form of S-Functions. Files associated with SIMULINK interface are [71],

File type	Description	Created by	Read/write access
MDL File	These are structured ASCII files that contain keywords and parameter value pairs describing a SIMULINK model. It is for non-real time simulation performed on the SIMULINK platform	SIMULINK/RTI Interface manually	Read/write access by SIMULINK Read/write access by ControlDesk
OBJ/PPC file	The object file is the executable file for the real-time processor on the slave DSP boards and PPC file is used for master devices.	SIMULINK/RTI Interface automatically when the model is built.	Write access by SIMULINK/RTI interface
Template Makefile	The template makes files that control the compiling and linking process.	SIMULINK/RTI Interface	Read access by SIMULINK RTI
Map file (MAP)	Map files are generated by the linker and map symbolic names to physical addresses. The MAP file must be located in the same directory as the TRC file. If this is not the case, opening the TRC file generates an error message in the LOG file	SIMULINK/RTI Interface	Read access by ControlDesk and SIMULINK RTI

3.3.2 MATLAB Interface

MATLAB interface provides design, analysis, and data visualization tools. MATLAB version used here is MATLAB 7.3. Robotic test bed has two hierarchies in its software architecture. The high level hierarchy is *fashioned* by MATLAB Interface. MATLAB M-files with the extension *.m are used for this purpose. Main ILC algorithm is implemented in a MATLAB interface. It contains procedures for,

- Generating and choosing desired trajectory.
- Choosing the operating time
- Choosing the ILC law
- Encoder reset and runtime calculation.
- Calculation of command input for the next iteration of ILC.
- Calculation of RMS errors and procedures for display of results

3.3.3 Software inter-communication interface files

Variable descriptor File (TRC)

Interface files make user-defined variables in MATLAB/S-functions to be accessible by dSPACE controller. TRC files provide information on available variables and how they are grouped. Interface files are *variable descriptor files* which communicate between MATLAB/SIMULINK and dSPACE. It consists of all variable declarations of all user defined variables. Declarations formats are compatible and usable by upgraded versions of dSPACE/MATLAB. RTI and RTI-MP complies SIMULINK variables in *model_name.trc* file. For both single-processor and multiprocessor systems, system descriptor file is used to access variables during real-time implementation *via ControlDesk*. User defined variables can also used by writing a user TRC file (*model_usr.trc)* in C or notepad and storing it in same working directory. When model is compiled, user defined custom TRC file will also be added to process.

Target Language Compiler File (TLC) interface

Real-Time Workshop uses Target Language Compiler files to translate SIMULINK model into executable code. Target Language Compiler uses two types of TLC files during *code generation* and *build process*. System target file, which describes how to generate code for a chosen target, is the entry point for TLC program that creates executable file. Block target files define how code looks for each of the SIMULINK blocks. System and block target files will be named as *model_name.tlc*.

RTW file

Model description is saved in ASCII file called *model.rtw*. RTW will read *model.rtw* and execute Target Language Compiler (TLC) program that consists of a set of target files (*rti1104.tlc*). The *.tlc files are automatically created when SIMULINK model is built. User can create *.trc file, while using variables from another program or device.

3.3.4 dSPACE Environment Interface

Graphical User Interface (GUI) for test bed is designed in dSPACE *ControlDesk*.

Structure of ControlDesk GUI

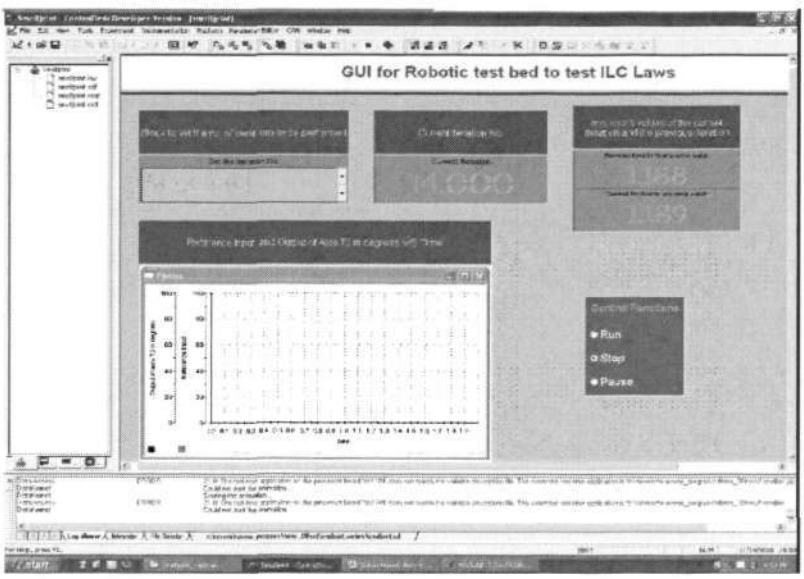


Figure 3.9 : ControlDesk GUI

ControlDesk (figure 3.9) provides a resourceful GUI to analyze and supervise real time data like joint trajectory, iteration number, rms(root mean square) error, initial position error and other process variables. Parameters can also be updated and changed on the fly in dSPACE *ControlDesk*. This saves considerable time in rebuilding of code. Software files associated with *ControlDesk* interface are listed below [71].

File type	Description	Created by	Access by
System Descriptor file (SDF)	Sdf file describe the files to be loaded to the individual components of a simulation platform. They are generated automatically when the TRC file is built.	System Descriptor File Editor and Simulink/Real-Time interface	Read/write access by ControlDesk Write access by Simulink/Real-Time Interface
ControlDesk experiment file (CDX)	It contains the links to all files related to an experiment	Experiment Manager in ControlDesk	Read/write access by ControlDesk.
Connection file (CON)	ControlDesk connection files contain information on the data connections between instruments and the real-time variables.	Instrument Navigator in ControlDesk	Read/write access by ControlDesk
Layout file (LAY)	Layout files contain information like instruments, size , position attributes of front end. It does not usually contain the connections between variables and instruments	Experiment Manager in ControlDesk	Read/write access by ControlDesk Read/write access by MATLAB and RTI
Matlab binary file (MAT)	MATLAB binary files are used for saving reference data captures or table editor data.	Experiment Manager in ControlDesk	Read/write access by ControlDesk Read/write access by MATLAB and RTI
dSPACE.ini	The dSPACE.ini file describes the configuration of the real-time hardware already registered	dSPACE ControlDesk	Read access by ControlDesk and RTI
dSPACE.log	The dSPACE.log file is ControlDesk's default LOG file	ControlDesk	Read/write access by ControlDesk
PAR file	Parameter files contain descriptions ,types and values of parameter sets	Variable Browser	Read/write access by ControlDesk

MATLAB-dSPACE Interface Libraries (MLIB/MTRACE)

MATLAB-DSPACE interface libraries allow MATLAB M-files to access dSPACE real-time processor. It provides functions for read/write tasks in DS1104 board, data capture and status monitoring. It can be used to investigate real-time data, implement various control system interfaces and process optimization.

3.3.5 C File interface: (User defined S-Functions)

ILC implementation consists of calculation of updated control input and applying calculated control input to the system during next iteration. User defined S-functions coded in C, serve as *store houses* to store program data and control data, such as control input, desired trajectory, output and error. It works in conjunction with MATLAB interface library "*mlib/mtrace*". The purpose of MATLAB interface libraries are explained in section 3.3.4. These programs are integrated and downloaded to DS1104 board when the SIMULINK model is compiled.

3.3.6 Real Time workshop Interface(RTW)

Real-Time Workshop generates C program (*model_name.c*) from SIMULINK model. RTW interacts with interface libraries, SIMULINK model, S-Functions and MATLAB functions in order to generate a C code. Generated C code is compatible with standards of DS1104 processor. By using target language compilers and variable descriptor files an executable code is generated and downloaded on to DS1104 controller. RTW interface provides framework for running generated code in real-time environment, tuning control parameters and viewing real-time data.

The RTW consists of following characteristics,

- Embedded Real-Time control codes for real-time controllers or digital signal processors can be generated, cross-compiled, linked, and downloaded onto selected target processors.
- Control, signal processing and dynamic system algorithms can be implemented by developing graphical SIMULINK block diagrams and automatically generating C codes.
- Executable codes can be created for real-time simulation of systems.

3.4 Development and operating procedure of robotic test bed

In this section, development of *software interface*, which includes building SIMULINK models, coding MATLAB *.m files, interface files and dSPACE ControlDesk files, is explained. *Operating procedure* of test bed is also discussed, which includes flowcharts and screenshots of operation. The flowchart shown in figure 3.10 illustrates the operation of software interface.

Flowchart of the software interface

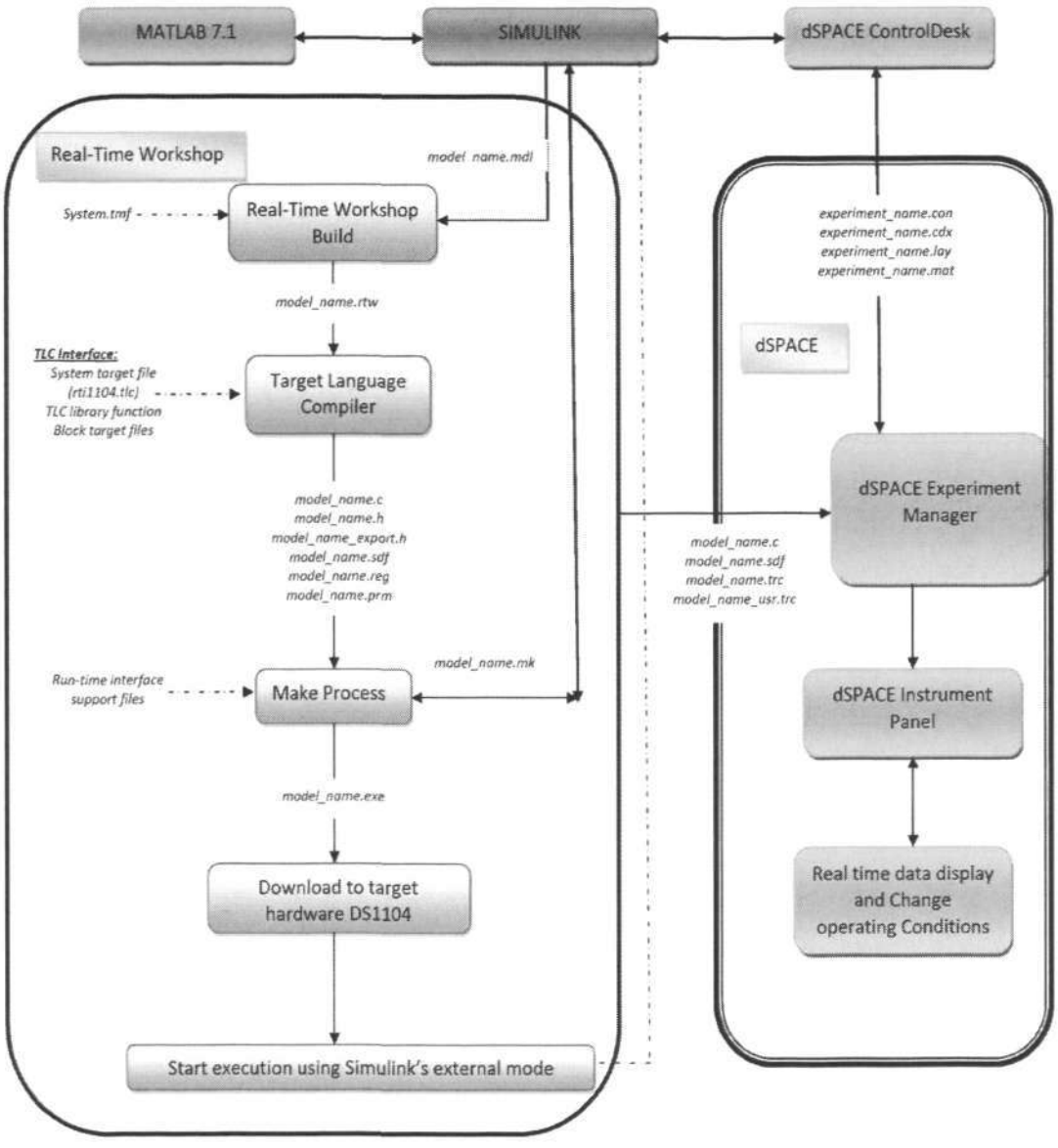


Figure 3.10 : Flowchart of software interface

Description

Software interface is constructed using MATLAB, SIMULINK, Real-Time Workshop and dSPACE interface library. All these interfaces perform in *unison* to make the test bed work efficiently. SIMULINK model is developed with IDE (Integrated Development Environment) given by SIMULINK interface and the model is saved as *model_name.mdl.* When model is *built* using RTW build module, model is eventually compiled and linked to form an *executable* file. In this process *system target file (rti1104.tlc), system.tmf, block target files and model_name.mk* are used. The compile and build process generates *model_name.sdf and model_name.c* files. The *model_name.obj* is downloaded on to dSPACE target using Real-Time Workshop interface. Now an executable file is downloaded and available for usage in dSPACE target system.

MATLAB and SIMULINK share the same workspace and hence an uncorrupted medium for exchange of real-time data is accomplished. MATLAB interface communicates with real-time target using variable references in *model_name.trc* files and hence parameters are updated on the go by MATLAB.

dSPACE interface develops an Instrument Panel(Graphical User Interface), which is used to analyze, monitor and display real-time data. dSPACE *Experiment Manager* is a tool which is used to create an experiment. An experiment is a Graphical User interface where variables used in dSPACE can be accessed for control operations. The *model_name.sdf, model_name.trc and model_name.c* files are loaded to the experiment. This process creates a reference in GUI for variables used in MATLAB and SIMULINK interfaces. *Experiment Manager* creates *experiment_name.cdx, experiment_name.con and experiment_name.lay* files when the experiment is constructed and linked to variable descriptor files.

3.4.1 Developing SIMULINK Interface (MDL)

In this section, steps involved in development and building of SIMULINK modules is explained in detail.

a. Creating a SIMULINK model

Open MATLAB and click New →model to create a new model. Save this model in the workspace as *model_name.mdl*. After creating the SIMULINK model, settings have to be changed in Simulation →Configuration Parameters. Figure 3.11 shows the settings page of Solver, Hardware implementation and Real-Time workshop tabs.

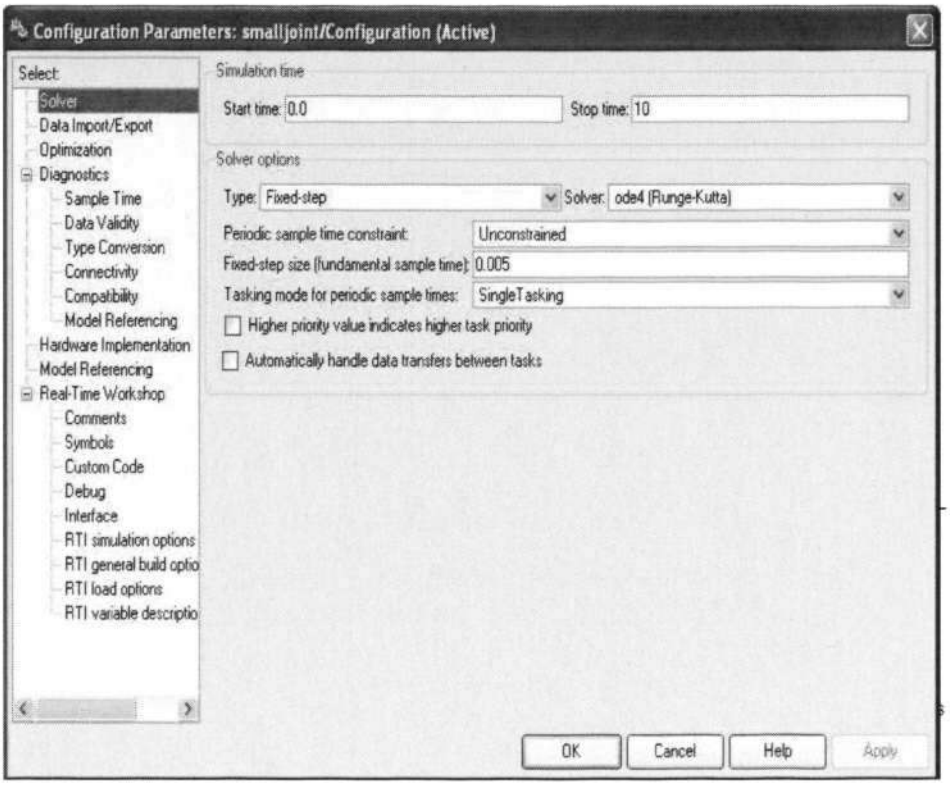


Figure 3.11 : Configuration parameters

The settings in the *solver tab* include start time and stop time of simulation, type of step options and sampling frequency settings. The settings in the *hardware implementation tab* include device type, byte ordering and emulation hardware. The typical settings of *Configurations tab* are shown in figure 3.11

Development of SIMULINK model

In this section, process of development of SIMULINK model according to the requirements of test bed is explained in detail. A case study consisting implementation of ILC algorithm on T2 axis of SCARA robot is used as an example for this purpose. This module uses SIMULINK library for building models. Figure 3.12 shows SIMULINK library view. It consists of SIMULINK main library and installed third-party dSPACE libraries.

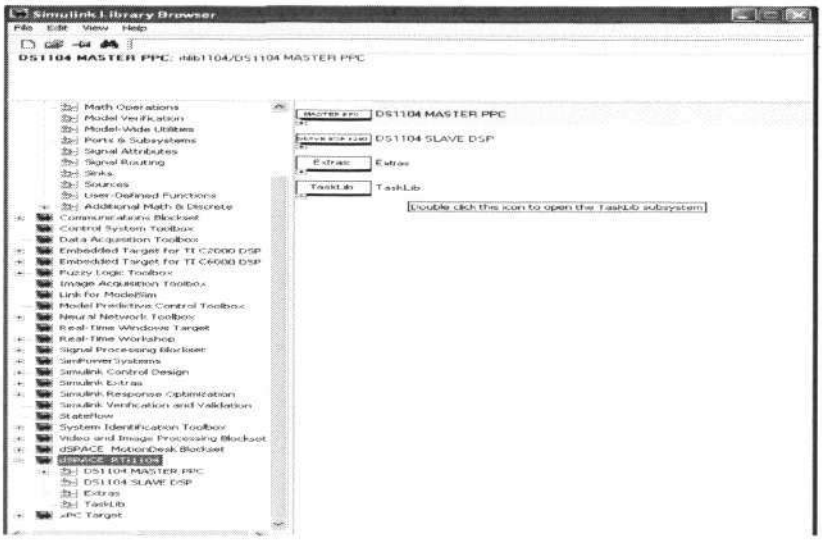


Figure 3.12 : SIMULINK library

Fig. 3.13 shows SIMULINK model developed for implementation of ILC on T2 axis

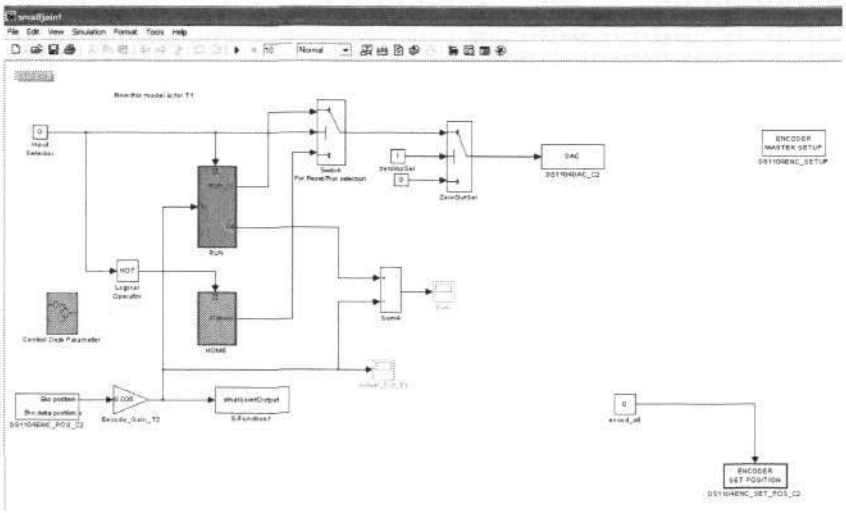
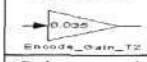
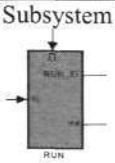
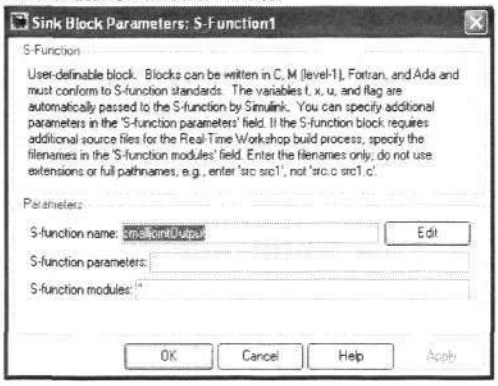
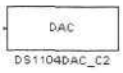
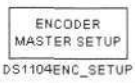
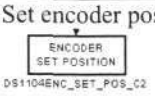
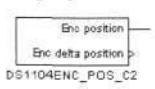


Figure 3.13 : Sample SIMULINK model

***Details of blocks commonly used in construction of SIMULINK model* [69]**

Block	Description
Constant block 	The Constant block generates a real or complex constant value. The block generates scalar (one-element array), vector (1-D array), or matrix (2-D array) output
Threshold switch block 	The Switch block passes through the first input or the third input based on the value of the second input. The first and third inputs are called data inputs. The second input is called the control input
Sum block 	The Sum block performs addition or subtraction on its inputs. This block can add or subtract scalar, vector, or matrix inputs
Gain block 	The Gain block multiplies the input by a constant value (gain)
Subsystem block 	A Subsystem block represents a subsystem of the system that contains it. The Subsystem block can represent a virtual subsystem or a true (atomic) subsystem, depending on the value of its Treat as atomic unit parameter. We are using a Sub-system which is activated by positive pulse.
User Built S-function block 	The S-Function block provides access to S-functions from a block diagram. The S-function named as the S-function name parameter can be a Level-1 M-file or a Level-1 or Level-2 C MEX-file S-function. We use a level-2 S-function and the S-function is written in C. The characteristics of S-function namely no. of I/O's, type of I/O, sample time, update parameters and dimensions are set in the C file. The C file should be placed in the same working directory as that of SIMULINK model. Once it is written, it is compiled by typing *mex filename.c* in the MATLAB Command Window. User defined S-function can now be used by entering the same *filename* of the C file
DAC block 	DAC block is obtained from dSPACE library and it gives digital input to one of the 8 DAC's of dSPACE board. Channel no. can be chosen from properties of the block.
Encoder Master set up block 	The encoder Master block determines the settings on the encoder whether it is double ended or single ended.
Set encoder position block 	This block is used to reset the encoder. We write zero to this block to reset the encoder.
Display encoder position block 	This block is used to read O/P of encoder counters. There are 2 counters, one for each encoder and the channel of the encoder can be chosen from the properties.

b. *Build procedure*

After creating SIMULINK model, model has to be *built* to generate an executable object file (*model_name.obj)*, C file (*model_name.c*), SDF file (*model_name.sdf*) and TRC file (*model_name.trc*). To build the model, Simulation→Configuration Parameters→Real-Time Workshop→Build or Ctrt+B or press  in SIMULINK. Figure 3.14 shows the settings of *Real-Time Workshop tab*. It consists of system target file (*rti1104.tlc*), language for compilation and TLC options settings.

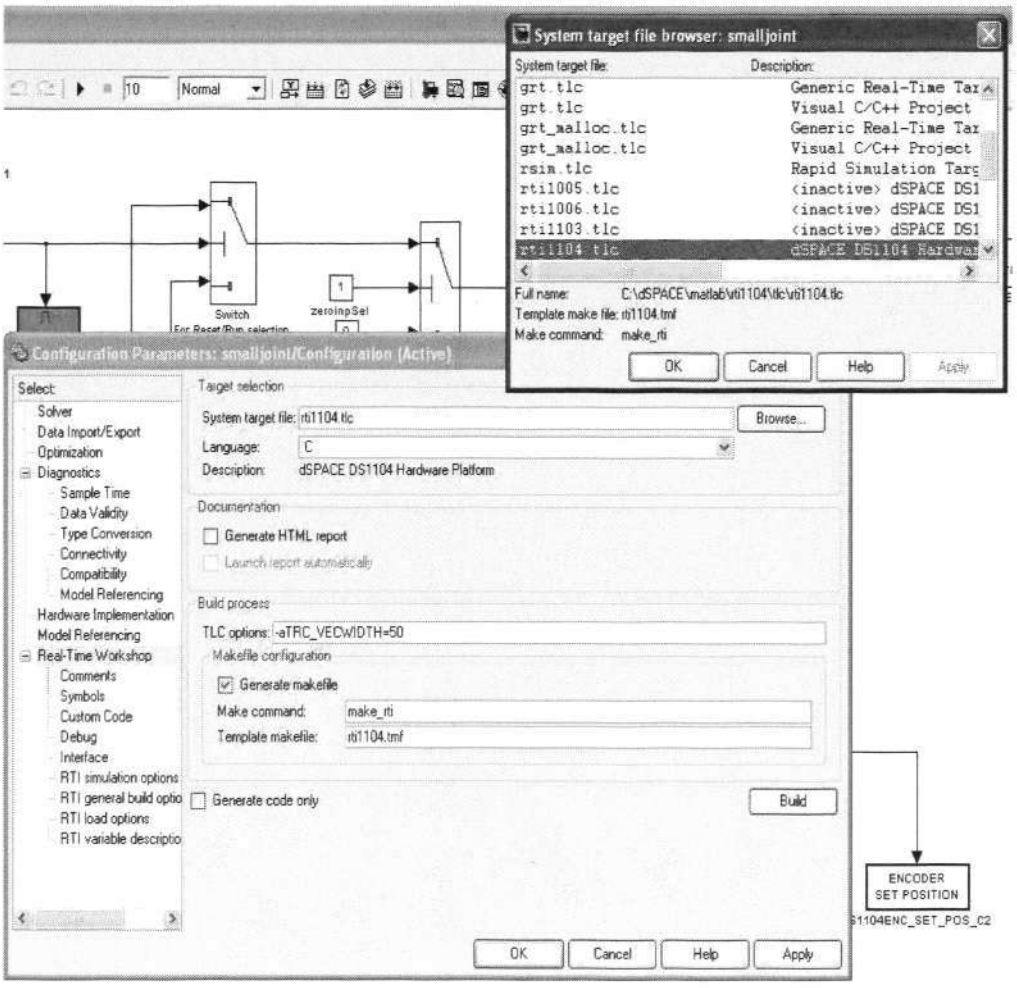


Figure 3.14 : Build procedure

3.4.2 Developing MATLAB Interface

The MATLAB interface is basically developed using the MATLAB M-files (with extension *.m). This interface is highly *modularized*, *i.e* it is divided into many modules. MATLAB interface communicates with dSPACE board with the help of MLIB/MTRACE interface library. Following are the modules developed in MATLAB.

- **Initialization module:** This module *initializes* the MLIB/MTRACE interface. It contains initialization and definition of variables used in dSPACE board. It also holds process variable references, such that MATLAB interface can communicate with other interfaces during real-time operation effectively.
- **Trajectory module**: This is a module developed to input desired trajectory. This module lets the designer choose from available desired trajectories and also feed a custom made desired trajectory using a simple MATLAB function.
- **Simulation time module**: This module is added to change operating time of iterations. In ILC, operating time of each iteration is fixed. Designer can choose a desired iteration time in this module and the system automatically compiles changes, accordingly.
- **ILC update law module**: There are different types of ILC update law. This module has a collection of standard update laws and a provision of adding custom designed update law also. System has an in-built collection of ILC laws such as, P-type, D-type, PD-type and A-type. If the designer wants to implement a new algorithm, there is a provision in this module to insert the new ILC update algorithm.

- **RMS (root mean square) error display module:** There is an improved display of the rms error histories in this system. Earlier, when desired numbers of iterations are completed then rms error of all the iterations was displayed. Now, the real-time value of rms error is displayed on the go.
- **Filter module:** This module is a MATLAB function which is used for signal filtering. If the designer wants to filter any signal in the system, adding references to this module is sufficient. He can choose cut-off frequencies or input his own cut-off frequency in this module.
- **Differentiation module:** This module is used for differentiation of signals. There are some algorithms that need some of the variables to be differentiated like, position differentiation and desired trajectory differentiation. If the designer wants differentiation to be performed, adding signal reference is sufficient.
- **Display module:** This module is related to read and write operations of the variables used in dSPACE Instrument Panel (GUI). Data exchanged between MATLAB and dSPACE GUI is basically used to plot and analyze process data. In this module, the data which is to be displayed is accessed from memory units in DS1104 board and MATLAB workspace. This data can be plotted, changed or displayed in GUI using MATLAB functions.
- **Manage module:** The manage module is used to observe the state of status variables and also to start and stop the control process. This module defines the start and stop procedures of ILC. It can be also use to monitor the status of process using status variables.

3.4.3 Developing dSPACE Interface

dSPACE interface is governed by *ControlDesk*. *ControlDesk* consists of *Experiment Manager, Instrument Selector, File Console* and *Layout Editor*. *ControlDesk* is used for real-time data monitoring and analysis. The *ControlDesk* is used to read/write real-time data to dSPACE board. The program variables used in SIMULINK/MATLAB can be displayed and changed from *ControlDesk* interface, because variable references in MATLAB/SIMULINK are downloaded to real time target through interface libraries. This section explains the development of dSPACE interface in detail. There are three modes of operation of *ControlDesk,* namely *edit* mode, *test* mode and *animation* mode. While creating the interface, *ControlDesk* should be *edit* mode.

i. Creating an Experiment

Experiment is created by the *Experiment Manager*. Open *ControlDesk* and select File→New experiment and store it by the name *experiment_name.cdx* in the same working directory as of MATLAB and SIMULINK files.

ii. Import files

Some software files have to be imported to an experiment like the object file, *model_name.map file* and *model_name.sdf file*. These files contain the references to the variables used in dSPACE board and MATLAB interface.

iii. Creating a layout

Layout is a file which contains the structure of instrument panel (Graphical User Interface). Select File→New→layout to create a layout and store it by the name *layout_name.lay* in the same working directory as that of MATLAB and SIMULINK files.

Figure 3.15 shows GUI developed in ControlDesk.

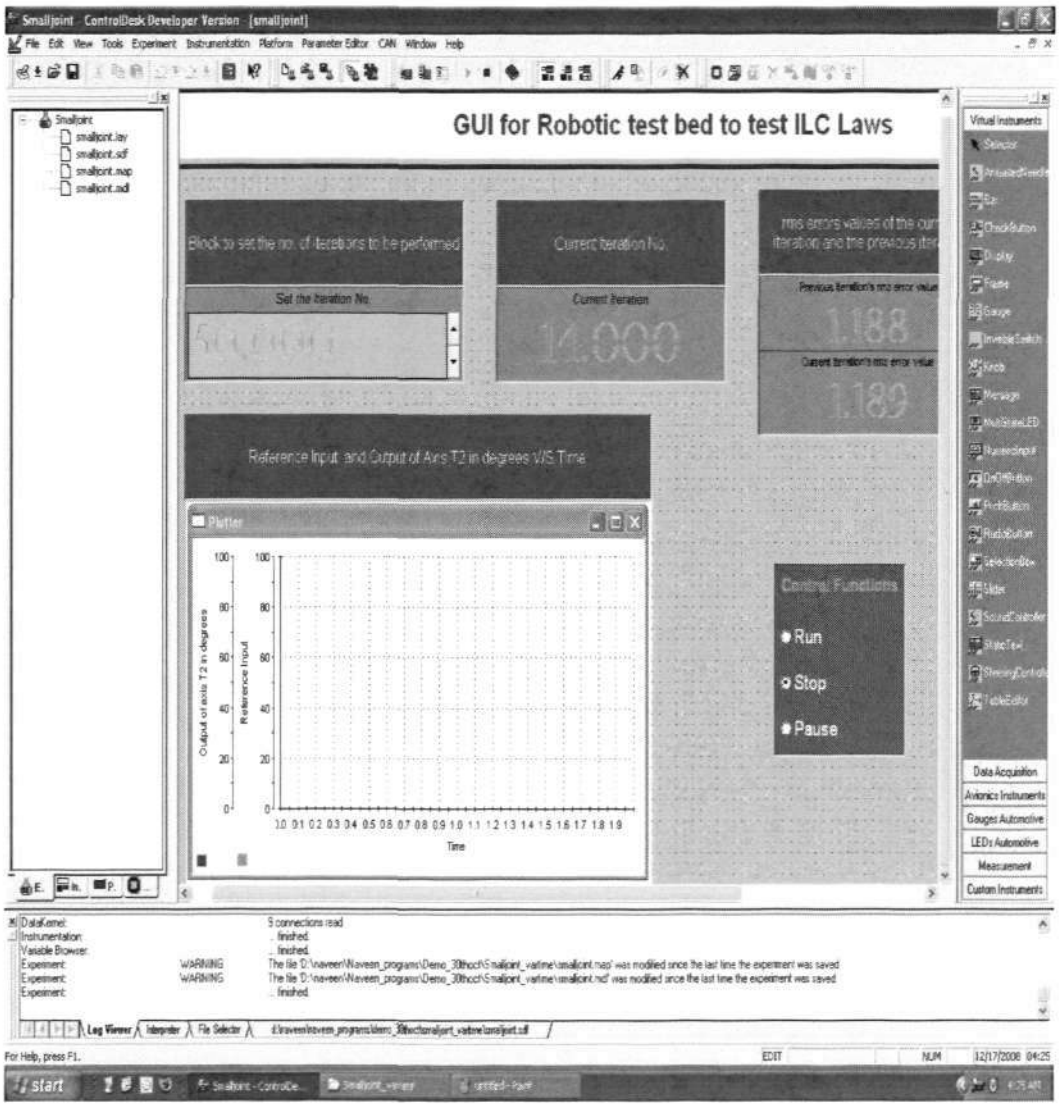
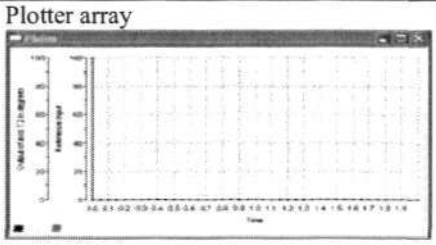


Figure 3.15 : GUI developed in ControlDesk

After creating a layout, layout file has to be imported to the experiment. There are a wide range of instruments available from *Instrument Selector*. We can drag and drop the instruments to the layout, as per developer's needs. After creating a structure of instrument panel in the layout, connection between instruments and process variables is established by creating *Layout_name.con* file. Following table [71] gives explanation of instruments used in test bed. Detailed explanation of all the instruments is found in the dSPACE manual [71] or dSPACE online help.

Instrument	Description
Plotter array 	Plotter array is an Instrument that is used to plot XY plot, Time plot and logic analyzer plot. We drag and drop the plotter array to the layout and double click on the plotter array to add these different types of plots. Using the properties tab of the plots the designer can set the attributes of the plot like the name, width, color, legend, axis dimension, gridlines, type of plotting, format of display and many more.
Radio button 	This instrument is used to assign a particular set of values to the variable connected to it. One click of a radio button activates a designer chosen value for the variable. The properties and values of the radio button can be set by double clicking on the radio button in the layout.
Display 	This instrument is used to display the numerical values of the variables.

iv. Creating a connection between instruments and variables

Once instruments are inserted into the layout, they have to be connected to process variables. These variable references will be available, once SDF file of the model is loaded to Experiment. Click on the chosen SDF file tab at bottom of the screen. We will get a list of available referenced variables in bottom left corner of the screen. Now the variables to be used can be dragged and dropped to instrument in the layout. After assigning the variables to the instruments in the layout, we save the layout. The connection between the variables and the instrument is stored in a file *layout_name.con*. Figure 3.16 shows the process of linking and instruments and variables. Bottom left corner shows the list of available variables and top right corner is the layout.

v. Operation of the interface

Save the experiment and switch mode of ControlDesk to *animation* mode to analyze and monitor real-time data.

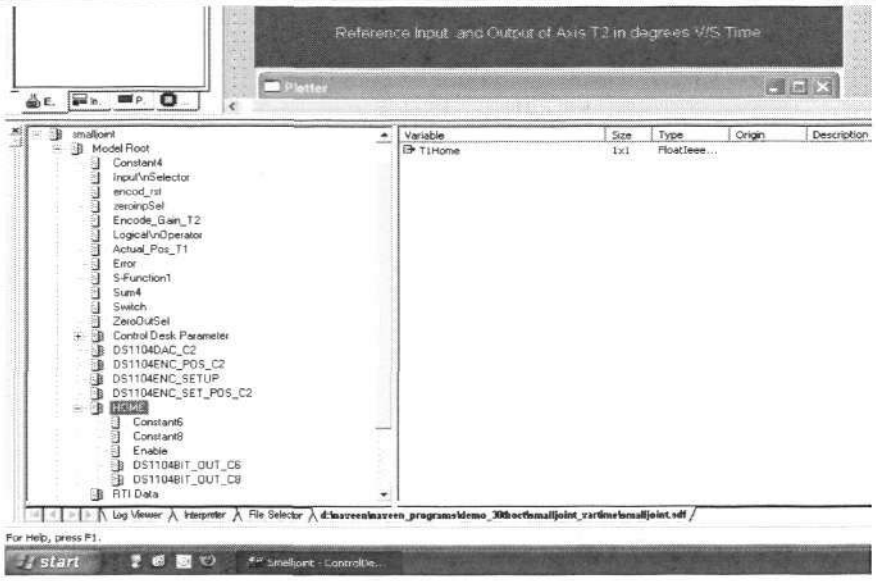


Figure 3.16 : ControlDesk GUI

3.5 Operating the Robotic test bed

1. Develop SIMULINK model with settings as explained in the sections above, according to design requirements. **Build** the model
2. Develop the dSPACE ControlDesk **experiment** and switch to animation mode.
3. Develop **MATLAB files** and run.
4. Open **experiment** and observe real-time data using ControlDesk GUI
5. Start and Stop the operation from ControlDesk GUI.

3.6 Summary

Robotic test bed is developed for implementing various types of ILC algorithms. Test bed developed meets the current industry trends and requirements. An attempt is made to recreate the atmosphere present in any industrial scenario. It is equipped with latest software like dSPACE 5.1 & MATLAB 7 and latest version of hardware like power amplifier 25A8. Fundamental attributes of test bed are that, it is highly user-friendly and flexible. It is easy to use and modify. Designer can choose from available pieces of code or integrate his own piece of code for testing and future development.

Chapter 4

Implementation of ILC on robotic test bed

In this chapter, development and implementation of ILC on robotic test bed is explained in detail. The vital requirement of robotic test bed is to be able to *test* various types of ILC algorithms on the SCARA robot. We have introduced the details of robotic test bed including specifications of SCARA robot, the hardware and software configuration in chapter 3. This chapter deals with algorithm implementation of ILC on robotic test bed.

Basic *ILC laws* like P-type and A-type laws are designed and implemented on test bed. Before proceeding to the implementation of ILC laws on test bed, we implement a simple *feedback control* on test bed. PID controller is designed and tuning of parameters is achieved through Ziegler-Nichols' method. This is done basically to condition the control platform for testing various ILC laws. The results obtained by ILC implementation on SCARA robot can be compared to a feedback controller. Furthermore, *feedback* controller constitutes the *online* phase of ILC and *offline* phase is formed by learning controller.

In later sections of the chapter, development and implementation of ILC on robotic test bed is discussed. The development is explicitly illustrated in a step-wise manner. This includes *flowcharts* and *algorithms* used in developing software architecture. It also consists of developed software models and GUI used to monitor and control ILC operation. The detailed description of software code and logic used in implementation of ILC is provided.

4.1 Implementation of feedback control

In this section, implementation of PID control on robotic test bed is discussed. PID control is an effective and most popular method of *feedback control*. PID algorithm *forces* the system to move in the direction, which should move the process output towards a set-point. PID control algorithm is realized in MATLAB/SIMULINK and dSPACE R&D controller board. Results are plotted on GUI developed in dSPACE ControlDesk.

An independent PID controller is designed for each of the control loops formed by joints T1 and T2 axis of the test bed. PID controller must be "*tuned*" for a particular control loop, in order to operate accurately. *Tuning* is based on the dynamics of process response. Effective PID loop tuning was achieved by Zeigler-Nichols method.

Tuned values of P, I and D obtained for T2 axis by Ziegler-Nichols' method

$Gain\ (\boldsymbol{P}) = \mathbf{0.6}\boldsymbol{K_u} = \mathbf{0.24};\quad Integral\ (\boldsymbol{I}) = \boldsymbol{P_u}/\mathbf{2} = \mathbf{0.1};\quad Derivative\ (\boldsymbol{D}) = \boldsymbol{P_u}/\mathbf{8} = \mathbf{0.025}$

where K_u and P_u are parameters obtained by PID tuning.

The SIMULINK model used in feedback control

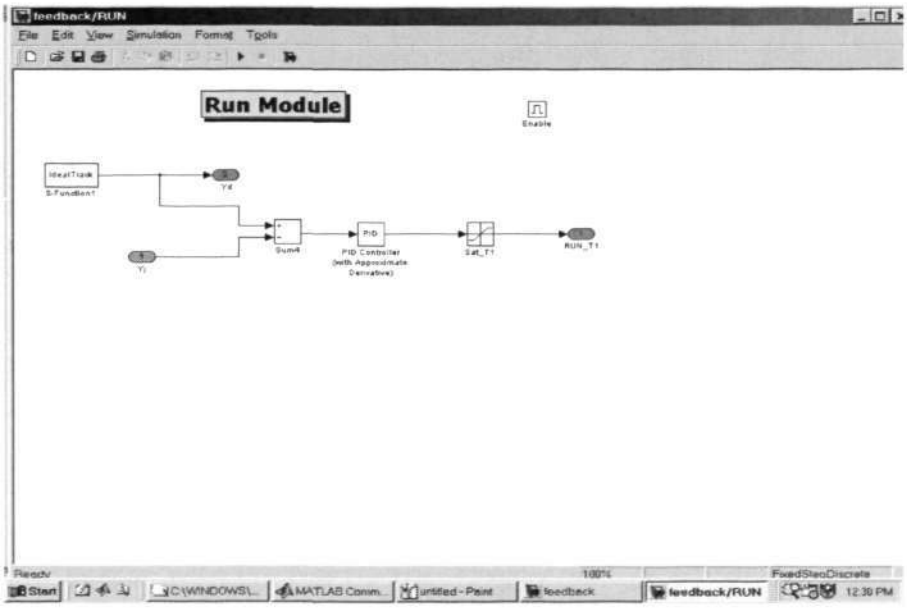


Figure 4.1 : SIMULINK model used in feedback control

GUI used for feedback control

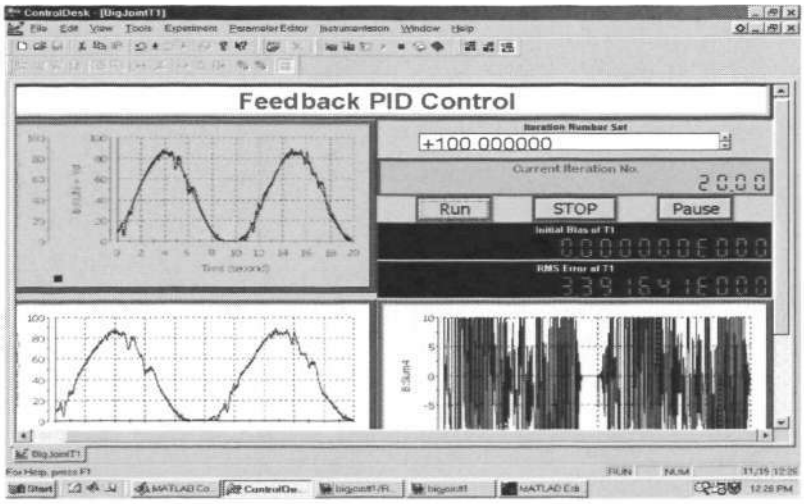


Figure 4.2 : GUI used in feedback control

Result for feedback control

T2 axis of a SCARA robot is allowed to execute a desired trajectory repeatedly with constant P, I and D values. Fig 4.3 shows RMS error history along iteration axis.

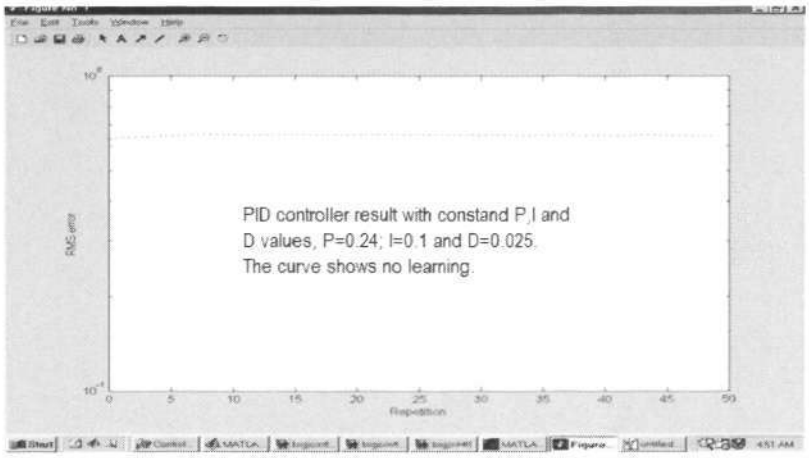


Figure 4.3 : Results for feedback control

The RMS (root mean square) error is constant at 1.2566. In other words, the graph demonstrates that there is no learning in feedback controllers. Hence this property of feedback controllers can be *overcome* using feed-forward ILC controllers, which reduces tracking error along the iteration axis. Therefore, ILC is found advantageous for systems which perform the same action again and again.

4.2 Implementation of ILC

Firstly, details regarding software flowchart, software code and software assembly are presented. This section also demonstrates the *synchronization* of various software interfaces used. It also gives an *insight* on different types of ILC. For this purpose, only two axes (T1 and T2) are used for implementing ILC laws. This limitation is by the fact that dSPACE ControlDesk has only two encoder interfaces. Control characteristics related to other two axes are identical to T1 and T2 axes.

4.2.1 Problem formulation for ILC implementation

Each axis of the test bed is considered as an independent Single-Input Single-Output system, for design of ILC. We use T2 axis in all the derivations presented in this chapter. The operating conditions for T1 axis is derived in the same way as T2 axis.

Consider a Single-Input Single-Output continuous time-invariant system represented in state space notation as,

$$\dot{x}(t) = Ax(t) + Bu(t) + w(t) \tag{4.1}$$

$$y(t) = Cx(t) + Du(t) + v(t) \tag{4.2}$$

where x is a *n-dimensional* state vector, u is the scalar input, y is the scalar output, w and v are state and output disturbances respectively.

One of the important *postulates* of ILC is that, the *operating time* should be constant in all iterations. Therefore, we define $t \in [0, t_f]$, t_f is the final time of each iteration.

At iteration no. k

$$\dot{x_k}(t) = Ax_k(t) + Bu_k(t) + w(t) \tag{4.3}$$

$$y_k(t) = Cx_k(t) + Du_k(t) + v(t) \tag{4.4}$$

Let the desired trajectory be $y_d(t)$ and $t \in [0, t_f]$.The main objective of ILC is to iteratively find control input that drives the system to produce the desired output.

A typical ILC update law is defined as,

$$\dot{x_k}(t) = Ax_k(t) + Bu_k(t) + w(t) \tag{4.5}$$

here $j \leq i$ and $e_k(t) = y_d(t) - y_k(t)$. $L(.)$ is the learning gain function chosen by designer. Problem is formulated to design and develop convergence and robustness conditions for P-type and A-type ILC laws. Based on convergence and robustness analysis performed in frequency domain, ILC laws are implemented on test bed. Typical update laws are stated to be,

$$\boldsymbol{P-Type}:\ u_{k+1}(t) = u_k(t) + \varphi(y_d(t) - y_k(t)) \tag{4.6}$$

$$\boldsymbol{D-Type}:\ u_{k+1} = u_k + \varphi(\dot{y}_d(t) - \dot{y}_k(t)) \tag{4.7}$$

$$\boldsymbol{A-Type}:\ u_{k+1} = u_k + \varphi(y_d(t+\Delta) - y_k(t+\Delta)) \tag{4.8}$$

where Δ is the lead time, φ is the learning gain.

4.2.2 Convergence analysis

The convergence analysis of P-type and A-type are performed for T2-axis of robotic test bed. Frequency domain approach for convergence analysis is exercised.

A typical P-type update law is represented as

$$u_{k+1}(t) = u_k(t) + \varphi\big(y_d(t) - y_k(t)\big) \tag{4.9}$$

In Laplace domain, $$U_{k+1}(s) = U_k(s) + \varphi E_k(s) \tag{4.10}$$

System is defined in Laplace domain as,

$$Y_k(s) = G_p(s)U_k(s) \tag{4.11}$$

$$Y_{k+1}(s) - Y_k(s) = G_p(s)\big(U_{k+1}(s) - U_k(s)\big) \tag{4.12}$$

We know that $$Y_{k+1}(s) - Y_k(s) = G_p(s)\varphi E_k(s) \tag{4.13}$$

Since, $$Y_{k+1}(s) - Y_k(s) = -\big(E_{k+1}(s) - E_k(s)\big) \tag{4.14}$$

We deduce that, $$E_{k+1}(s) = \big(1 - \varphi G_p(s)\big)E_k(s) \tag{4.15}$$

From Eq 4.15, in order to ensure convergence,

$$\left|1 - \varphi G_p(s)\right| < 1 \tag{4.16}$$

In frequency domain, $$\left|1 - \varphi G_p(j\omega)\right| < 1 \tag{4.17}$$

Considering, $$G_p(j\omega) = N_p(\omega)\, e\left(j\theta_p(\omega)\right) \tag{4.18}$$

where $N_p(\omega)$ is magnitude characteristic of plant (T2 axis) and $\theta_p(\omega)$ is phase characteristic. The convergence condition is reframed as,

$$\left|1 - \varphi N_p(\omega)\, e(j\theta_p(\omega))\right| < 1 \tag{4.19}$$

$$\left|1 - \varphi N_p(\omega)\, [cos\ \theta_p(\omega) + jsin\ \theta_p(\omega)]\right| < 1 \tag{4.20}$$

Simplifying the above inequality,

$$\varphi^2 N_p(\omega) < 2\varphi \cos \theta_p(\omega) \tag{4.21}$$

The convergence condition for the P-type ILC law is

$$\boldsymbol{\varphi N_p(\omega) < 2 \cos \theta_p(\omega)} \tag{4.22}$$

To satisfy the above mentioned inequality,

$$\boldsymbol{\left|\theta_p(\omega)\right| < 90^0} \tag{4.23}$$

In a similar way, convergence conditions for other ILC schemes, such as A-type and D-type ILC can be derived.

Convergence conditions for A-type ILC

$$\boldsymbol{\varphi N_p(\omega) < 2\cos\left(\theta_p(\omega) + \Delta\omega\right)} \tag{4.24}$$

$$\boldsymbol{\left|\theta_p(\omega) + \Delta\omega\right| < 90^0} \tag{4.25}$$

Convergence conditions for D-type ILC

$$\boldsymbol{\varphi N_p'(\omega) < 2 \cos \theta_p'(\omega)} \tag{4.26}$$

$$\boldsymbol{\left|\theta_p'(\omega)\right| < 90^0} \tag{4.27}$$

where $N_p{}'(\omega)$ and $\theta_p{}'(\omega)$ are magnitude and phase response of $G_p{}'(s) = sG_p(s)$

4.2.3 Robustness analysis

ILC has to be *robust* to plant uncertainties and modeling uncertainties. The *divergence* between the actual system and its mathematical model can lead to infringement of performance conditions and also affect stability of the control system. Plant uncertainty can parametric or non-parametric or both. ILC should *offer* robustness to such plant uncertainties and variations.

Typical sources of *uncertainty* include unmodelled high frequency dynamics, neglected parameters, such as saturation, threshold, hysteresis, unmodelled non-linearity in plant dynamics and environmental factors. In order to make ILC based control system robust, convergence conditions are modified or relaxed, as shown below.

The convergence and robustness conditions together are defined as

Robustness conditions for P-type ILC

$$|\theta_p(\omega)| < 90^0 - \epsilon \tag{4.28}$$

Robustness conditions for A-type ILC

$$|\theta_p(\omega) + \Delta\omega| < 90^0 - \epsilon \tag{4.29}$$

Robustness conditions for D-type ILC

$$|\theta_p{}'(\omega)| < 90^0 - \epsilon \tag{4.30}$$

where $\epsilon > 0$ and chosen by the designer, $\theta_p(\omega)$ *and* $\theta_p{}'(\omega)$ are parameters from convergence equations Eq 4.24 and Eq 4.28. Based on convergence and robustness analysis performed for P-type and A-type ILC, obtained learning gain values are applied to T2 axis of robotic test bed.

4.3 ILC based control system

Figure 4.4 explains the block diagram of ILC based control system. It depicts implementation details of ILC algorithm on robotic test bed. The implementation of ILC algorithm is divided to two phases, namely *feedback* and *feedforward* phase. The *feedback* phase is realized using a PID controller and learning controller is constitutes the *feedforward* phase.

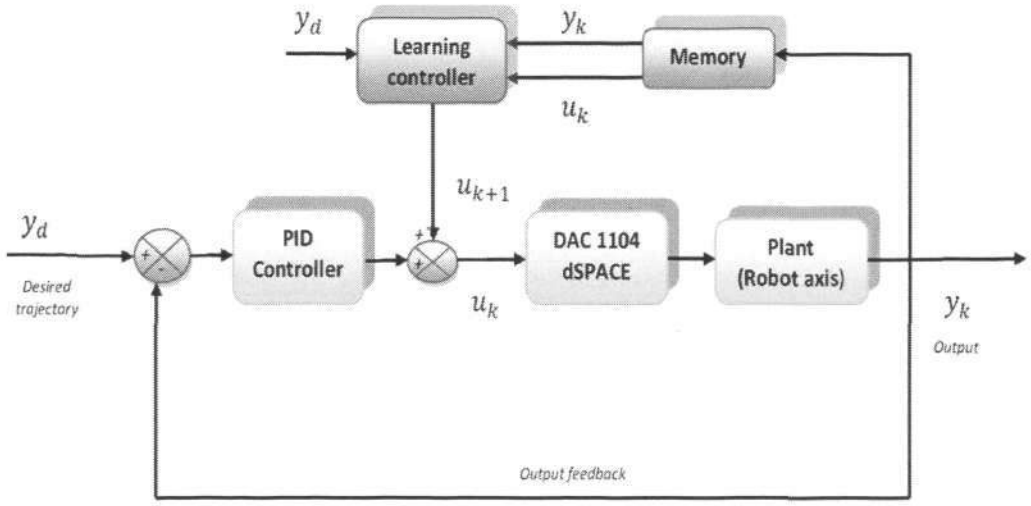


Figure 4.4 : ILC based control system

The *feedback phase* consists of PID controller which uses output position feedback of the current iteration and desired trajectory for its operation. *Feedforward* phase consists of a *memory unit* and *learning controller*. Memory unit stores output and input values of current and previous iterations. Learning controller uses these stored values from memory unit and calculates control input for the next iteration. The advantage this architecture offers is that, even when *feedforward* phase is isolated from system, the control system behaves as a *feedback* controller. *Feedback controller* can be allowed to operate on the system even without *learning controller*.

Software Flowchart of control system

Figure 4.5 below shows software flowchart of the ILC based control system

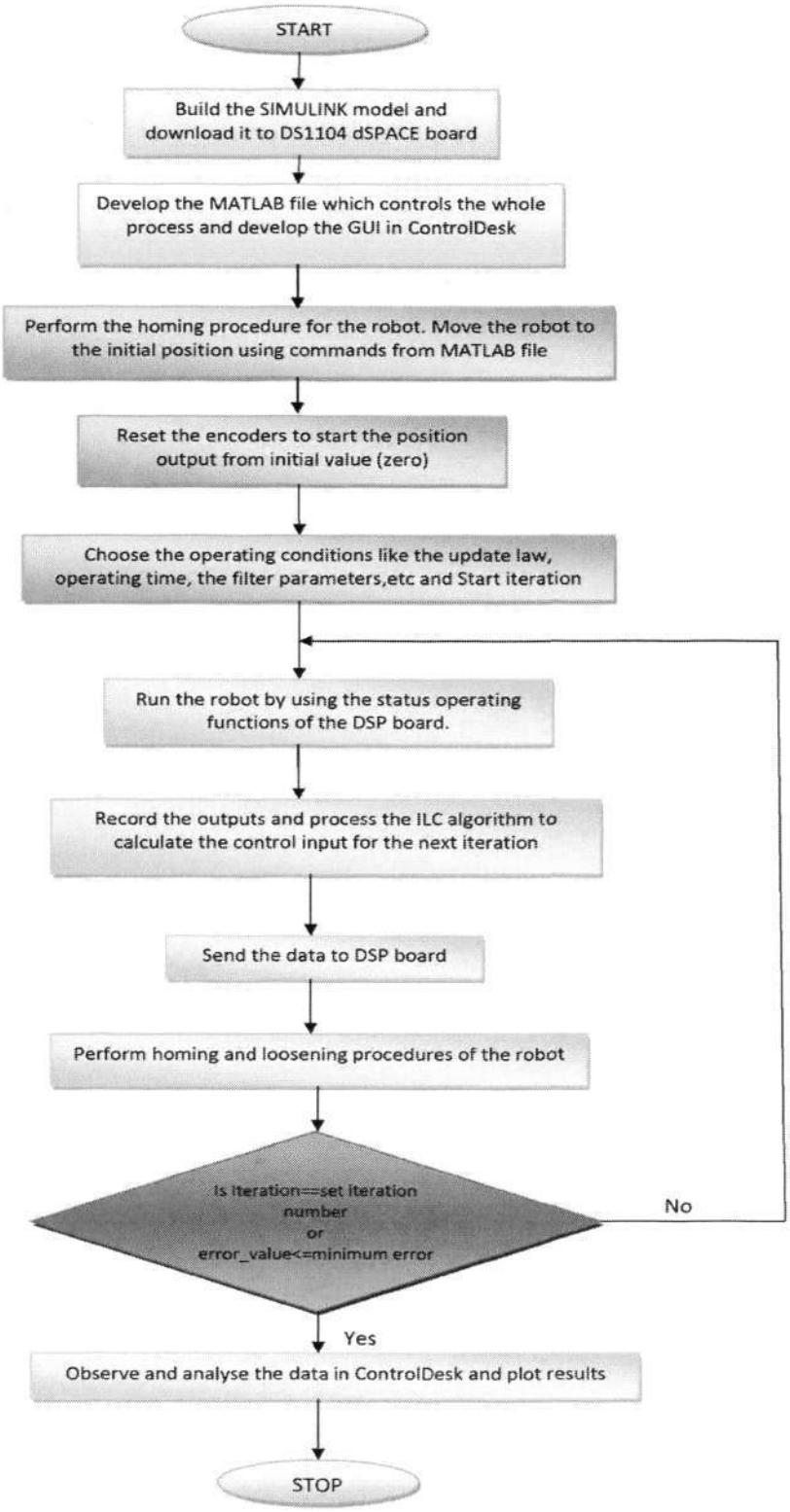


Figure 4.5 : Software flowchart of ILC based control system

Software flowchart explains the *Software development life cycle* of ILC based control system. The first step is to develop software programs and models. SIMULINK model is constructed, which is the backbone of control system. Constructed SIMULINK model is downloaded to dSPACE DS1104 processor board using RTW. Graphical User Interface (GUI) is developed in ControlDesk, which is used to observe and analyze real-time data. MATLAB M-file is developed, which is main software file to control the entire ILC operation.

Robot is moved to the initial position, in other words referred to as *homing procedure*. The initial position is determined by the *mechanical limitations*, which are provided by the restrictions mounted on robot. The position encoders are reset, so that the output position starts from zero after the robot is moved to home position. Resetting the encoder is an important operation in this process because the encoder used here is an *incremental encoder i.e*, after the counter is reset, encoder counter starts from zero irrespective of position of the robot arm within the work envelope. The initial operating conditions of ILC like the operating time, update law, filter parameters, the desired trajectory are chosen from MATLAB M-file.

Now, iterations are started. Based on control input calculated or assumed during the first iteration, robot is operated by using DSP board status based functions from MATLAB. Position outputs are recorded and ILC law is applied based on tracking error and control input for the next iteration is calculated. The data is sent to the DSP board and robot homing and loosening procedures are performed so that next iteration starts with the same initial conditions. The iterative learning procedure is performed until number of iterations reaches the set number of iterations or until RMS (root mean square) error value reaches a *threshold* value. The results are plotted in MATLAB and ControlDesk GUI.

4.3.1 Construction of SIMULINK model

SIMULINK offers an effective and flexible method for constructing models which can be used for various engineering applications. In this control application, SIMULINK is used to construct the *basic module* which forms the backbone of the ILC operation. This basic module is sub-divided into two inter-dependent modules, namely *homing module* and *run module*.

Module 1: Basic Module

Basic module (figure 4.6) is a key module, which houses various triggered sub-modules like *run* and *home* modules. It also consists of assembly of switches to shift between run, home and loosening procedures. Basic module also comprises of DAC interface and encoder output interface added from dSPACE interface library.

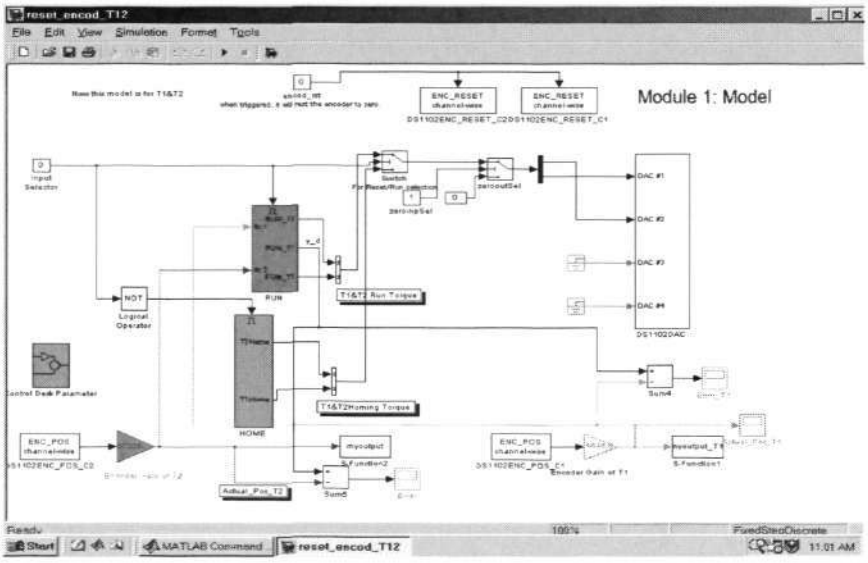


Figure 4.6 : Basic SIMULINK module for ILC with home and run sub-modules

Switching between run and home modules is performed by changing status of selector variable, because home and run procedures are *mutually exclusive*. There exists a loosening procedure, which outputs 0 to DAC. During ILC operation, homing is executed first and it is followed by loosening and run procedures. Encoder and DAC blocks represent the output and input of system, respectively.

Module 2: Homing Module

After each cycle of operation, system should return to its initial (home) position. Homing procedure in this context represents the return of robot joints to its hardware limit positions. At the end of each trial, homing module (figure 4.7) is activated by *basic module*.

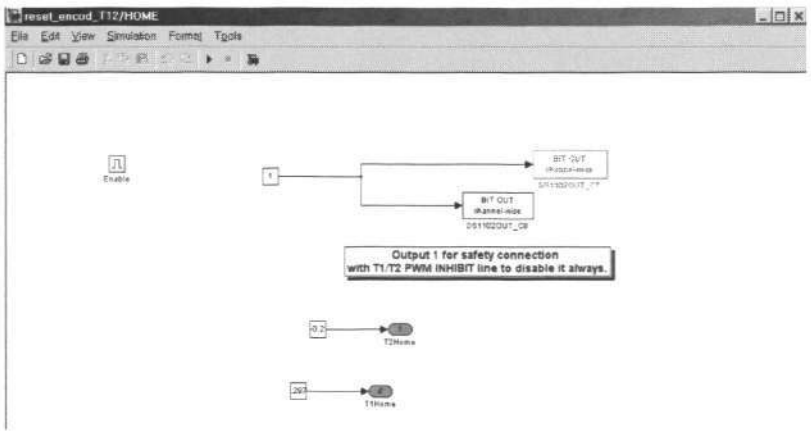


Figure 4.7 : Homing module

Module 3: Run Module

The *run module* (figure 4.8) consists of interfaces to control input (calculated by ILC algorithm) and feedback controller.

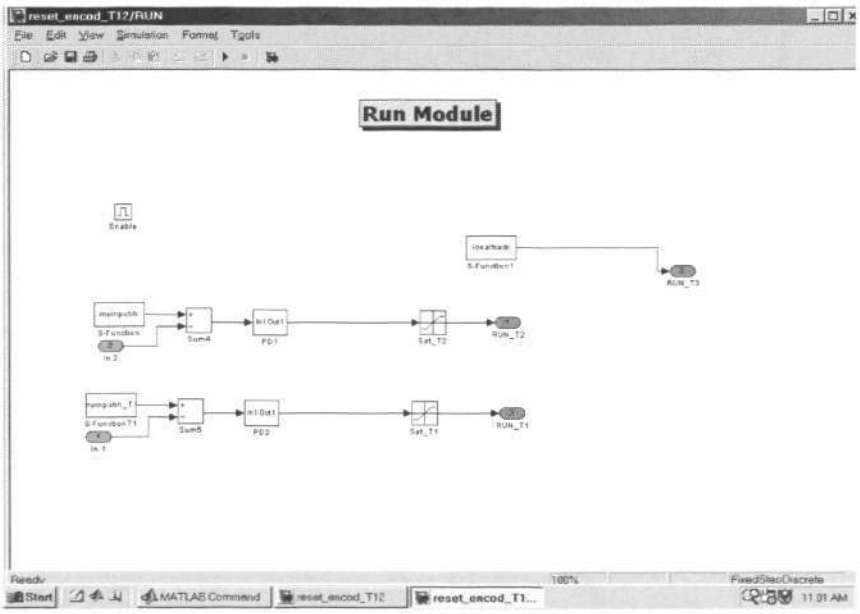


Figure 4.8 : Run module

Since, two axes (T1 and T2 axis) are used for implementing ILC, two independent SISO loops with a PID controller and ILC are implemented. The updated control input values for the next iteration are passed on from the memory unit to run module using interface libraries.

4.3.2 C Files (User defined S-Functions)

C-files (S-functions) work in conjunction with "MLIB/MTRACE" interface library. Read and write operations related to these S-Functions are governed by interface libraries. These interface libraries are invoked using MATLAB *.m files. Different types of S-functions developed and their functions are explained below.

***Output_Tx.c**:* This C-coded S-Function stores position output values of current iteration for specified number of samples for T1 and T2 axes. These values are used in the ILC algorithm to calculate the control input for the next iteration.

***Input_Tx.c**:* S-Function to store control input for next iteration for T1 and T2 axes.

***Idealtrack.c**:* S-function to store desired trajectory for a specified number of samples.

Here ***x*** represents axis number. The advantage of using C-coded S-functions is that, it can be *reusable*. These functions can be used in any other SIMULINK models by including the function in main MATLAB work directory. In other words, compilation of C-codes leads to addition of functions to the existing MATLAB library.

4.3.3 Construction of MATLAB file

Main ILC algorithm is implemented by MATLAB M-file file. This file *controls* the entire ILC operation. It consists of *library* functions which can communicate with DS1104 board. The operations like resetting encoders, switching between run, home and loosening modules, setting or resetting status of DSP board are invoked within this module.

It consists of a number of *sub-routines* (also MATLAB M-files), which offers robotic test bed a higher flexibility and performance. The sub-routines that are included in main MATLAB M-file are,

- Function for generating the desired trajectory
- Function for selecting operating time for each iteration
- Function for re-building the SIMULINK model, when settings change
- Function for selecting a suitable update law or to develop a new update law
- Function for error calculation and processing
- Function for calculating control input for the next iteration
- Function to send the updated parameters to DS1104 board
- Function to calculate RMS(root mean square) error values
- Function to plot, store and display the real-time results
- Function to control start and stop operation of the ILC on test bed

4.3.4 Construction of dSPACE GUI

The Graphical User Interface (GUI) for robotic test bed is designed in dSPACE Control desk, version 5.3. It consists of plotter display for output signal (position value from encoder), desired trajectory, error profile, control input and RMS (root mean square) error. All the results that are plotted in the GUI are real-time as it obtains real-time data from the DS1104 board using interface libraries.

It also can be used to enter or alter values of parameters dynamically to DS1104 processor. Some parameters that can be altered dynamically are number of iterations, choosing learning control law, desired trajectory and some initial values like initial offset and initial bias. Variable descriptor files link variables in MATLAB/SIMULINK to Control desk GUI. Connections between GUI and MATLAB/SIMULINK variables are stored in *layout_name.con* file in *Experiment Manager*.

Graphical User Interface developed in dSPACE (figure 4.9)

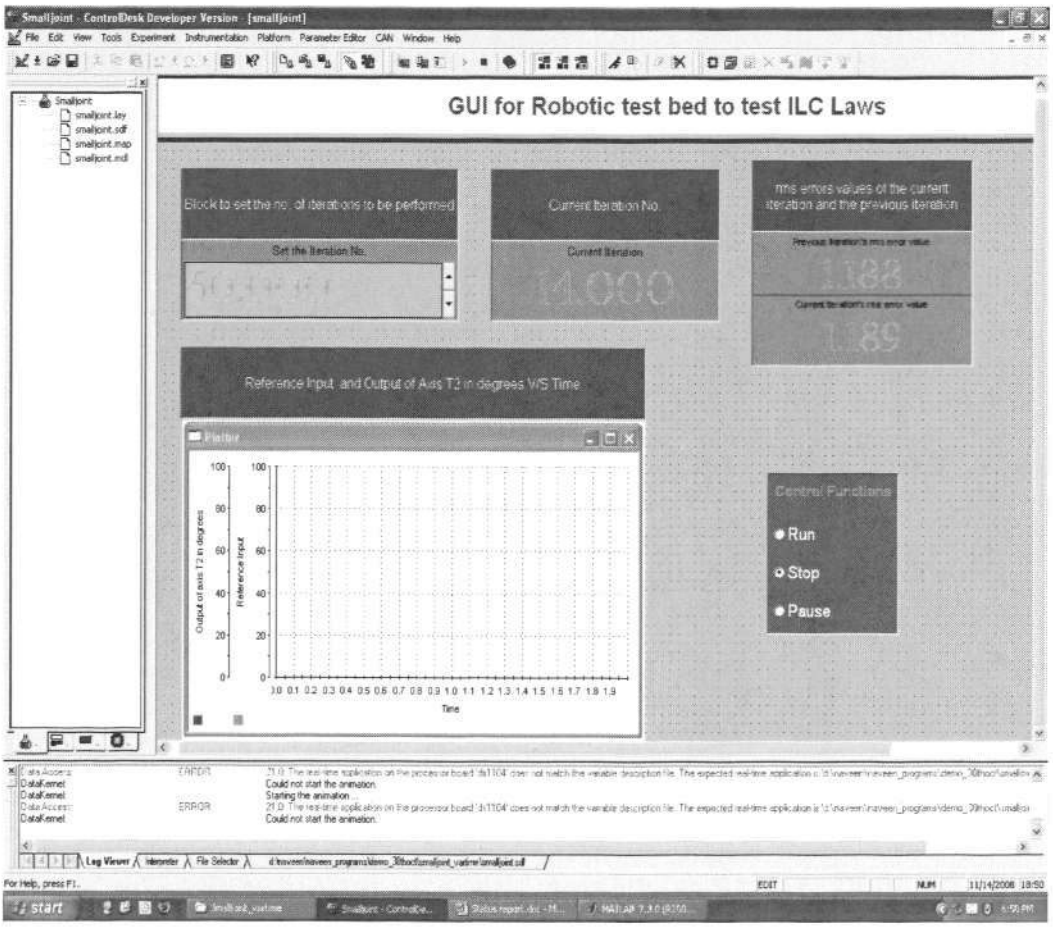


Figure 4.9 : Complete GUI for ILC application

4.3.5 Interface files

Interface files are written to make user-defined variables in MATLAB/S-functions to be accessible by the dSPACE controller board and GUI. Interface files are variable descriptor files which act as a communicator between MATLAB/SIMULINK and dSPACE. These are either written in notepad or C. This file consists of all the variable declarations that are user-defined in the algorithm. The formats of these declarations are written in a way compatible and usable by upgraded versions of dSPACE and MATLAB.

4.4 Experimental Results

P-Type ILC algorithm was developed and following diagrams (fig 4.10, fig 4.11, fig 4.12 and fig 4.13) depict the results obtained.

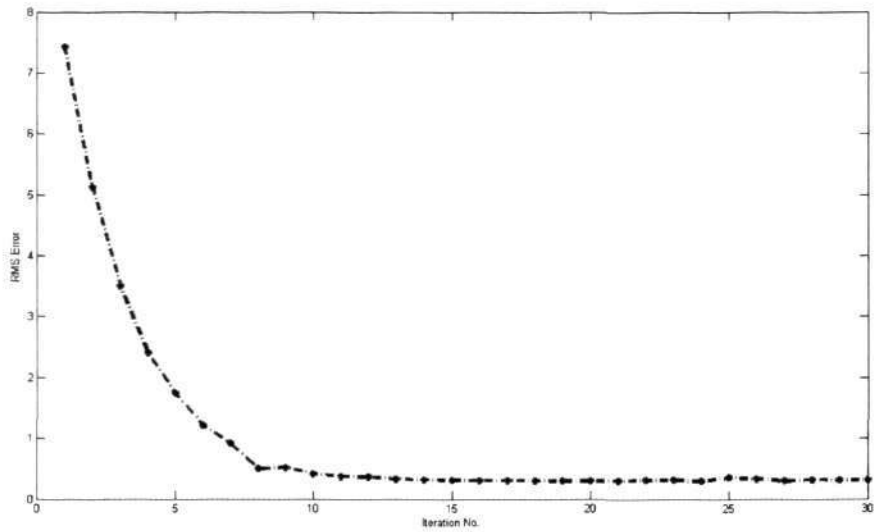


Figure 4.10 : RMS error profile of P-type ILC

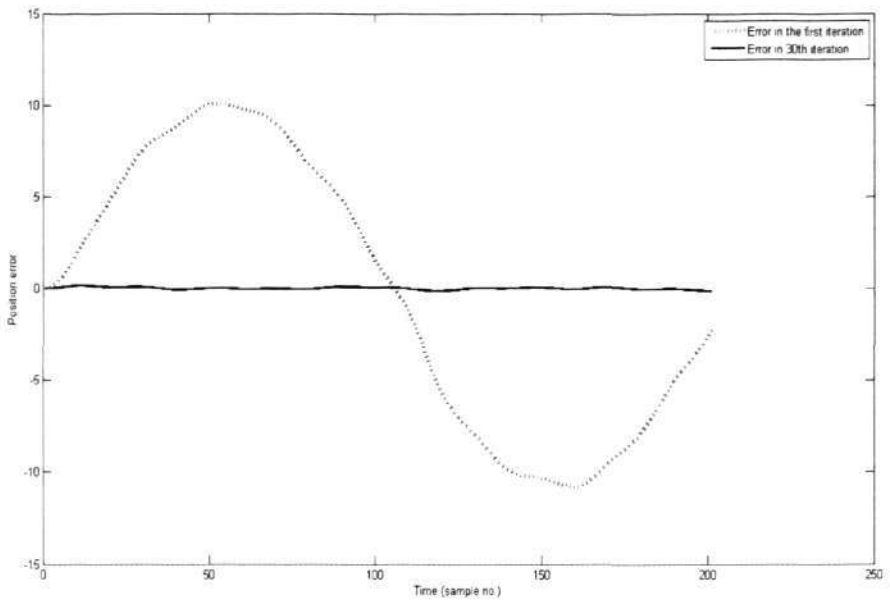


Figure 4.11 : Error profile of P-type ILC at iterations 1 and 30

Unit of vertical axis : Position in degrees

Unit of horizontal axis : Time (sample number)

Sampling period is 0.01 sec. From figure 4.11 the number of samples in the desired trajectory is 200. Hence the desired trajectory is tracked in 2 seconds.

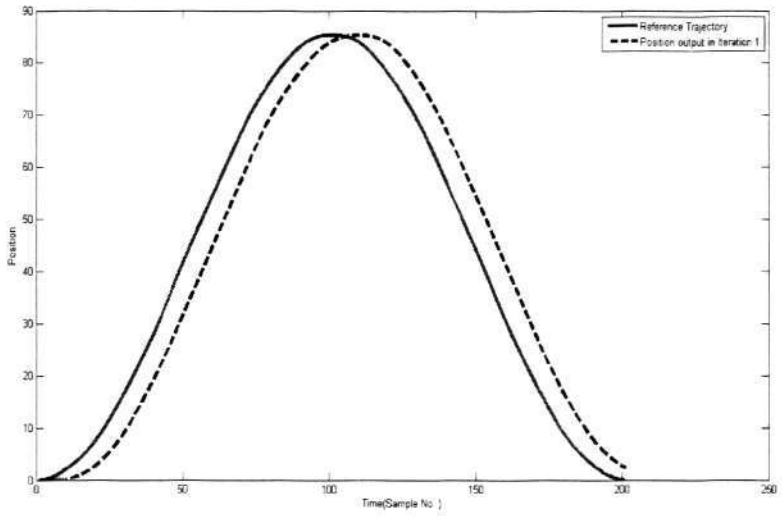


Figure 4.12 : Input and output of P-type ILC at iteration 1

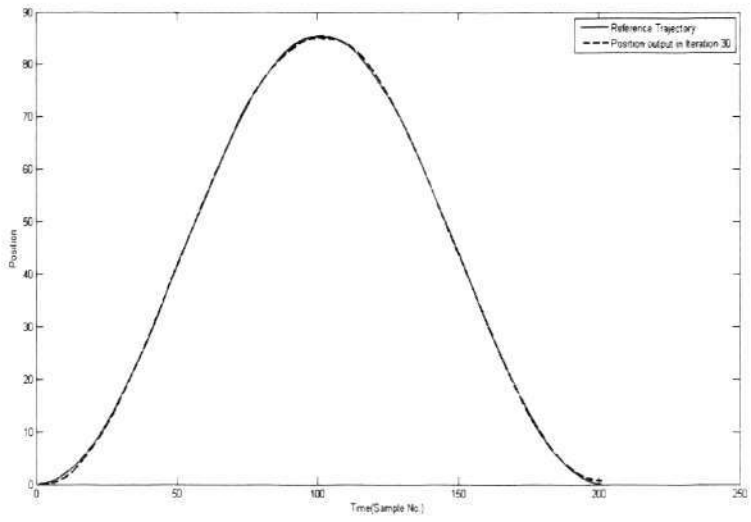


Figure 4.13 : Input and output of P-type ILC at iteration 30

A-type ILC

Detailed description of steps to choose the lead time (Δ) of A-type ILC is provided in Appendix C. The value of lead time used in this experiment is $\Delta = 0.04\ sec$

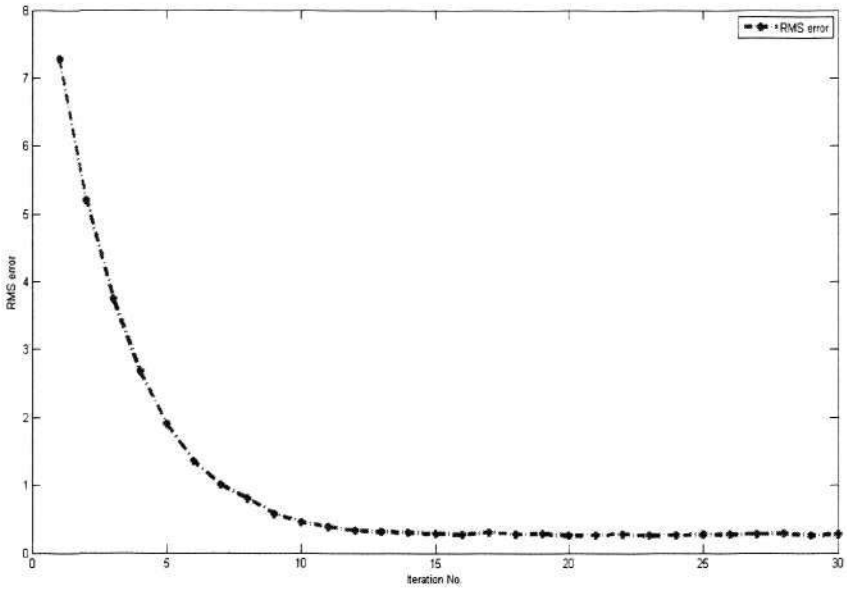


Figure 4.14 : RMS error profile of A-type ILC

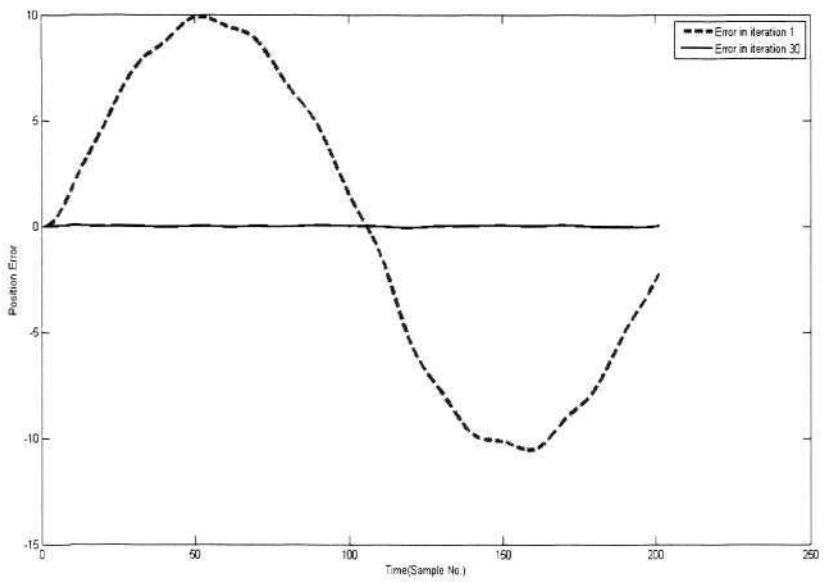


Figure 4.15 : Error profile of P-type ILC at iterations 1 and 30

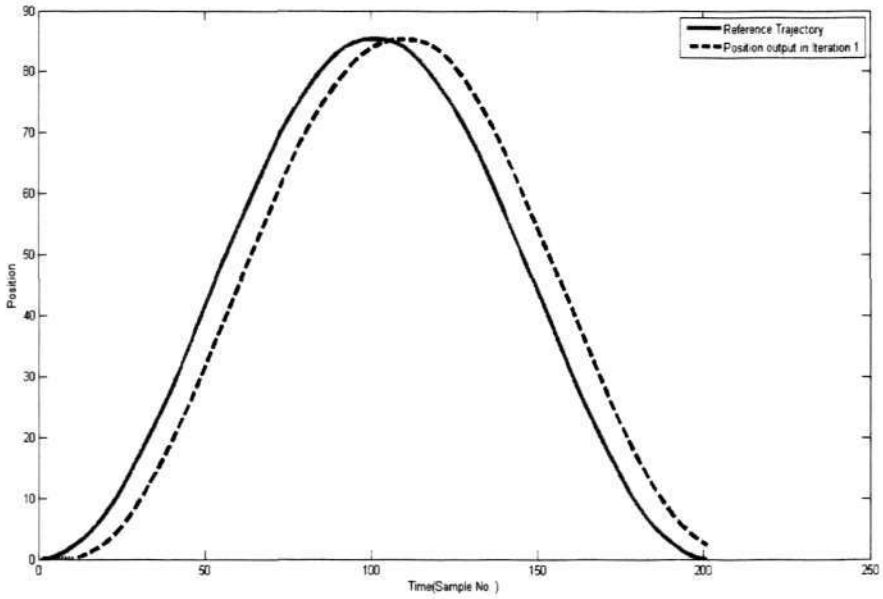


Figure 4.16 : Input and output of A-type ILC at iteration 1

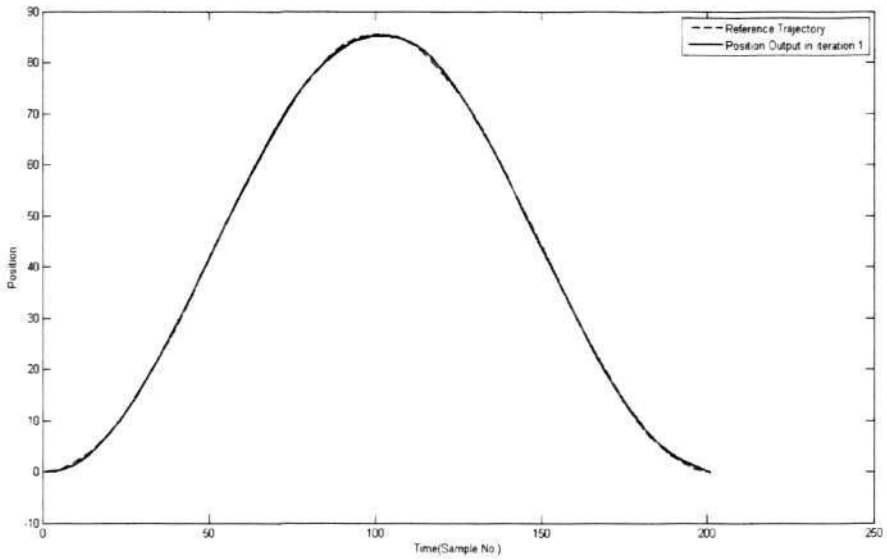


Figure 4.17 : Input and output of A-type ILC at iteration 30

4.5 Concluding remarks

Figure 4.3 shows the output of a PID controller. It depicts that, the tracking error in each of the iteration is same. In other words, figure 4.3 demonstrates that there is no learning in feedback controllers. As iteration number increases, results in figures 4.10 and 4.14 show reduction in tracking error. Hence, convergence of tracking error is exhibited by both P-type and A-type ILC. Comparing the results of P-type and A-type ILC laws, we observe that after 30 iterations the error profile is smooth and almost zero in case of A-type ILC. Because of use of causal pair of input applied and output obtained, A-type ILC shows better performance. Tracking accuracy can be improved by tuning the learning gain. The results demonstrate that ILC requires minimal knowledge about the plant/system as it uses concept of learning. Performance of ILC algorithm can be improved by using filters. ILC offers higher flexibility to the control process.

Chapter 5

ILC design for MIMO systems

This chapter focuses on iterative learning control for multi-input multi-output (MIMO) systems. The design and implementation of ILC laws on MIMO systems are explicitly depicted. To begin with, a description of a general MIMO system is presented. Related research works in field of ILC for MIMO systems are briefly surveyed. Convergence and robustness conditions for ILC design for MIMO systems are investigated both in time-domain and frequency-domain.

Basic concepts and objectives of decoupling technique are explained. The use of decoupling technique with ILC is described. The merits of using ILC with decoupling technique are illustrated. Two different types of decoupling approaches with ILC are formulated on MIMO systems. The results are presented and advantages of this approach are explained.

5.1 Motivation

Many real world systems have *multiple interacting* variables. Systems with more than one input and/or more than one output are known as MIMO systems. Every input can affect more than one output and every output can be influenced by more than one input. Design of effective controllers for MIMO systems is a *challenging* task, because of inherent interactions between input and output variables. Good controllers for MIMO systems should use available information from all the plant outputs and modify the plant inputs accordingly, in response to a reference input or disturbance. The challenge presented in controlling a MIMO system provides a *driving force* to design ILC on MIMO system.

Many techniques that can *compensate* for undesired cross-couplings can improve control system performance. However, it is impossible to eliminate the cross couplings completely. One alternate is to *compensate* for the noted cross-coupling effects by using decoupling technique.

Compensation for the effect of *cross-couplings* provides motivation to use decoupling technique with ILC. The main objective in decoupling is to balance for the effect of undesirable interactions brought about by *cross-couplings* of the process variables. This can be achieved by introducing an *additional* transfer function (*decoupling block*) block in control system design. In ideal case, a *decoupler* forces a MIMO system to act as a group of independent subsystems. Thus, decoupling technique can simplify the controller design process with the use of multiple independent SISO controllers for control loops in a MIMO system, instead of using a single MIMO controller with all the inherent interactions.

5.2 Related works

5.2.1 ILC for MIMO systems

In [44], Wang and Ye discussed ILC approaches for MIMO time-invariant systems. The analysis was carried out in frequency domain. Two non-causal filtering based ILC approaches were proposed. ILC laws were based on *reverse-time* filtering with no use of high order derivatives. Based on batch update feature of ILC, learning compensator was designed such that compensator is causal. Approach followed was based clean system inversion ILC, with use of system inverse model. This causality condition eliminates use of high order derivatives and numerical differentiation. Experiments were carried out an on an industrial robot system. This literature demonstrated that proposed learning control is less-noisy and applicable to practical situations.

In [41], a sufficient condition for stability of linear time-varying MIMO learning control system was investigated. Necessary conditions for *stability* of MIMO learning system were presented using Lyapunov theory. Learning law stability was analyzed in time-domain. Merits of learning control, such as robustness to model inaccuracy and drift in system parameters were explained with substantial proof. In [43], Tan *et al.* developed a necessary and sufficient condition for convergence of ILC for MIMO linear time-varying systems. Discrete-time learning controller is developed and convergence conditions of learning control are derived. It shown that convergence condition is guaranteed, if input/output coupling matrices at any point of time, are at full column rank. In [42], Xu and Tan proposed a new non-linear ILC scheme called the *Newton-type* ILC, for nonlinear non-affine dynamic MIMO systems. This paper presented a quantitative survey on importance of learning performance indices, such as convergence factor and convergence order. This paper effectively proves that, this approach can improve ILC performance when compared with linear-type ILC.

Both linear-type ILC approach and Newton-type were integrated to improve the performance of system. In [46], ILC is developed for MIMO systems with less model knowledge. To design an effective ILC, some *priori* knowledge about system is necessary. But in some cases, it is difficult to obtain prior knowledge of system. This note presents an ILC approach for such systems with less model knowledge. It is based on the concept of Nussbaum gain. Proposed ILC approach exhibited efficient tracking error convergence. ILC scheme was built using a Nussbaum-type gain selector and a refined compensator learned through repetitive tracking.

In [47-48], ILC design for MIMO-LTI systems is discussed. Two particular ILC schemes are considered and analyzed in both frequency and time domains.

Scheme 1: Tracking errors of current iteration

$$u_k(t) = K_1 e_k(t) + \theta_k(t) \tag{5.1}$$

$$\theta_k(t) = \theta_{k-1}(t) + K_2 e_k(t)$$

Scheme 2: Tracking errors of previous iteration

$$u_k(t) = K_1 e_k(t) + \theta_k(t) \tag{5.2}$$

$$\theta_k(t) = \theta_{k-1}(t) + K_2 e_{k-1}(t)$$

Convergence, implementation, robustness of learning control algorithm with respect to a linear time-invariant system was discussed. This literature presented discussions regarding robustness of ILC against reinitialization errors, positive realness issues and disturbances. It also showed substantial proof that design of convergent ILC approaches for MIMO-LTI system is straightforward and does not require knowledge about the system parameters, if the system is extended strict positive realness (ESPR) system.

In [62], an ILC approach with initial rectifying action for nonlinear continuous multivariable systems was proposed. D-type ILC approach was proposed such that, MIMO system based controller is robust to random fluctuations of initial outputs at different iterations. The convergence analysis was performed in time-domain. It generates an initial rectifying action on a minute initial time interval, and the reference trajectory tracking is pushed beyond the initial time interval. The tracking error beyond the initial time interval is driven to a residual set. The size of the residual set is dependent on the estimation error of the input matrix. This paper showed that proposed ILC approach exhibited better robustness than the conventional ILC approaches.

5.2.2 Decoupling Technique

In [64], Wenjian presented constructive concepts on design of controllers for *multivariable* systems. The objective of this note is to design a true multivariable controller that acts on all system outputs and modifies control inputs in response to reference input. Techniques for developing MIMO controllers based on decoupling are also discussed. Concepts relevant to *decoupler* design are presented based on ***dynamic decoupling, steady-state decoupling or partial decoupling***. Dynamic decoupling eliminates cross-couplings from all loops at every instant of time and steady-state decoupling eliminates steady state interactions. Partial decoupling is based on critical loops, in which interactions are strongest. It works on a subset of control loops. Two different types of decoupling approaches, namely, generalized decoupling and simplified decoupling is formulated. Merits and demerits of both approaches are described. Stability and causality conditions of *decouplers* are analyzed. Literature is extended to develop advanced decoupling techniques such as *inverted decoupling* and *normalized decoupling*.

5.3 MIMO system description

5.3.1 Time-domain representation

Continuous-time system representation

The time-domain representation of continuous time-varying MIMO system is

$$\dot{X}(t) = A(t)X(t) + B(t)u(t) + w(t) \tag{5.3}$$

$$y(t) = C(t)X(t) + D(t)u(t) + v(t) \tag{5.4}$$

where *u(t)* is *r*x*1 dimensional* input vector ,*y(t)* is *p*x*1 dimensional* output vector and *X(t) is m*x*1 dimensional* state vector, with *m* being number of states. A(*t*), B(*t*), C(*t*) and D(*t*) time-variant matrices with dimensions *m*x*m,m*x*r,p*x*m and p*x*r* respectively. *w(t)* and *v(t)* are *r*x*1 and p*x*1* dimensional input and output disturbances, respectively.

The time-domain representation of continuous time-invariant MIMO system is

$$\dot{X}(t) = AX(t) + Bu(t) + w(t) \tag{5.5}$$

$$y(t) = C\,X(t) + Du(t) + v(t) \tag{5.6}$$

where A, B, C and D time-invariant matrices are *m*x*m,m*x*r,p*x*m and p*x*r* respectively.

$$u(t) = [u_1(t), u_2(t), \ldots . u_r(t)]^T \text{ and } y(t) = [y_1(t), y_2(t), \ldots . y_p(t)]^T \tag{5.7}$$

Discrete-time system representation

The discrete time representation of time-varying MIMO system is

$$x(k+1) = A(k)x(k) + B(k)u(k) + w(k) \tag{5.8}$$

$$y(k) = C(k)\,x(k) + D(k)u(k) + v(k) \tag{5.9}$$

If the number of samples in a iteration is *q*, then *k=1,2,3,...q*. Here *u(k)* is the *r*x*q dimensional* input vector ,*y(k)* is the *p*x*q dimensional* output vector and *x(k) is m*x*q dimensional* state vector, with *m* being number of state variables. *w(k)* and *v(k)* are *r*x*q and p*x*q* dimensional input and output disturbances, respectively. A(*k*), B(*k*), C(*k*) and D(*k*) in Eq 5.8 and Eq 5.9 are time-variant matrices with dimensions *m*x*m,m*x*r,p*x*m and p*x*r* respectively.

The discrete time representation of time-invariant MIMO system is

$$x(k+1) = Ax(k) + Bu(k) + w(k) \tag{5.10}$$

$$y(k) = C\,x(k) + Du(k) + v(k) \tag{5.11}$$

5.3.2 Laplace domain

Continuous-time MIMO system can also be described in Laplace domain.

Using Eq. 5.5 and Eq. 5.6, Laplace transform equation can be expressed as,

$$Y(s) = G_p(s)U(s) + C(sI - A)^{-1}X(0) + C(sI - A)^{-1}W(s) + V(s) \tag{5.12}$$

where $Y(s)$ is $px1$ *dimensional* output matrix, $G_p(s)$ pxr *dimensional* is transfer function matrix of the MIMO plant & $U(s)$ is $rx1$ *dimensional* input matrix. $X(0), W(s)$ *and* $V(s)$ are initial state matrix, Laplace transform of input disturbance and Laplace transform of output disturbance.

5.3.3 Coupled representation

MIMO system can also represented as,

$$y_i(t) = f_i\big(u_1(t), u_2(t), \ldots . u_r(t)\big), \quad \textit{where } i = 1,2,\ldots .p$$

Controller design for MIMO systems is basically performed by choosing best value of u_j, *where* $j = 1,2,\ldots .r$. r is number of inputs and p is number of outputs

Let us define the error as,

$$\varepsilon_i = y_{di} - y_i \tag{5.13}$$

If there exists a controller g_{cj} to calculate each input u_j, an effective controller must decide the value of u_j from the entire error set $[\varepsilon_1, \varepsilon_2, \ldots . \varepsilon_p]$. The control input to be applied can be formulated as

$$u_j = f_j\big(\varepsilon_1, \varepsilon_2, \ldots . \varepsilon_p\big) \tag{5.14}$$

5.4 Objectives and formulation of decoupling approaches

5.4.1 Objectives

Cross-couplings and interactions in a MIMO system presents a *challenge* in designing a good controller. The basic objective of ILC approach followed here is to *overcome uncertainties* caused by *cross-coupling* and *inherent disturbances* in ILC based MIMO system. Aim of this scheme is to develop ILC algorithm in conjunction with *decoupling* technique. The principle of decoupling technique is to discover a suitable combination of input/output variables and construct a *decoupling* block, which can eliminate or reduce the control loop interactions. One of the important objectives to be accomplished is to simplify the ILC design process with use of *decoupling* technique.

5.4.2 Approach

The approach adopted here is to use *mathematical transformations* on combinations of process input and/or output variables and derive an equivalent system, which is free from inherent interactions. The intention is to develop a *decoupler* and introduce it in the control loop, such that the combination of plant transfer function and decoupling block generates a *decoupled* system, free from interactions (or with minimum interactions). In other words, *decoupler* design aims to obtain *decoupled* systems. The purpose is to apply ILC to *decoupled* system instead of actual system.

Consider a continuous-time MIMO system expressed as,

$$Y(s) = G_p(s)U(s) \tag{5.15}$$

The problem is to develop additional decoupling transfer function block *F(s)* and implement ILC on decoupled system $G_{total}(s)$ (figure 36). We have to compute the transfer function matrix *F(s)* so as to compensate interactions between loops.

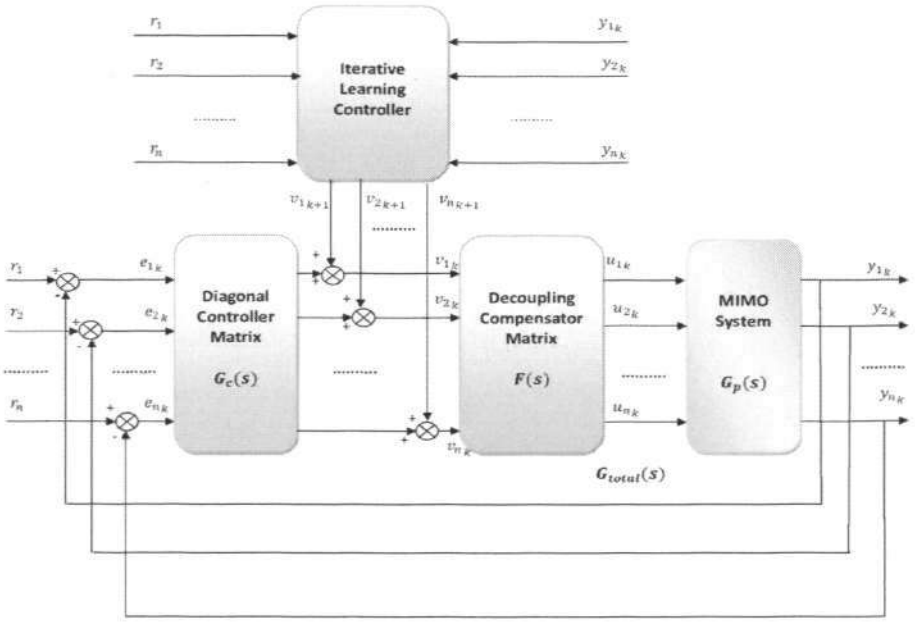


Figure 5.1 : Decoupling block diagram

Referring to figure 5.1, control input and output can be expressed as,

$$U = Fv \qquad (5.16) \qquad \text{and} \qquad Y = G_p F\, v \qquad (5.17)$$

where v & F is modified control input and decoupling matrix, respectively. Hence,

$$G_{total} = G_p F \qquad (5.18) \qquad \text{and} \qquad Y = G_{total}\, v \qquad (5.19)$$

We have to determine matrix F such that resultant matrix G_{total} is free from interactions. In other words, G_{total} is a diagonal matrix. Decoupling produces a *decoupled* system with independent sub-systems. Thus, by using decoupling technique, we can design SISO ILC laws for independent sub-systems. Problem is to derive convergence and robustness conditions of ILC for decoupled system G_{total} with v as modified control input. P-type update law should be designed as per equation,

$$v_{k+1} = v_k(t) + L(.)e_k(t) \qquad (5.20)$$

$$\begin{bmatrix} v_{1(k+1)}(t) \\ v_{2(k+1)}(t) \\ \vdots \\ v_{n(k+1)}(t) \end{bmatrix} = \begin{bmatrix} v_{1k}(t) \\ v_{2k}(t) \\ \vdots \\ v_{nk}(t) \end{bmatrix} + \begin{bmatrix} l_{11} & \cdots & l_{1n} \\ l_{21} & & l_{2n} \\ \vdots & \ddots & \vdots \\ l_{n1} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} y1_d - y_{1k} \\ y2_d - y_{2k} \\ \vdots \\ yd_n - y_{nk} \end{bmatrix} \qquad (5.21)$$

5.5 ILC design for MIMO systems: A description

Concepts of ILC can be explained in time-domain and frequency-domain. In [63], Bien and Xu presented detailed explanation on analysis, design, integration and applications of ILC. Theory and experimental results on ILC design for MIMO systems are also enlightened.

5.5.1 Tracking error convergence in time-domain

Consider a continuous time MIMO system which is represented as,

$$\dot{x}_k(t) = A(t)x_k(t) + B(t)u_k(t) \tag{5.22}$$

$$y_k(t) = C(t)x_k(t)$$

where k is iteration number. To simplify discussion, P-type ILC law is used to explain ILC design on MIMO system $\boldsymbol{G_p(s)}$. P-type update law is expressed as,

$$u_{k+1}(t) = u_k(t) + L(e_k(t)) \tag{5.23}$$

$$u_k = \begin{bmatrix} u_{1_k} \\ u_{2_k} \\ \vdots \\ u_{r_k} \end{bmatrix}, y_k = \begin{bmatrix} y_{1_k} \\ y_{2_k} \\ \vdots \\ y_{p_k} \end{bmatrix} \quad and \quad L = \begin{bmatrix} l_{11} & \cdots & l_{1p} \\ l_{21} & & l_{2p} \\ \vdots & \ddots & \vdots \\ l_{r1} & \cdots & l_{rp} \end{bmatrix} \tag{5.24}$$

$$\begin{bmatrix} u_{1_{(k+1)}}(t) \\ u_{2_{(k+1)}}(t) \\ \vdots \\ u_{r_{(k+1)}}(t) \end{bmatrix} = \begin{bmatrix} u_{1_k}(t) \\ u_{2_k}(t) \\ \vdots \\ u_{r_k}(t) \end{bmatrix} + \begin{bmatrix} l_{11} & \cdots & l_{1p} \\ l_{21} & & l_{2p} \\ \vdots & \ddots & \vdots \\ l_{r1} & \cdots & l_{rp} \end{bmatrix} \begin{bmatrix} y1_d - y_{1_k} \\ y2_d - y_{2_k} \\ \vdots \\ yp_d - y_{p_k} \end{bmatrix} \tag{5.25}$$

where $\boldsymbol{t} \in [\boldsymbol{0}, \boldsymbol{t_f}]$, $u_k(t)$ *and* $y_k(t)$ are rx1 input and px1 output resp., L is rxp learning matrix. $e_k(t)$ is px1 dimensional error matrix. Referring to Eq. B.28 from appendix B,

$$\|\boldsymbol{e_{k+1}(t)}\|_\lambda \leq \boldsymbol{\rho}\|\boldsymbol{e_k(t)}\|_\lambda, \ \ when\ \lambda - norm\ is\ applied \tag{5.26}$$

Substantial proof can be established in time domain that, if $\rho < 1$, Eq 5.26 exhibits effective convergence of tracking error. We can select the value of λ to be sufficiently large , in order to guarantee that $\rho < 1$

5.5.2 Tracking error convergence in frequency-domain

Continuous-time MIMO system can be described in Laplace domain as,

$$Y(s) = G_p(s)U(s) \tag{5.27}$$

where $Y(s)$ is px1 output matrix, $G_p(s)$ pxr is transfer function matrix of MIMO plant & $U(s)$ is rx1 input matrix, r & p being number of inputs and number of outputs.

In Laplace domain P-type ILC update equation can be expressed as,

$$U_k(s) = U_{k-1}(s) + L(s)E_{k-1}(s) \tag{5.28}$$

We know that, $$E_k(s) = Y_d(s) - Y_k(s) \tag{5.29}$$

$$Y_k(s) - Y_{k-1}(s) = G_p(s)\big(U_k(s) - U_{k-1}(s)\big) \tag{5.30}$$

Rearranging Eq 5.28, $$U_k(s) - U_{k-1}(s) = L(s)E_{k-1}(s) \tag{5.31}$$

Substituting Eq 5.31 in Eq 5.30,

$$Y_k(s) - Y_{k-1}(s) = G_p(s)L(s)E_{k-1}(s) \tag{5.32}$$

Also, $$E_k(s) = Y_d(s) - Y_k(s) \tag{5.33}$$

$$E_{k-1}(s) = Y_d(s) - Y_{k-1}(s) \tag{5.34}$$

Subtracting Eq 5.33 from Eq 5.34,

$$Y_k(s) - Y_{k-1}(s) = -\big(E_k(s) - E_{k-1}(s)\big) \tag{5.35}$$

Equating equations 5.35 and 5.32 we get,

$$E_k(s) = \big(1 - L(s)G_p(s)\big)E_{k-1}(s) \tag{5.36}$$

Error in the current iteration should be less than error in previous iteration. Chosen learning gain matrix $\boldsymbol{L}$ will cause the tracking error to converge to zero if and only if the following convergence condition is satisfied.

$$\|I - L(j\omega)G_p(j\omega)\| < 1 \tag{5.37}$$

The frequency-domain convergence condition is a *sufficient* condition for convergence, though ILC is a finite time problem.

5.6 Two-input two output MIMO system: Convergence and robustness analysis with decoupling technique

In this section, convergence and robustness analysis of two-input two-output MIMO system without decoupling are depicted. Design techniques for the design decoupler for a MIMO system are formulated. Two different types of decoupling technique, namely *generalized decoupling and simplified decoupling* are illustrated. Convergence and robustness analysis is performed in frequency domain on decoupled system.

5.6.1 MIMO system description

Consider a two-input two-output continuous time MIMO system,

$$Y(s) = G_p(s)U(s)\ , \tag{5.38}$$

$$where\ G_p(s) = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix}$$

In this discussion, we assume the desired trajectory is of the form,

$$y_d(t) = 80e^{-2\pi t}(1 - \cos 2\pi t) \quad \text{[figure 5.2]}$$

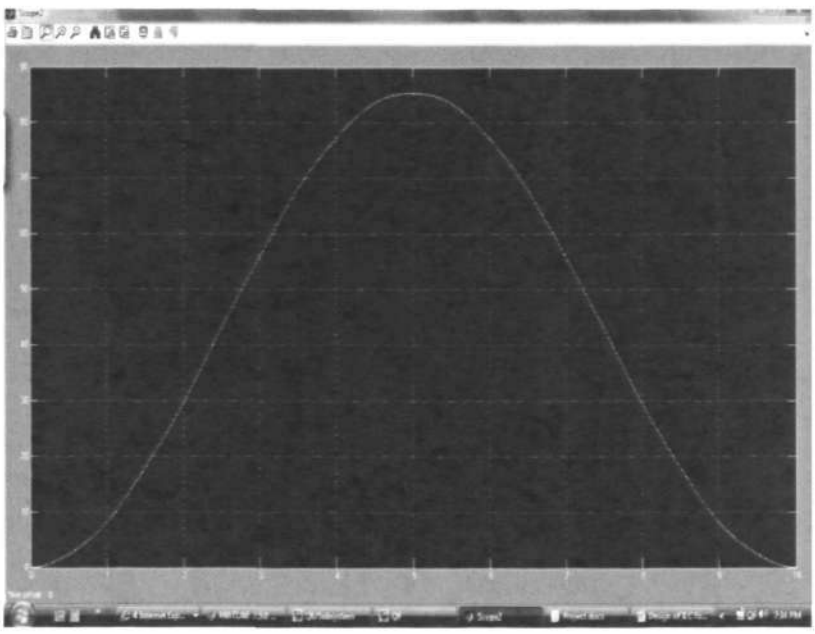


Figure 5.2 : Desired trajectory in MIMO ILC development

5.6.2 Convergence and robustness analysis without Decoupling

P-type ILC update law for MIMO system described in Eq 5.38 is,

From Eq 5.25,
$$\begin{bmatrix} u_{1_{(k+1)}}(t) \\ u_{2_{(k+1)}}(t) \\ \vdots \\ u_{r_{(k+1)}}(t) \end{bmatrix} = \begin{bmatrix} u_{1_k}(t) \\ u_{2_k}(t) \\ \vdots \\ u_{r_k}(t) \end{bmatrix} + \begin{bmatrix} l_{11} & \cdots & l_{1p} \\ l_{21} & & l_{2p} \\ \vdots & \ddots & \vdots \\ l_{r1} & \cdots & l_{rp} \end{bmatrix} \begin{bmatrix} y1_d - y_{1_k} \\ y2_d - y_{2_k} \\ \vdots \\ yp_d - y_{p_k} \end{bmatrix} \tag{5.39}$$

$$\begin{bmatrix} u_{1_{(k+1)}} \\ u_{2_{(k+1)}} \end{bmatrix} = \begin{bmatrix} u_{1_k} \\ u_{2_k} \end{bmatrix} + \begin{bmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} y_d(t) - y_{1_k}(t) \\ y_d(t) - y_{2_k}(t) \end{bmatrix} \tag{5.40}$$

From Eq 5.37, the convergence condition of a MIMO is expressed as,

$$\left\| \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{pmatrix} \begin{pmatrix} G_{11}(j\omega) & G_{12}(j\omega) \\ G_{21}(j\omega) & G_{22}(j\omega) \end{pmatrix} \right\| < 1 \tag{5.41}$$

The above inequality can be simplified to the following equations,

$$|1 - (l_{11}G_{11}(j\omega) + l_{12}G_{21}(j\omega))| < 1 \tag{5.42}$$

$$|l_{11}G_{12}(j\omega) + l_{12}G_{22}(j\omega)| < 1 \tag{5.43}$$

$$|l_{21}G_{11}(j\omega) + l_{22}G_{21}(j\omega)| < 1 \tag{5.44}$$

$$|1 - (l_{21}G_{12}(j\omega) + (l_{21}G_{22}(j\omega)| < 1 \tag{5.45}$$

We choose a diagonal learning gain matrix for our discussion in this section, i.e, $l_{12} = l_{21} = 0$. Splitting Eq 5.42 & 5.45 in magnitude and phase form and simplifying,

$$\boldsymbol{l_{11}N^1{}_p(\omega) < 2\cos\theta^1{}_p(\omega)} \text{ and } \boldsymbol{\theta^1{}_p(\omega) < 90^0}$$

$$\boldsymbol{l_{22}N^2{}_p(\omega) < 2\cos\theta^2{}_p(\omega)} \text{ and } \boldsymbol{\theta^2{}_p(\omega) < 90^0} \tag{5.46}$$

$N^1{}_p(\omega)$ and $N^2{}_p(\omega)$ are magnitude responses of $G_{11}(s)$ and $G_{22}(s)$ respectively. $\theta^1{}_p(\omega)$ and $\theta^2{}_p(\omega)$ are phase responses of $G_{11}(s)$ and $G_{22}(s)$ respectively.

Robustness analysis: In order ensure robustness against uncertainties, we introduce an additional term $\varepsilon = 10^0$ in Eq 5.46, such that

$$\boldsymbol{l_{11}N^1{}_p(\omega) < 2\cos\theta^1{}_p(\omega)} \text{ and } \boldsymbol{\theta^1{}_p(\omega) < 90^0 - \varepsilon}$$

$$\boldsymbol{l_{22}N^2{}_p(\omega) < 2\cos\theta^2{}_p(\omega)} \text{ and } \boldsymbol{\theta^2{}_p(\omega) < 90^0 - \varepsilon} \tag{5.47}$$

Considering Eq 5.43 and 5.44

$$|l_{11}G_{12}(j\omega)| < 1 \quad and \quad |l_{22}G_{21}(j\omega)| < 1 \tag{5.48}$$

$$|l_1 l_2 G_2(j\omega)G_3(j\omega)| < 1 \tag{5.49}$$

Using Eq 5.49, Eq 5.47 and bode plots (fig 5.3, 5.4), we calculate values of $l_{11} and\ l_{22}$.

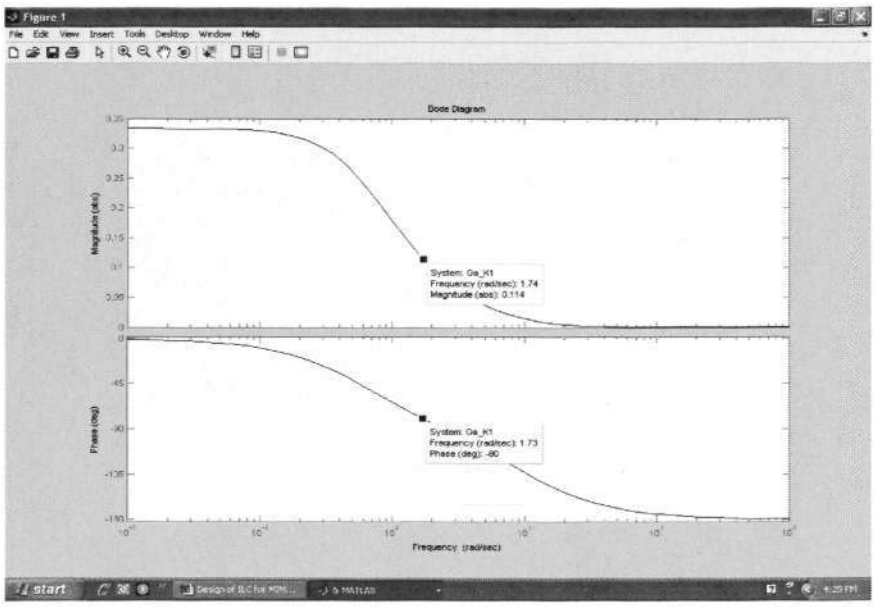


Figure 5.3 : Bode plot for MIMO ILC to compute value of l_{11}

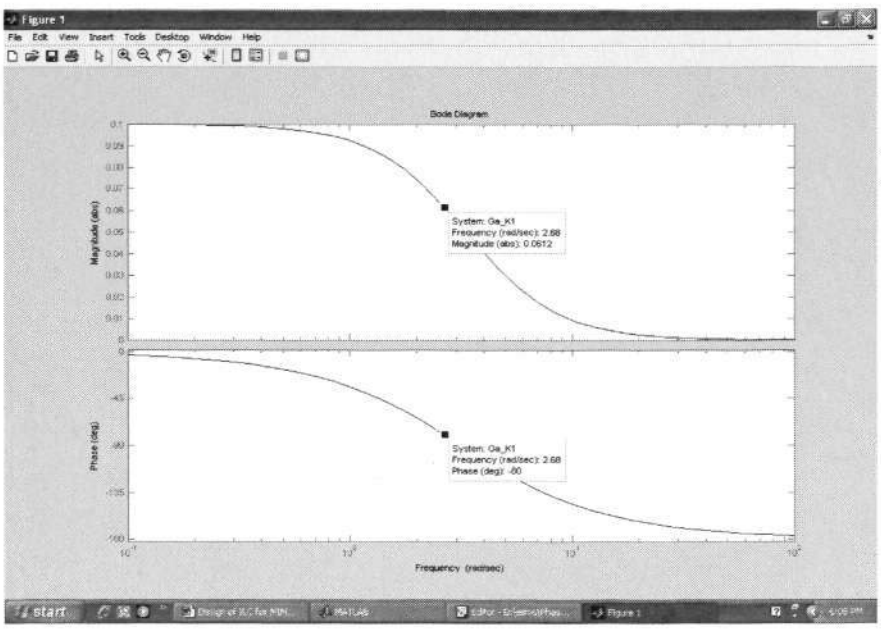


Figure 5.4 : Bode plot for MIMO ILC to compute value of l_{22}

5.6.3 Convergence and robustness analysis with decoupling

In this thesis, we focus on *dynamic decoupling*. Dynamic decoupling eliminates cross-couplings from all loops at every instant of time. Firstly, *decoupler* is designed by using generalized and simplified decoupling. Based on design, decoupled system is developed such that it does not contain interactions. ILC is designed on decoupled system and convergence and robustness analysis is performed in frequency domain.

5.6.3 a Generation of decoupling block (decoupler)

Generalized decoupling

Generalized decoupling [64] technique is a diagonal decoupling control system. Generalized decoupling technique is based on assumption that the decoupled system is diagonal of the transfer function matrix. The generalized *decoupler* is developed in the same way as mentioned in **[64].** Therefore,

$$G_{total} = \begin{bmatrix} G_{11} & 0 \\ 0 & G_{22} \end{bmatrix} \tag{5.50}$$

From Eq 5.38, $$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \Rightarrow Y = G_p U \tag{5.51}$$

The control input can be modified as,

From Eq 5.51 , $$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \Rightarrow U = Fv \tag{5.52}$$

$where\ F$ is decoupling matrix and where v is modified control input.

Hence, $$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \Rightarrow Y = G_p F v \tag{5.53}$$

Therefore, $$G_{total} = G_p F \tag{5.54}$$

$$Y = G_{total}\, v \tag{5.55}$$

$$F = G_p^{-1} G_{total} \quad and \quad G_{total} = \begin{bmatrix} G_{11} & 0 \\ 0 & G_{22} \end{bmatrix} \quad [\mathbf{64}] \tag{5.56}$$

From Eq 5.56, decoupler F can be calculated, since $G_p\ and\ G_{total}$ are known.

Simplified decoupling

Simplified decoupling **[64]** generates a simplified decoupling matrix compared to generalized case. It uses a *decoupler* with g_{I1} and g_{I2} (figure 5.5). Here, v_1 and v_2 are controller outputs and u_1 and u_2 are actual control input.

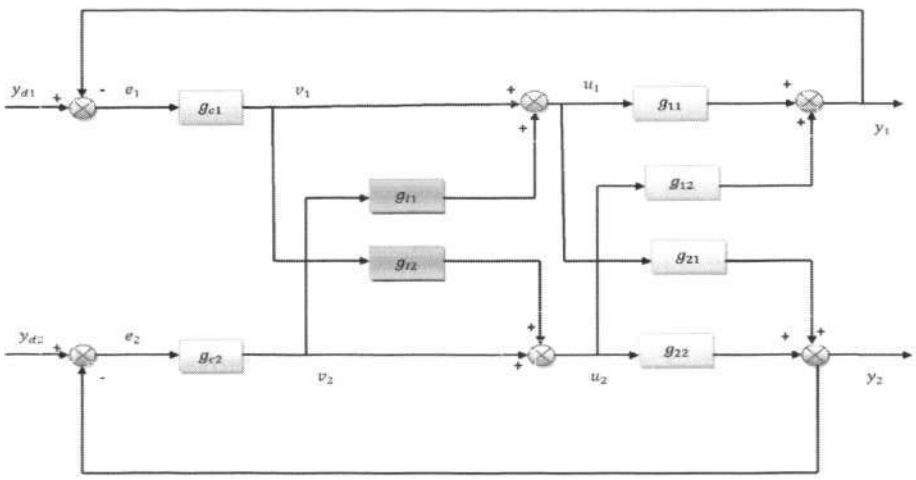


Figure 5.5 : Simplified decoupling block diagram [64]

The equations governing the control action are,

$$u_1 = v_{1+} g_{I1} v_2 \quad \textbf{and} \quad u_2 = v_{2+} g_{I2} v_2 \tag{5.57}$$

$$y_1 = (g_{11} + g_{I2} g_{12}) v_1 + (g_{12} + g_{11} g_{I1}) v_2 \qquad \textbf{[64]} \tag{5.58}$$

$$y_2 = (g_{21} + g_{I2} g_{22}) v_2 + (g_{22} + g_{21} g_{I1}) v_1 \qquad \textbf{[64]} \tag{5.59}$$

According decoupling concept, v_1 should affect only y_1 and $v_2$should affect only y_2.

$$g_{12} + g_{11} g_{I1} = 0 \quad \textbf{and} \quad g_{22} + g_{21} g_{I1} = 0 \tag{5.60}$$

$$g_{I1} = -\frac{g_{12}}{g_{11}} \quad \textbf{and} \quad g_{I2} = -\frac{g_{21}}{g_{22}} \qquad \textbf{[64]} \tag{5.61}$$

Substituting Eq 5.61 in 5.59 and 5.58,

$$y_1 = \left(g_{11} - \frac{g_{12} g_{21}}{g_{22}}\right) v_1 \;\; and \;\; y_2 = \left(g_{22} - \frac{g_{12} g_{21}}{g_{11}}\right) v_2 \qquad [64] \tag{5.62}$$

$$\boldsymbol{Y = G_{total} * v} \quad \textbf{and} \quad \boldsymbol{F = \begin{bmatrix} 0 & g_{I1} \\ g_{I2} & 0 \end{bmatrix}} \qquad \textbf{[64]} \tag{5.63}$$

where, $G_{total} = \begin{bmatrix} G_{m1} & 0 \\ 0 & G_{m2} \end{bmatrix}$, $G_{m1} = g_{11} - \frac{g_{12} g_{21}}{g_{22}}$ *and* $G_{m2} = g_{22} - \frac{g_{12} g_{21}}{g_{11}}$ [64]

5.6.3 b Advantage of decoupling approaches

Important attribute of ILC is selection of learning gain matrix based on system model.

Typical structure of learning gain matrix L

- Diagonal L or Diagonal L matrix with non-zero diagonal elements shifted towards the upper right corner producing a linear phase lead.
- Two learning gain controllers with L having a diagonal and a sub-diagonal.
- Band of diagonals with the center of band shifted towards upper right in matrix
- Upper triangular block, lower triangular block or L can be a full matrix of gains

From Eq. 5.42- Eq. 5.45, we have

$$|1 - (l_{11}G_{11}(j\omega) + l_{12}G_{21}(j\omega))| < 1 \quad and \; |l_{11}G_{12}(j\omega) + l_{12}G_{22}(j\omega)| < 1$$

$$|l_{21}G_{11}(j\omega) + l_{22}G_{21}(j\omega)| < 1 \; and \quad |1 - (l_{21}G_{12}(j\omega) + (l_{21}G_{22}(j\omega)| < 1$$

Solving these equations to obtain learning gain values is cumbersome. To simplify convergence analysis, in most cases a diagonal learning gain matrix is assumed. Other assumptions such as, $2l_{12} = l_{11}$ and $2l_{21} = l_{22}$ are also used. But, in all the cases selection of learning gain matrix is random or based on assumptions. There is a need for a *general approach* to choose learning gain matrix. From Eq 5.55, 5.56 and 5.63,

$$\boldsymbol{Y = G_{total}v}$$

$$\boldsymbol{G_{total}} = \begin{bmatrix} \boldsymbol{G_{11}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{G_{22}} \end{bmatrix} \; (Generalized) \qquad \boldsymbol{G_{total}} = \begin{bmatrix} \boldsymbol{G_{m1}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{G_{m2}} \end{bmatrix} \; (Simplified)$$

Using decoupling approaches, *decoupled* system is obtained. In this case, we have to compute only diagonal learning gain values, i.e, $l_{11}$and l_{22}. Off-diagonal elements are eliminated, because of use of decoupler. This method can be extended all MIMO systems. Hence a *general* approach for selecting learning gain matrix is formulated. Complexity involved in convergence and robustness analysis is reduced by this approach. Another advantage of this scheme is that independent SISO ILC laws can be applied to each of the sub-systems (control loops).

5.6.3 c Convergence and robustness analysis

In 5.6.3 a, decoupler and decoupled system was computed. The task now is to design P-type ILC law on decoupled system. From Eq 5.55, Eq 5.56 and Eq 5.63, we have,

$$Y = G_{total}\, v$$

For generalized decoupling, $G_{total} = Diag(G_p) = \begin{bmatrix} G_1 & 0 \\ 0 & G_2 \end{bmatrix}$ **[64]** (5.64)

$$F = G_p^{-1} G_{total}$$

$$G_{total} = \begin{bmatrix} G_{m1} & 0 \\ 0 & G_{m2} \end{bmatrix} \quad [64]$$

For simplified decoupling,

$$G_{m1} = g_{11} - \frac{g_{12}g_{21}}{g_{22}} \; and \quad G_{m2} = g_{22} - \frac{g_{12}g_{21}}{g_{11}}$$

$$F = \begin{bmatrix} 0 & g_{I1} \\ g_{I2} & 0 \end{bmatrix}, \; g_{I1} = -\frac{g_{12}}{g_{11}} \quad \textbf{and} \quad g_{I2} = -\frac{g_{21}}{g_{22}} \tag{5.65}$$

The decoupled system in both cases does not contain any cross-couplings. We now have to design an ILC for decoupled system G_{total}.

From equation 5.41, convergence condition for a MIMO system $G_p(s)$ is,

$$\|I - L(j\omega)G_p(j\omega)\| < 1 \tag{5.66}$$

when, $u(t)$ is the plant input. But, in this case, $G_p(s) = G_{total}(s)$ and $u(t) = v(t)$

P-type ILC update equation in this case is modified as,

$$v_k(s) = v_{k-1}(s) + L(s)E_{k-1}(s) \tag{5.67}$$

Here L is the learning gain matrix, which is assumed to be $L = \begin{bmatrix} l_1 & 0 \\ 0 & l_2 \end{bmatrix}$

Convergence condition in frequency domain, Eq 5.66, is modified as,

$$\|1 - L(j\omega)G_{total}(j\omega)\| < 1 \tag{5.68}$$

From Eq 5.68, we arrive at convergence conditions for decoupled system.

Convergence condition using generalized decoupling

$$|1 - l_1 G_1(j\omega)| < 1 \ \ and \ \ |1 - l_2 G_2(j\omega)| < 1 \tag{5.69}$$

Convergence condition using simplified decoupling

$$|1 - l_1 G_{m1}(j\omega)| < 1 \ \ and \ \ |1 - l_2 G_{m2}(j\omega)| < 1 \tag{5.70}$$

Splitting equations Eq 5.69 and Eq 5.70 in magnitude and phase form and simplifying,

Generalized decoupling

$$\boldsymbol{l_1 N^1{}_p(\omega) < 2\cos\theta^1{}_p(\omega)} \ \ and \ \ \boldsymbol{\theta^1{}_p(\omega) < 90^0 - \varepsilon} \tag{5.71}$$

$$\boldsymbol{l_2 N^2{}_p(\omega) < 2\cos\theta^2{}_p(\omega)} \ \ and \ \ \boldsymbol{\theta^2{}_p(\omega) < 90^0 - \varepsilon}$$

Simplified decoupling

$$\boldsymbol{l_1 N^{m1}{}_p(\omega) < 2\cos\theta^{m1}{}_p(\omega)} \ \ and \ \ \boldsymbol{\theta^{m1}{}_p(\omega) < 90^0 - \varepsilon} \tag{5.72}$$

$$\boldsymbol{l_2 N^{m2}{}_p(\omega) < 2\cos\theta^{m2}{}_p(\omega)} \ \ and \ \ \boldsymbol{\theta^{m2}{}_p(\omega) < 90^0 - \varepsilon}$$

where $N^1{}_p(\omega)$, $N^2{}_p(\omega)$, $N^{m1}{}_p(\omega)$ and $N^{m2}{}_p(\omega)$are magnitude responses of $G_1(s)$, $G_2(s)$, $G_{m1}(s)$ and $G_{m2}(s)$, respectively. $\theta^1{}_p(\omega)$, $\theta^2{}_p(\omega)$, $\theta^{m1}{}_p(\omega)$ and $\theta^{m2}{}_p(\omega)$ are phase responses of $G_1(s)$, $G_2(s)$, $G_{m1}(s)$ and $G_{m2}(s)$, respectively.

In order to ensure that the system is robust, $\boldsymbol{\varepsilon = 10^0}$

We use the magnitude and frequency response plots to analyze and choose learning gain values. Referring to the Eq 5.71, Eq 5.72 and using bode plots we obtain values of l_1 *and* l_2. The steps to obtain values of the parameters l_1 *and* l_2 are explained in the section **5.6.2,** but in this case the decoupled system $G_{total}(s)$ is used.

5.7 Results

MIMO ILC without decoupling

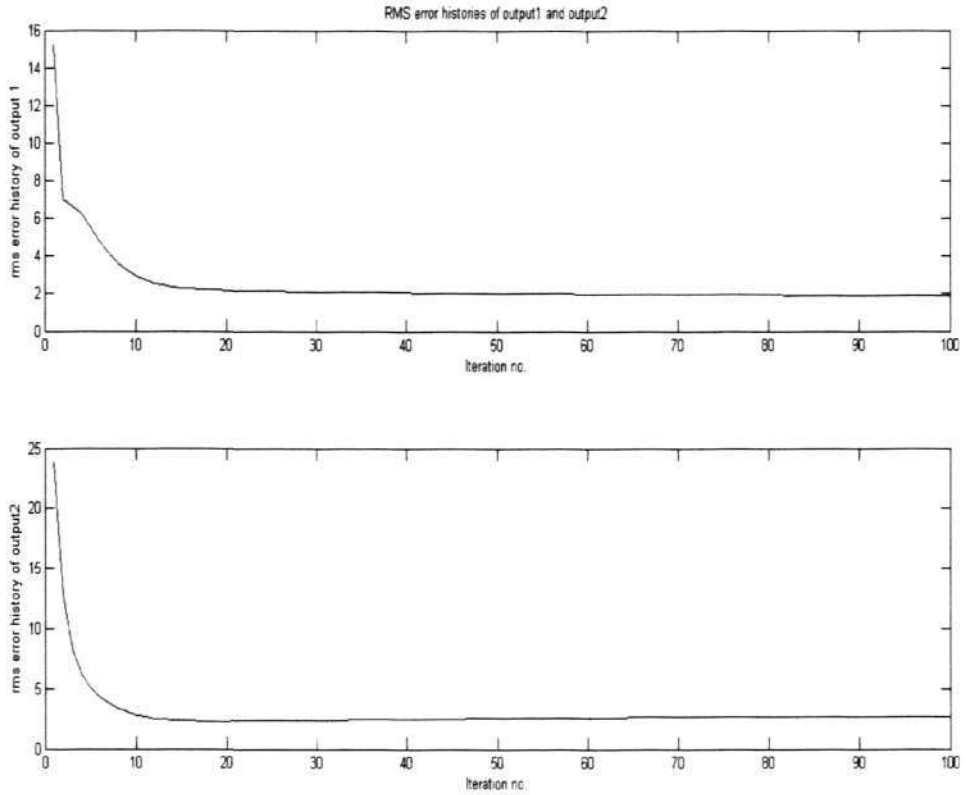


Figure 5.6 : Results for MIMO ILC without decoupling for 100 iterations

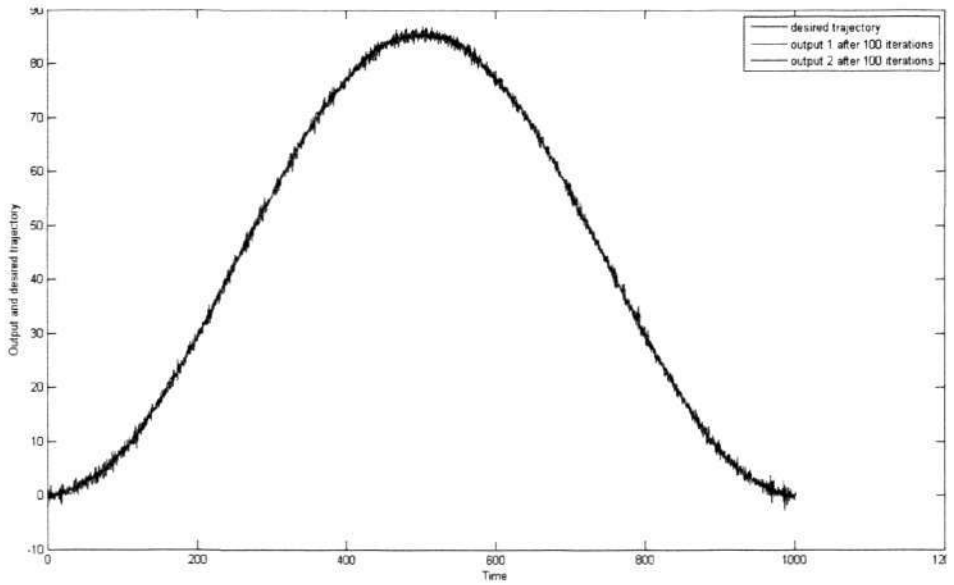


Figure 5.7 : Output 1, output 2 and desired trajectory for MIMO ILC after 100 iterations

MIMO ILC with generalized decoupling

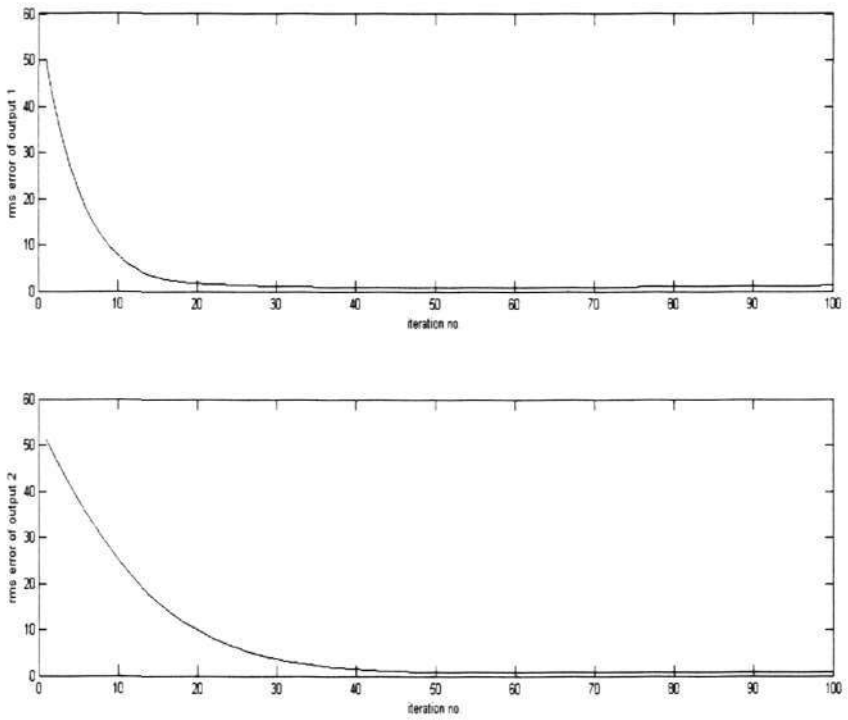


Figure 5.8 : Results for MIMO ILC with generalized decoupling for 100 iterations

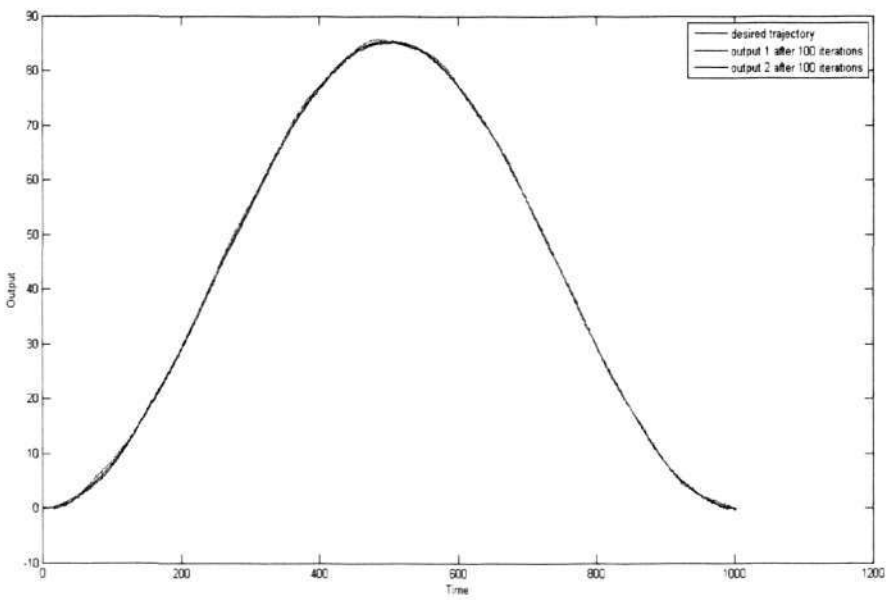


Figure 5.9 : Output 1, output 2 and desired trajectory for MIMO ILC with generalized decoupling after 100 iterations

MIMO ILC using simplified de-coupling

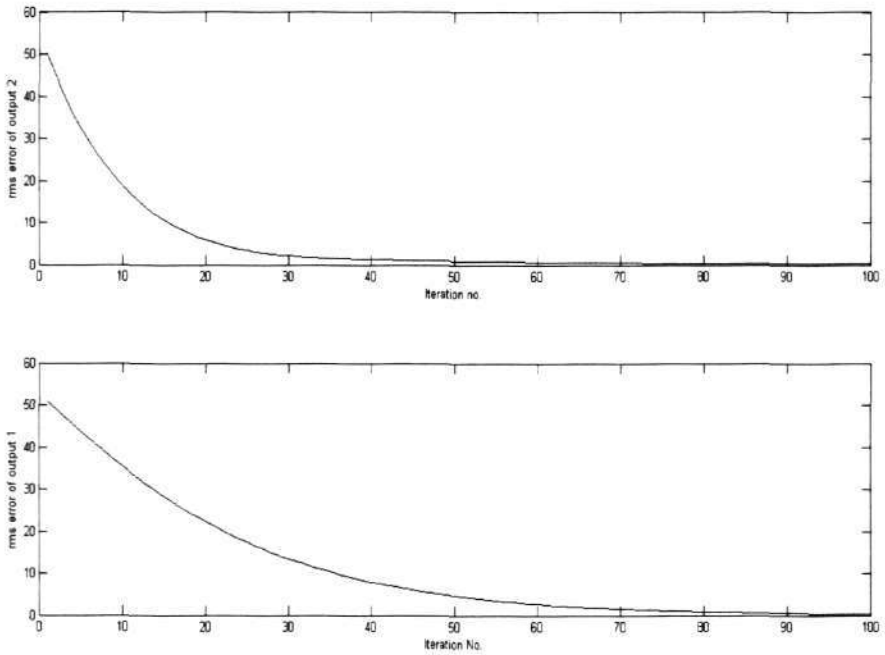


Figure 5.10 : Results for MIMO ILC with simplified decoupling for 100 iterations

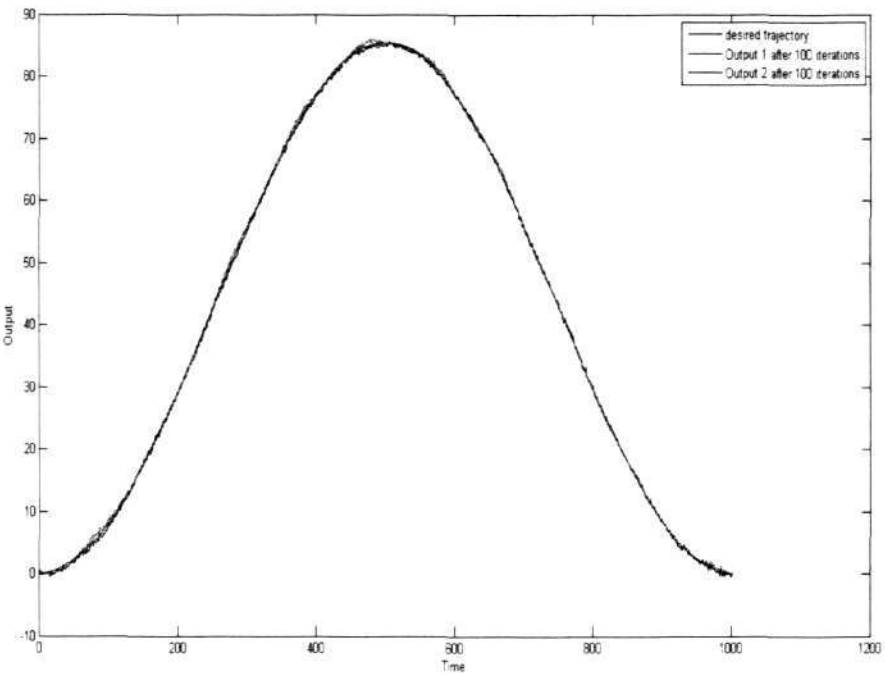


Figure 5.11 : Output 1, output 2 and desired trajectory for MIMO ILC with simplified decoupling after 100 iterations

5.8 Concluding remarks

Convergence and robustness conditions for MIMO systems are presented both in time-domain and frequency domain. A two-input two-output continuous-time MIMO system with disturbance was considered and ILC algorithms were developed. Two different types *decoupling* techniques (generalized decoupling and simplified decoupling) with ILC was demonstrated. Figures 5.6, 5.8, and 5.10 depict that RMS (root mean square) errors decreases from iteration to iteration. In other words, convergence of MIMO-ILC is shown in the results.

From Figures, 5.6 and 5.7, we can infer that ILC exhibits convergence. Due to the interactions in the MIMO system and based on the assumption that learning gain matrix is diagonal, decrease in RMS error after 20 iterations is minimal. Output is also distorted due to the interactions and output disturbance introduced. This can be improved by using suitable filters to produce a good learning transient or assuming a different structure of learning gain matrix. Selection of learning gain matrix has to be performed by trial and error.

From figures, 5.8 and 5.9, we can infer that, generalized decoupler produces better convergence of tracking error and output response is smooth. This is because of the fact, that interactions are completely eliminated and sub-systems (control loops) act independently. Simplified decoupler (fig 5.10 and 5.11) also shows good convergence but the convergence is slow for output 2 and output waveforms have little distortion. This behavior can be explained by using the decoupled system. In generalized case, decoupled system is the diagonal matrix of plant transfer function. Though, simplified decoupling uses a simpler *decoupler*, the diagonal elements of the plant transfer function are modified by *decoupler* parameters as mentioned in Eq 5.63.

Hence the performance of a simplified decoupler is not as effective as generalized case. Generalized decoupling can be extended to any number of inputs and outputs. Simplified decoupling is effective if the number of inputs and/or outputs is less than 3. Because, the process of mathematical calculations for higher number of input/output variables is cumbersome. Generalized decoupling is a clear choice in this case, as it involves calculation of a decoupling matrix, irrespective of the number of inputs/outputs.

Instead of using multiple single loop control ignoring the interactions between the single loops, decoupling offers flexibility to design. A *general approach* for selection of learning gain matrix in ILC is formulated using decoupling technique. Uncertainties due interactions in a ILC based MIMO system are reduced by using decoupling technique.

One important advantage of this scheme is that ILC and decoupling *supplement* each other. Decoupling technique is based on the fact that the model is nominal. If there is some discrepancy in model parameters, decoupling can give rise to erroneous results. If real world scenario, this condition can be violated by the presence of unpredictable disturbances. On the contrary, ILC can be designed using minimal knowledge about the system parameters. ILC can compensate for model inaccuracy due to uncertainties. Hence, control system performance is improved.

Chapter 6

System identification of continuous-time systems using ILC

6.1 Introduction

A dynamical mathematical model is a mathematical representation of the *dynamic behavior* of a system, plant or process. **System identification** is a universal phrase used in description of *mathematical tools* and algorithms to construct dynamical models from measurement or available data. The objective of system identification is to predict behavior of the system in the presence of external influences (inputs/disturbances of the system) and thereby attempt to determine a mathematical relation between the parameters of the system. System identification is potentially important for control system design, control automation, process/plant analysis, simulation, prediction, monitoring and system diagnosis.

System identification is performed in time -domain or frequency-domain. It can also be carried out in continuous-time or discrete-time. Most of the design procedures in system identification are suitable for continuous-time systems because capturing plant dynamics is relatively simpler in continuous-time domain. Generating continuous-time models directly from the sampled I/O data is more advantageous, in spite of the fact that most existing identification methods are described in discrete-time.

In this chapter, system identification using ILC is applied and implemented to T1 and T2 axis of the SCARA robot TT-3000. System identification using ILC approach is carried out based on projection of I/O data on finite dimensional sub-space.

6.1.1 Basic steps in system identification

1. Collection of experimental data set

Common approach is to start from measurements of available data from the system using experiments. Input-output data are recorded using an explicitly designed identification experiment. Experiment designed should obtain a data set, which can provide maximum information about the system. Design of experiment consists of attributes such as, choice of input signal, sampling period, initial conditions, signals to be measured, and amount of data required.

2. Determination of model structure

Generally, a certain model structure is chosen by the designer which contains unknown parameters. Choosing a suitable model structure is an important and difficult step in system identification. Physical parameters of the model can be built using basic physical laws and mathematical equations governing the system. We can also estimate a suitable model structure based on a *priori* knowledge and engineering intuition.

3. Procedure or a rule, by which model can be identified

A rule or a procedure is chosen such that the model can be evaluated using the data set. Commonly used methods are Least Square selection rule, frequency response identification rule, and other parametric and non-parametric methods.

4. Model Validation

Model validation is used to check the effectiveness of the parameters identified. After system identification, system should be validated using a chosen data set. It demonstrates whether the system behaves efficiently with the identified parameters.

6.2 Motivation

Motivation for designing and implementing system identification algorithms is based on the importance of system identification in learning control. System identification *interacts naturally* with learning control, because of the inherent property of learning through iterations. Learning controllers have a potential for achieving perfect tracking accuracy, with little knowledge about the system. On the contrary, if the model of system is completely determined by identification, there is no need for learning control as one can invert the identified model to generate the desired *feedforward* control input. But, perfect identification is not possible under practical circumstances. Knowledge about the system gained by system identification can help us design sufficiently accurate learning control algorithms. Although, the identified model may be inaccurate, it is often sufficiently accurate when used in conjunction with a learning controller. Thus, system identification supplies knowledge about the system, which can be useful to design of suitable learning controller. For high performance learning controllers, system identification can add significant value to the process of learning. These factors provide immense motivation to implement system identification algorithms.

Direct identification technique has an intricacy in handling the time-derivatives of I/O data in the presence of measurement noise. Sakai and Sugie [51,52,53,54] proposed a different method for identifying continuous-time systems using iterative learning control. System models developed using this sufficiently accurate and work on the concept of learning over iterations. Enthusiasm to perform system identification using iterative learning is derived from accuracy of the models obtained and robustness to disturbances.

6.3 System identification *via* iterative learning : Basic theory

The system identification *via* iterative learning is based on the approach developed by Sugie and Sakai[[51,52,53,54]. Approach is to identify continuous-time systems directly from the sampled I/O data based on trial or iterations. Input and output data is *projected* on to finite dimensional sub-space and identification approach uses *basis functions* for its approximation.

6.3.1 Objective of system identification

Consider the continuous time Single Input Single Output system described by

$$y(t) = \frac{B^0(p)}{A^0(p)}\,u(t) \triangleq \frac{\beta_0^0 + \beta_1^0 p + \cdots + \beta_m^0 p^m}{1 + \alpha_1^0 p + \alpha_2^0 p^2 + \cdots + \alpha_n^0 p^n} u(t) \tag{6.1}$$

Where $u(t)$ and $y(t)$ are the input and output respectively. $\alpha_i^0 \in R(i = 0,1,...,n)$ And $\beta_i^0 \in R(i = 0,1,...,m)$ are co-efficient parameters while p is a differential operator.

The goal is to find the system model described by the model class

$$\mathcal{M} = \left\{ \frac{B(p)}{A(p)} = \frac{\beta_0 + \beta_1 p + \cdots + \beta_m p^m}{1 + \alpha_1 p + \cdots + \alpha_n p^n} \right\} \quad [54] \tag{6.2}$$

The following assumptions are made in this type of approach [54]

- The system has zero initial state.
- Though the true parameters α_i and β_i are unknown, $A(p)$ and $B(p)$ are coprime and their order n and m are known.
- We can measure $\tilde{y}(t)$, the output contaminated with noise, $\tilde{y}(t) = y(t) + w(t)$, where $w(t)$ is zero-mean measurement noise.
- We can repeat the experiments at the same initial condition on the time interval [0, T]

6.3.2 Steps involved in system identification using ILC [54]

System identification approach is pursued in the same way as given in [51,52,54].

- Define,

$$A^k(p) = \alpha_0^k + \alpha_1^k p + \cdots . + \alpha_n^k p \ \ \boldsymbol{and} \ \ B^k(p) = 1 + \beta_1^k p + \cdots . + \beta_m^k p \tag{6.3}$$

- Generate $\boldsymbol{u^k(t) = A^k(p)r(t)}$ **[54]** (6.4)

- Inject $u^k(t)$into the system, and collect $\boldsymbol{\tilde{y}^k(t)}$

- Generate $\boldsymbol{B^k(p)r(t)}$

- Obtain the mismatch signal $\epsilon^k(t)$ as $\boldsymbol{\epsilon^k(t) = \tilde{y}^k(t) - B^k(p)r(t)}$ (6.5)

Therefore, $$\epsilon^k(t) = \frac{B^0(p)}{A^0(p)} A^k(p)r(t) + w^k(t) - B^k(p)r(t) \quad \textbf{[54]} \tag{6.6}$$

Now project $\epsilon^k(t)$onto a finite-dimensional sub-space

$$\mathcal{F} \triangleq span\{f_1(t), f_2(t), \ldots\ldots, f_{n+m+1}(t)\} \tag{6.7}$$

Therefore the data generation scheme for *kth* trial is depicted in the following discussion. Now the projection of $\epsilon^k(t)$ onto the subspace is given by

$$\varepsilon^k(t)_{\mathcal{F}} = \delta_1^k f_1(t) + \cdots + \delta_{n+m+1}^k f_{n+m+1}(t) \tag{6.8}$$

Now defining, $$\delta^k \triangleq [\delta_1^k, \ldots\ldots, \delta_{n+m+1}^k]^T$$

Let, $$\gamma^0 = [\alpha_1^0, \ldots\ldots, \alpha_n^0, \beta_0^0, \ldots\ldots, \beta_m^0]^T$$

$$\gamma^k = [\alpha_1^k, \ldots\ldots, \alpha_n^k, \beta_0^k, \ldots\ldots, \beta_m^k]^T \tag{6.9}$$

Then the iterative identification procedure [54] is described as follows

1. Given γ^0, set $k = 0$.
2. Generate δ^k from γ^k according to the scheme.
3. Update γ^k by the following rule.

$$\gamma^{k+1} = \gamma^k + H^k \delta^k \quad \textbf{[54]} \tag{6.10}$$

where H^k is the learning gain. If $\|\gamma^{k+1} - \gamma^k\| \leq a$, stop iterations. Else, set $k = k+1$ and go back to Step 1, where a is the error in parameters.

6.3.3 Modes of system identification

System identification can be performed in open loop [52] or closed loop [53]. Figures 6.1 and 6.2 explain open loop identification method [52] and closed loop identification technique [53], respectively.

Open loop identification technique

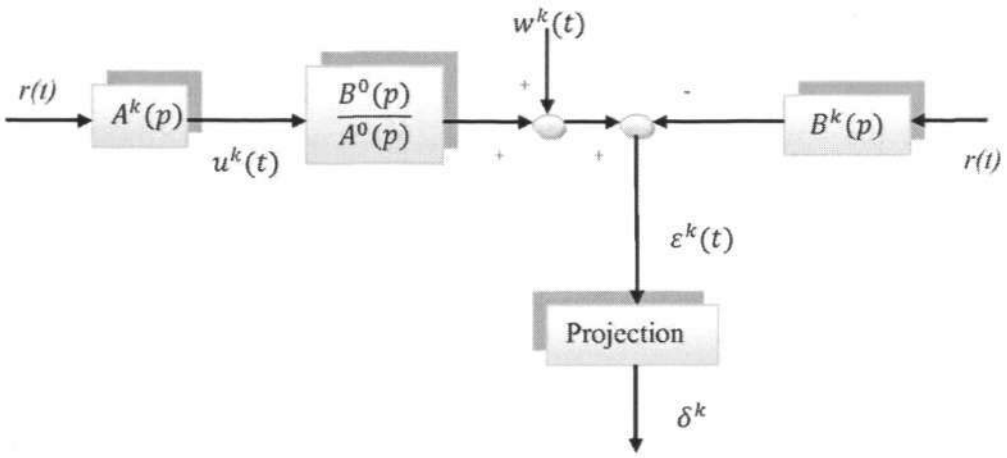


Figure 6.1 : Block diagram of open loop system identification [52]

Closed loop system identification technique

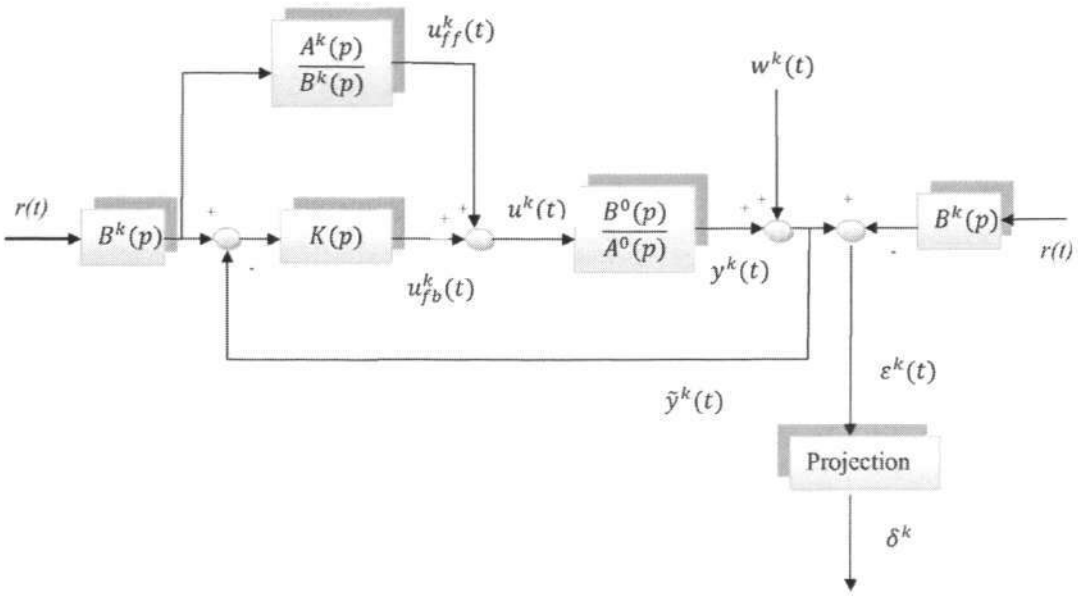


Figure 6.2 : Block diagram of closed loop system identification [53]

6.3.4 Implementation

In this section, we discuss digital implementation of iterative identification approach [54], when I/O data is available at sampled times. For example input and output data are represented as $u(iT_s)$ *and* $y(iT_s)$, respectively, where $i=0,1,....q$, T_s is the sampling time, q is number of samples and $qT_s = T$. The input is applied to the plant through a zero-order hold (ZOH). Choose a smooth reference signal *r(t)* and define,

$$\gamma^0 = [\alpha_1^0, \ldots\ldots, \alpha_n^0, \beta_0^0, \ldots., \beta_m^0]^T \quad and \quad \gamma^k = [\alpha_1^k, \ldots\ldots, \alpha_n^k, \beta_0^k, \ldots., \beta_m^k]^T \tag{6.11}$$

$$V_r(t) = \left[r(t), \frac{d\,r(t)}{dt}, \ldots\ldots\ldots, \frac{d^{\,n} r(t)}{dt^n} \right] \quad [54] \tag{6.12}$$

Define, $V_{dr} \in \mathbb{R}^{(q+1)(n+1)}$, $V_{dr} \triangleq \begin{bmatrix} r(0) & \dot{r}(0) & \cdots & r^{(n)}(0) \\ r(T_s) & \dot{r}(T_s) & & r^{(n)}(T_s) \\ \vdots & & \cdots & \vdots \\ r(qT_s) & \dot{r}(qT_s) & \cdots & r^{(n)}(qT_s) \end{bmatrix}$ [54] (6.13)

Hence we have, $$u^k = V_{dr}\gamma^k \tag{6.14}$$

Let the finite-dimensional subspace described as

$$\mathcal{F} \triangleq span\{f_1(t), f_2(t), \ldots\ldots, f_{n+m+1}(t)\} \tag{6.15}$$

We define another matrix $V_{df} \in \mathbb{R}^{(q+1)(n+m+1)}$

$$V_{df} \triangleq \begin{bmatrix} f_1(0) & f_2(0) & \cdots & f_{n+m+1}(0) \\ f_1(T_s) & f_2(T_s) & & f_{n+m+1}(T_s) \\ \vdots & & \cdots & \vdots \\ f_1(qT_s) & f_2(qT_s) & \cdots & f_{n+m+1}(qT_s) \end{bmatrix} \quad [54] \tag{6.16}$$

QR decomposition of V_{df}

$$V_{df} = QR \quad and \quad Q^TQ = I_{n+m+1} \tag{6.17}$$

where $Q \triangleq [f_1, f_2, \ldots\ldots f_{n+m+1}] \in \mathbb{R}^{(q+1)(n+m+1)}$ and $R \in \mathbb{R}^{(n+m+1)(n+m+1)}$ is a non-singular upper triangular matrix. Vector representation of mismatch signal ϵ^k is given to be,

$$\boldsymbol{\delta^k = Q^T \epsilon^k} \quad [54] \tag{6.18}$$

$$\boldsymbol{\epsilon^k = \tilde{y}^k - B(p)r} \quad [54] \tag{6.19}$$

6.3.5 Update law

We use Kalman filter type learning law and its typical form is given as [53, 54,55,61],

$$\gamma^{k+1} = \gamma^k + H^k \delta^k \tag{6.20}$$

where H^k is learning gain and $\gamma^k = [\alpha_1^k, \ldots\ldots, \alpha_n^k, \beta_0^k, \ldots\ldots, \beta_m^k]^T$ is parameter matrix. We discuss choice of H^k in this section. It is evident that tracking error δ^k is linearly dependent on parameter estimate γ^k. We introduce a matrix $M \in \mathbb{R}^{(n+1)(n+1)}$ and an offset term $\bar{\delta} \in \mathbb{R}^{(n+1)}$.

$$\delta^k = M\gamma^k + \bar{\delta} \tag{6.21}$$

As derived in [54], we use following condition to compute learning gain matrix,

$$H^k = -\frac{1}{k+1} M^{-1} \tag{6.22}$$

6.3.6 Estimate of M

Model of a linear system can be represented as

$$y^k = Gu^k \tag{6.23}$$

where u^k is the input with ZOH and y^k is the corresponding output, $G \in \mathbb{R}^{(q+1)(q+1)}$ is a matrix of the form,

$$G = \begin{bmatrix} g_0 & 0 & 0 & & 0 \\ g_1 & g_0 & 0 & \cdots & 0 \\ g_2 & g_1 & g_0 & & 0 \\ & \vdots & & \ldots & \vdots \\ g_q & g_{q-1} & g_{q-2} & \cdots & g_0 \end{bmatrix} \tag{6.24}$$

The first column of G is the output y when impulse input is injected to the system.

Tracking error is represented as $\boldsymbol{\varepsilon^k = Gu^k - r}$ (6.25)

From [55], we use the following equations

$$\delta^k = Q^T G\, V_{dr} (\gamma^k - \gamma^0) \tag{6.26}$$

$$M = Q^T G\, V_{dr} \;\; and \;\; \gamma^0 = M^{-1} Q^T r \quad [55] \tag{6.27}$$

We obtain the estimate $\widehat{G}$ of G using simple experiments and calculate $\widehat{M}$ and γ^0.

6.4 System identification of parameters of SCARA robot *via* iterative learning

6.4.1 Dynamics of the SCARA robot used in robotic test bed

The dynamics of a robot arm with n joints is governed by the differential equation,

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) + F_v(\dot{\theta}) + F_c = \tau \tag{6.28}$$

where $M(\theta)$ is n×n inertia matrix of the manipulator, $C(\theta,\dot{\theta})$ is n×n matrix of centrifugal and Coriolis terms, $G(\theta)$ is n×1 vector of gravity terms, $F_v(\dot{\theta})$is n×1 vector of viscous friction terms, $F_c$is n×1 vector of coulomb terms and τ is n×1 vector of input torque (generated by the joint motor). The terms $\theta, \dot{\theta}$ *and* $\ddot{\theta}$ are nx1 vectors of the output link position, velocity and acceleration, respectively.

In this thesis, joint 2 (T1) and joint 3 (T2) are used for parameter identification. Both the links move in horizontal plane. The dynamics are these two links together can be expressed as

$$\begin{bmatrix} (m_1+m_2)a_2^2 + m_3a_3^2 + 2m_3a_2a_3cos\theta_3 & m_3a_3^2 + m_3a_2a_3cos\theta_3 \\ m_3a_3^2 + m_3a_2a_3cos\theta_3 & m_3a_3^2 \end{bmatrix}\begin{bmatrix}\ddot{\theta}_2\\ \ddot{\theta}_3\end{bmatrix} \tag{6.29}$$

$$+\begin{bmatrix} -m_3a_2a_3(2\dot{\theta}_2\dot{\theta}_3 + \dot{\theta}_3^2)sin\theta_3 \\ m_3a_2a_3\dot{\theta}_2^2 sin\theta_3 \end{bmatrix} + \begin{bmatrix} f_{v2}\dot{\theta}_2 \\ f_{v3}\dot{\theta}_3 \end{bmatrix} + \begin{bmatrix} f_{c2} \\ f_{c3} \end{bmatrix} = \begin{bmatrix} \tau_2 \\ \tau_2 \end{bmatrix}$$

where θ_j, f_{vj}, f_{cj} *and* τ_j, j=2,3, are joint angles, viscous frictions, Coulomb frictions and control torques of joints 2(T1-axis) and joints 3(T2-axis), respectively. m_j *and* $a_j, j = 2,3$, are masses and mass centers of links 2 and 3, respectively.

Joint 2 and joint 3 are stabilized by using feedback P controllers with $k_{p2} = k_{p1} =$ 0.1. Linear approximation of the closed loop system is performed.

Figure 6.3 shows the system structure of the feedback loop.

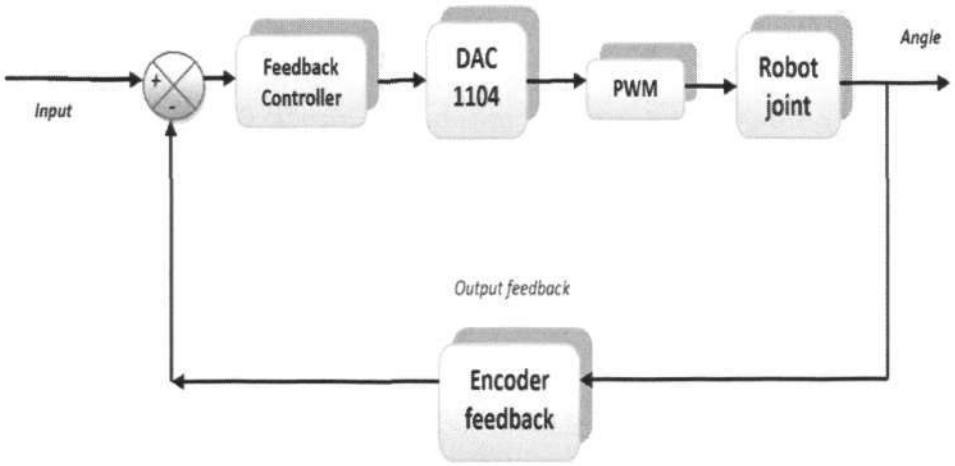


Figure 6.3 : System model of the feedback loop, one joint

Linearize the non-linear terms and transfer function of the two joints can be represented as,

$$G_m(s) = \frac{\begin{bmatrix} c_1 s^2 + c_2 s + c_3 & c_4 s^2 \\ c_5 s^2 & c_6 s^2 + c_7 s + c_8 \end{bmatrix}}{s^4 + c_9 s^3 + c_{10} s^2 + c_{11} s + c_{11}} \tag{6.30}$$

where $c_j, j = 1,2, \ldots 11$, are constants to be determined using system identification.

When approximating the SISO transfer function of joint 2, joint 3 is locked at $90^0 (\theta_3 = 90^0)$. Hence, SISO transfer function of joint 2 can be approximated to

$$G_{m2}(s) = \frac{1}{\boldsymbol{\alpha 1}_2 s^2 + \boldsymbol{\alpha 1}_1 s + \boldsymbol{\alpha 1}_0} \tag{6.31}$$

When approximating the SISO transfer function of joint 3, joint 2 is locked at $90^0 (\theta_2 = 90^0)$. Hence, SISO transfer function of joint 3 can be approximated to

$$G_{m3}(s) = \frac{1}{\boldsymbol{\alpha 2}_2 s^2 + \boldsymbol{\alpha 2}_1 s + \boldsymbol{\alpha 2}_0} \tag{6.32}$$

where , $\boldsymbol{\alpha j}_0, \boldsymbol{\alpha j}_1 and\ \boldsymbol{\alpha j}_2$, $j = 1,2,$ are the system parameters to be identified using ILC approach. The differential operator in this case is "s" in place of "p".

6.4.2 Objective of system identification of SCARA robot

From Eq 6.2, we can arrive at the objective of system identification of SCARA robot.

$$\mathcal{M}_{T1} = \left\{ \frac{B(s)}{A(s)} = \frac{1}{\boldsymbol{\alpha 1}_2 s^2 + \boldsymbol{\alpha 1}_1 s + \boldsymbol{\alpha 1}_0} \right\} \ldots\ldots T1\ axis \tag{6.33}$$

$$\mathcal{M}_{T2} = \left\{ \frac{B(s)}{A(s)} = \frac{1}{\boldsymbol{\alpha 2}_2 s^2 + \boldsymbol{\alpha 2}_1 s + \boldsymbol{\alpha 2}_0} \right\} \ldots\ldots\ldots T2\ axis \tag{6.34}$$

System identification of continuous time systems should be carried out using iterative learning control based identification. Parameters that should be identified using ILC based system identification are $\boldsymbol{\alpha 1}_0, \boldsymbol{\alpha 1}_1$ *and* $\boldsymbol{\alpha 1}_2$*(for joint* 2) and $\boldsymbol{\alpha 2}_0, \boldsymbol{\alpha 2}_1$ *and* $\boldsymbol{\alpha 2}_2$ *for joint* 3.

6.4.3 Implementation

In this section, we discuss digital implementation of iterative identification approach on joints 2 and joints 3, when I/O data is available at sampled times. I/O data are represented as $u(iT_s)$ *and* $y(iT_s)$, respectively, where *i=0,1,....q,* T_s is the sampling time, q is number of samples and $qT_s = T$. The input is applied to the plant through a zero-order holder (ZOH).

Since the assumed system is second order, $\boldsymbol{B^k(s) = 1}$

In this case, for initial parameters for joint 2,

$$\boldsymbol{\gamma^0}_{\boldsymbol{T1}} = [\,\boldsymbol{\alpha 1_0}^0, \boldsymbol{\alpha 1_1}^0, \boldsymbol{\alpha 1_2}^0,]^T \text{ and } \boldsymbol{\gamma^k}_{\boldsymbol{T1}} = [\,\boldsymbol{\alpha 1_0}^k, \boldsymbol{\alpha 1_2}^k, \boldsymbol{\alpha 1_2}^k]^T \tag{6.35}$$

For joint 3,

$$\boldsymbol{\gamma^0}_{\boldsymbol{T2}} = [\,\boldsymbol{\alpha 2_0}^0, \boldsymbol{\alpha 2_1}^0, \boldsymbol{\alpha 2_2}^0]^T \text{ and } \boldsymbol{\gamma^k}_{\boldsymbol{T2}} = [\boldsymbol{\alpha 2_0}^k, \boldsymbol{\alpha 2_1}^k, \boldsymbol{\alpha 2_2}^k]^T \tag{6.36}$$

In our experiment *n=2 and m=0.* Therefore equation 6.12 reduces to,

$$V_r(t) = \left[r(t), \frac{d\,r(t)}{dt}, \frac{d^2 r(t)}{dt^2} \right] \tag{6.37}$$

To simplify the discussion, we present the identification of joint 3 (T2-axis). Therefore let $\boldsymbol{\gamma^k} = \boldsymbol{\gamma^k}_{\boldsymbol{T2}}$ and $\boldsymbol{\gamma^0} = \boldsymbol{\gamma^0}_{\boldsymbol{T2}}$. Input to be applied at iteration k,

$$u(t)^k = A^k(s)r(t) = V_r(t)\gamma^k \tag{6.38}$$

Choose q=1000, T= 10 sec and $T_s = 0.01\ sec$. Therefore Eq 6.13 reduces to,

$$V_{dr} \triangleq \begin{bmatrix} r(0) & \dot{r}(0) & r^{(2)}(0) \\ r(T_s) & \dot{r}(T_s) & r^{(2)}(T_s) \\ \vdots & \vdots & \vdots \\ r(qT_s) & \dot{r}(qT_s) & r^{(2)}(T_s) \end{bmatrix} \tag{6.39}$$

Let the finite-dimensional subspace described as

In our case, $\mathcal{F} \triangleq span\{f_1(t), f_2(t), f_3(t)\}$

In this case,

$$V_{df} \triangleq \begin{bmatrix} f_1(0) & f_2(0) & f_3(0) \\ f_1(T_s) & f_2(T_s) & f_3(T_s) \\ \vdots & \vdots & \vdots \\ f_1(qT_s) & f_2(qT_s) & f_3(qT_s) \end{bmatrix} \tag{6.40}$$

QR decomposition of V_{df}

$$V_{df} = QR \ \ and \ \ Q^TQ = I_{n+m+1} \tag{6.41}$$

where $Q \triangleq [f_1, f_2, f_2] \in \mathbb{R}^{(1001)(3)}$ and R $\in \mathbb{R}^{(3)(3)}$ is a non-singular upper triangular matrix. Vector representation of mismatch signal ϵ^k is given to be,

$$\boldsymbol{\delta^k = Q^T \epsilon^k} \tag{6.42}$$

$$\boldsymbol{\epsilon^k = \tilde{y}^k - r} \tag{6.43}$$

6.4.4 Update law

We use Kalman filter type learning law and its typical form is given as [53, 54,55,61],

$$\gamma^{k+1} = \gamma^k + H^k\delta^k \tag{6.44}$$

$\gamma^k = [\alpha_0^k, \alpha_1^k, \alpha_2^k]^T$. Based on the Eq 6.21 and 6.22, H^k is computed to be

$$\delta^k = M\gamma^k + \bar{\delta} \ \ and \ \ H^k = -\frac{1}{k+1}M^{-1} \quad [54]$$

6.4.5 Estimate of M

Simple experiments on the system is performed and system parameters such as $\widehat{G}$ of G, $\widehat{M}$ and γ^0. Using Eq 6.26 and Eq 6.27,the parameters computed are,

$$\delta^k = Q^T G\, V_{dr}\,(\gamma^k - \gamma^0) \tag{6.45}$$

$$M = Q^T G\, V_{dr} \;\; and\; \gamma^0 = M^{-1} Q^T r \tag{6.46}$$

6.4.6 Steps involved in system identification using ILC

In this section, basic steps in system identification technique for T2-axis

1. Define,

$$A^k(s) = \alpha_0^k + \alpha_1^k s + \alpha_2^k s^2 \;\; \boldsymbol{and} \;\; B^k(s) = 1 \tag{6.47}$$

2. Given γ^0, i.e, $\boldsymbol{\gamma^0 = [\alpha_0^0, \alpha_1^0, \alpha_2^0]^T}$ and set $\boldsymbol{k = 0}$.
3. Generate $\boldsymbol{u^k(t) = A^k(s) r(t)}$ **[52]**
4. Inject $\boldsymbol{u^k(t)}$ into the system, and collect $\boldsymbol{\tilde{y}^k(t)}$
5. Generate $\boldsymbol{B^k(s) r(t) = r(t)}$
6. Obtain the mismatch signal $\epsilon^k(t)$ as $\boldsymbol{\epsilon^k(t) = \tilde{y}^k(t) - r(t)}$ (6.48)
7. Project $\epsilon^k(t)$ onto the finite-dimensional subspace

$$\mathcal{F} \triangleq span\{f_1(t), f_2(t), \ldots\ldots, f_{n+m+1}(t)\} \triangleq span\{f_1(t), f_2(t), f_3(t)\}$$

Projection of $\epsilon^k(t)$ onto the subspace is given by

$$\varepsilon^k(t)_{\mathcal{F}} = \delta_1^k f_1(t) + \delta_2^k f_2(t) + \delta_3^k f_3(t) \tag{6.49}$$

Now defining, $\delta^k \triangleq [\delta_1^k, \delta_2^k, \delta_3^k]^T$ and $\gamma^k = [\alpha_0^k, \alpha_1^k, \alpha_2^k]^T$ (6.50)

8. Update γ^k by the following rule.

$$\boldsymbol{\gamma^{k+1} = \gamma^k + H^k \delta^k} \tag{6.51}$$

where H^k is the learning gain. If $\|\gamma^{k+1} - \gamma^k\| \le a$, stop the iteration. Otherwise, set k = k+1 and go back to Step 3, where a is the error in the parameters. In our case, we choose a=0.05.

6.4.7 Flowchart for system identification using ILC

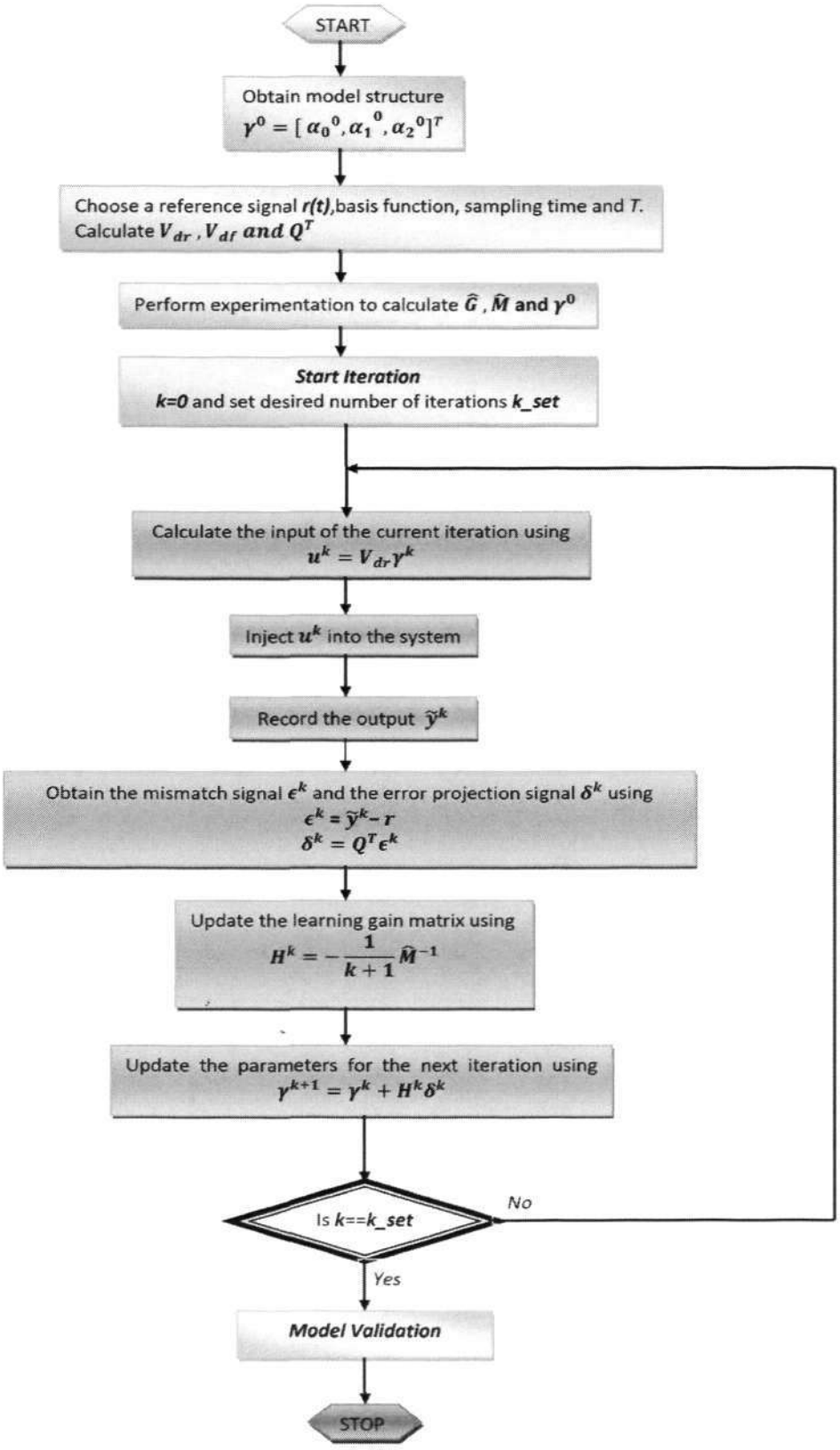


Figure 6.4 : Flowchart for system identification using ILC

6.5 Experimental results

The signal *r(t)* is chosen to be the output of the following system when the input *u(t)* is injected into the system.

$$F(s) = \frac{10^2}{(s+10)^2} \tag{6.52}$$

$$u(t) = 80e^{-2\pi t}(1 - \cos 2\pi t) \tag{6.53}$$

We choose the basis function to be $f_i(t) = r^{i-1}(t)$, *i=1,2,3.* From these functions we generate the matrices V_{df}, Q, R *and* V_{dr} . In our case, $V_{dr} = V_{df}$.

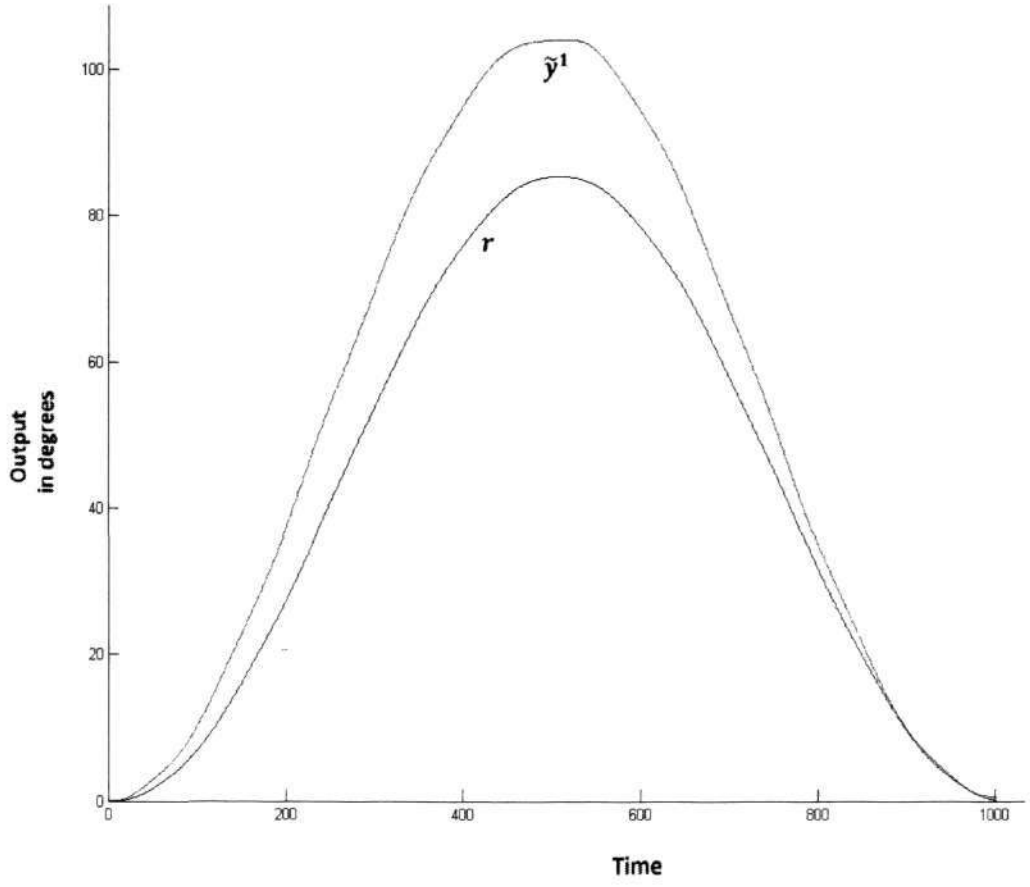


Figure 6.5 : Output $\tilde{y}$ and signal r after first iteration

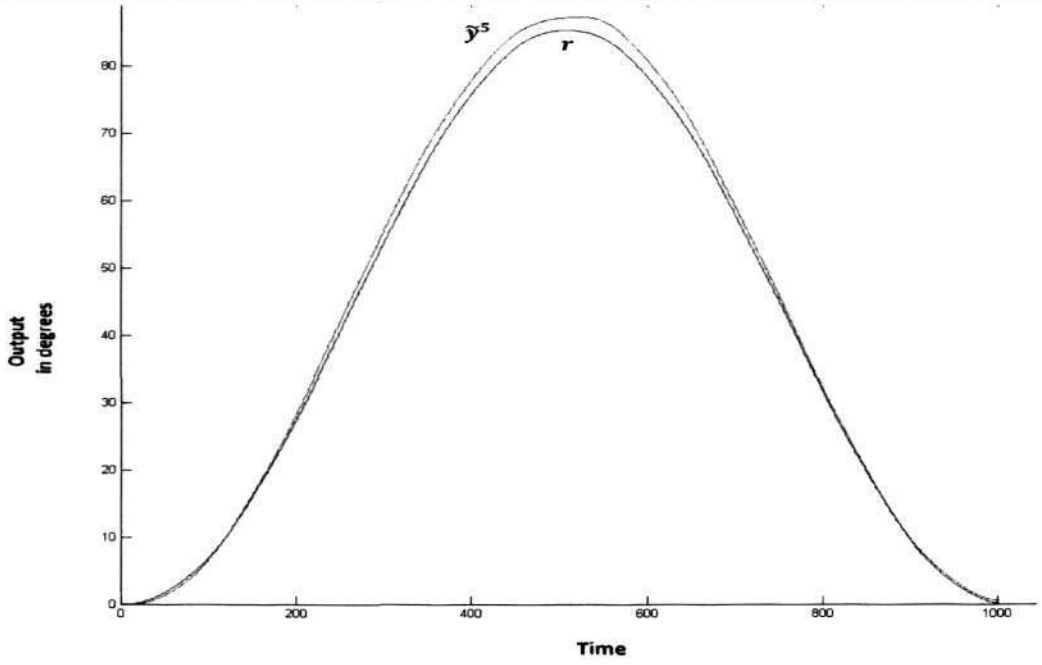


Figure 6.6 : Output $\tilde{y}$ and signal r after 5 iterations

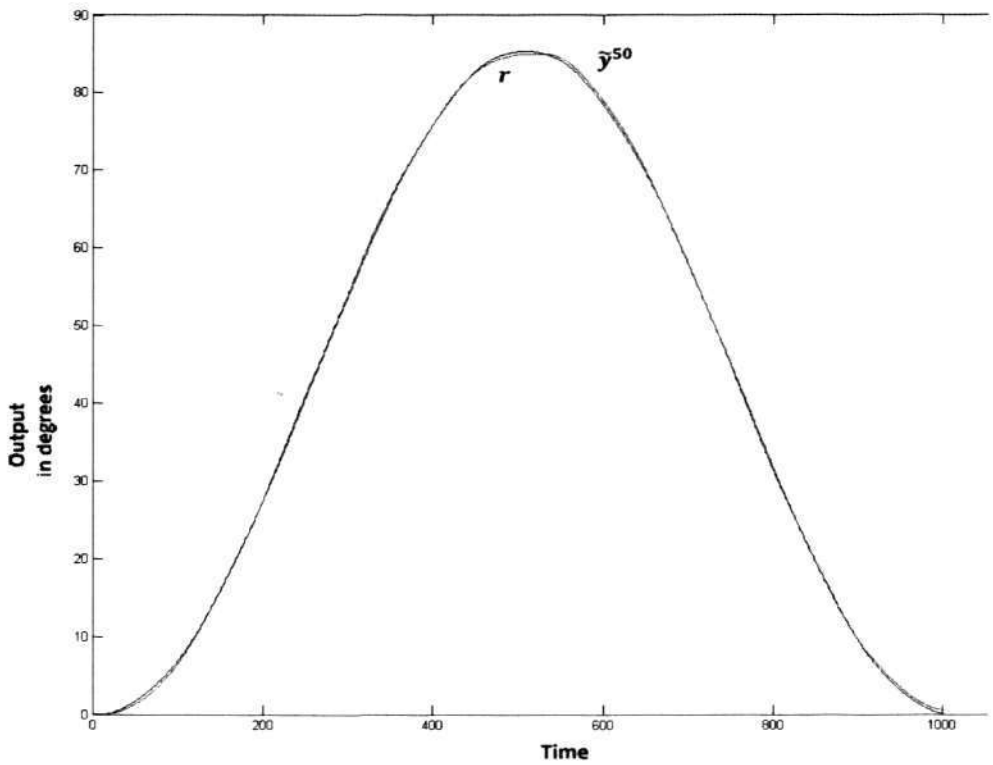


Figure 6.7 : Output $\tilde{y}$ and signal r after 50 iterations

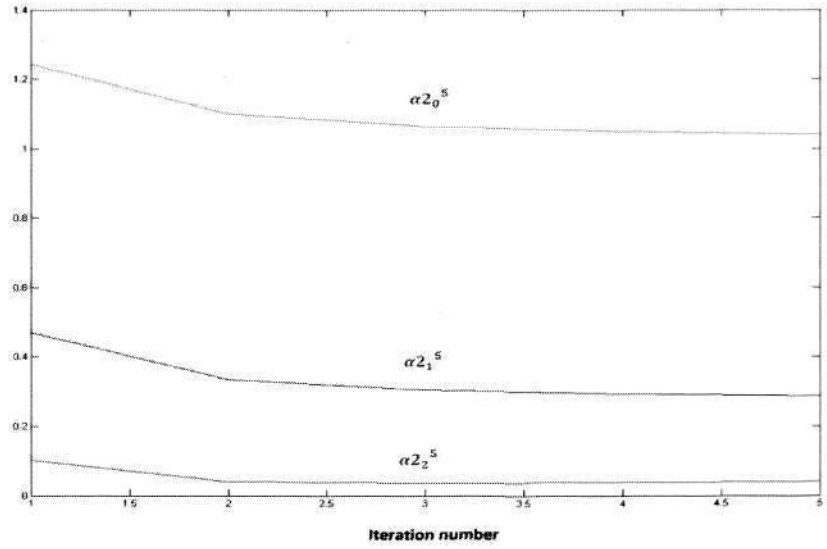


Figure 6.8 : parameters identified after 5 iterations

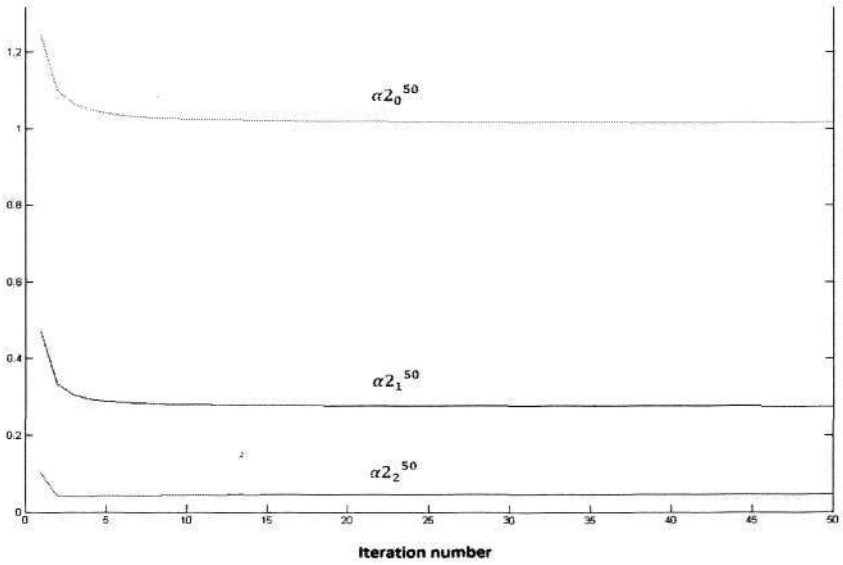


Figure 6.9 : parameters identified after 50 iterations

Obtained transfer function for T2 axis after 50 iterations is

$$G_{T2}(s) = \frac{22.05}{s^2+6.074s+22.41}$$

Above mentioned approach is applied to T1-axis also and obtained transfer function is

$$G_{T1}(s) = \frac{281.9}{s^2+47.89s+282.05}$$

Model Validation: Same input is applied to the actual system and simulated system in MATLAB using the parameters identified using ILC approach. Output of the actual system y_{act} and output of the simulated system y_{sim} are recorded and plotted. The root mean square error is determined

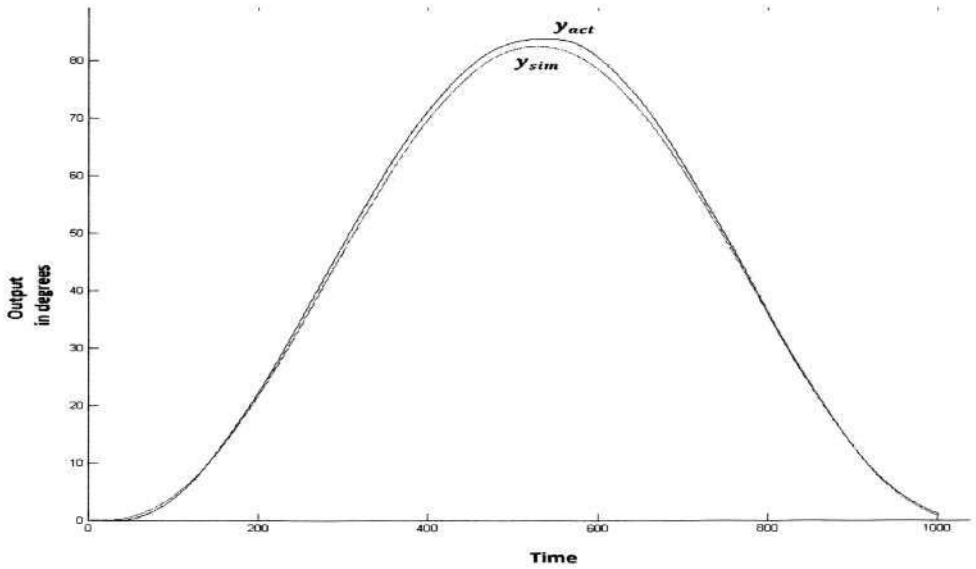


Figure 6.10 : Outputs y_{sim} and y_{act} after 5 iterations

rms error =1.101 (after 5 iterations)

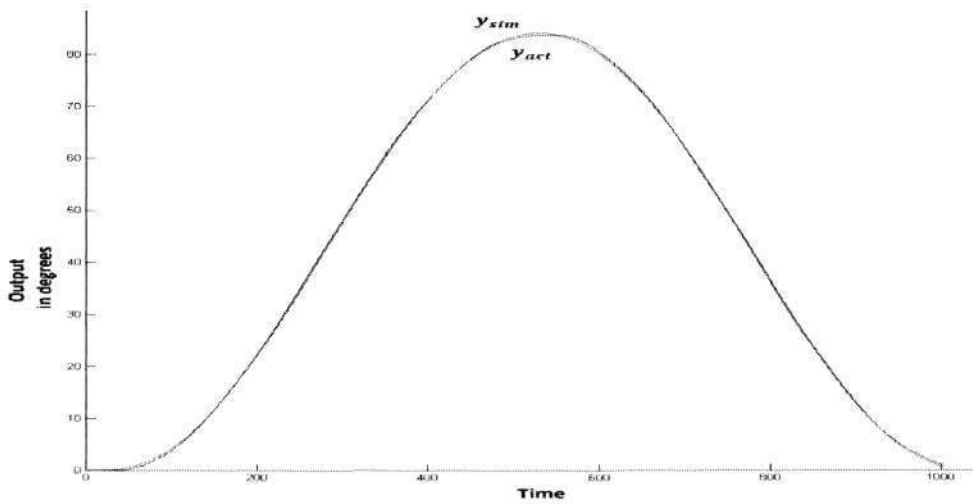


Figure 6.11 : y_{sim} and y_{act} after 50 iterations

rms error = 0.0275 (after 50 iterations)

6.6 Concluding remarks

The transfer function obtained from system identification using ILC for T1 and T2 axes, respectively after 50 iterations are

$$G_{T1}(s) = \frac{281.9}{s^2 + 47.89s + 282.05} \quad \textbf{\textit{and}} \quad G_{T2}(s) = \frac{22.05}{s^2 + 6.074s + 22.41} \qquad (6.54)$$

Figure 6.11 is a plot which constitutes output of actual system y_{act} and output of simulated system y_{sim}, when same input is applied. Both responses y_{act} and y_{sim} are similar with root mean square error of 0.0275. This shows that model identified is a *valid* model. Figure 6.9 depicts the parameter values identified up to 50 iterations. As we can observe, that parameter values remain constant after 25 iterations. This reveals that identification of parameters is complete with negligible errors. Figure 6.5 plots output response of first iteration and reference signal *r*. We can note that there is huge variation between *r* and output response. Output response in the first iteration is obtained without the parameters being identified using ILC. Figure 6.7 shows that, after 50 iterations error $\boldsymbol{\varepsilon^k = y^k - r}$, *k*=50, is minute. Using ILC based identification approach parameters of the T1 and T2 axis of SCARA robot are identified and the results depict that model identified is valid.

One of important merit of this approach is that, no time-derivatives of I/O data is required in identification process. This procedure avoids numerical differentiation in identification process, which can give rise to unwanted errors. Quality of identified model can be easily estimated by observing the tracking control performance in experiments.

Chapter 7

Conclusions and future works

7.1 Conclusions

This project is dedicated to the development of a robotic test bed for experimental research on iterative learning control and real-time implementation of advanced ILC design methods on robotic test bed.

The project starts from the construction and development of a robotic test bed. It is developed using MATLAB /SIMULINK and Real-time workshop (RTW), dSPACE interface libraries and an industrial SCARA robot SEIKO TT-3000. The experimental platform meets the requirements for implementation of advanced control algorithms. The test bed is flexible, expandable and user-friendly. It is easy to use because of the modular software architecture. Code developed is divided into numerous custom-made modules, which makes it easier for the designer to visualize the entire system and develop the program accordingly. Flexibility in operation of the test bed is demonstrated by the fact that designer can choose different parameters like the desired trajectory, the operating time, the control law, the filter parameters and the number of iterations. There is also a provision for the designer to input new parameters or functions in the software architecture according to the requirements of the control algorithm. This property demonstrates that test bed is open for future expansion. Graphical user interface developed using *ControlDesk* and MATLAB provides an efficient means for analyzing and observing the real-time data. For control algorithms, two basic classes of learning control algorithms are studied and implemented, namely P-type and A-type.

Once the test bed is developed, basic ILC laws are tested and results prove that test bed can be used for implementing advanced ILC algorithms. All the ILC laws implemented shows good tracking error convergence and the controllers realized are effective in learning. ILC laws were implemented and tested with minimal knowledge of the system.

Convergence and robustness conditions for MIMO systems are presented. Continuous time MIMO system with disturbance was considered and ILC algorithms were developed. Two different types decoupling techniques (generalized decoupling and simplified decoupling) with ILC were explained. Performance of ILC algorithms with *decoupler* is effective due to the elimination or reduction of interactions in a MIMO system. It is also shown that generalized decoupling technique is better than simplified decoupling in terms of error convergence and output tracking. Advantages include,

- ILC and decoupling technique supplement each other
- General approach for selection of learning gain matrix
- Elimination or reduction of interactions in a MIMO system.
- Decoupling offers flexibility to design and control system performance is improved.

System identification of T1 and T2 axes of the SCARA robot using ILC is performed. For the identification process, the structure of the SIEKO robot is utilized. Simulations are performed initially in MATLAB and later the system identification algorithms are implemented on the test bed. The parameters identified are tested and proved to be reliable.

In conclusion the robotic test bed is set up which meets the needs of research in iterative learning control. This project demonstrates effectively that ILC algorithms in theory can be used in practice.

7.2 Future Works

This project offers a wide scope for extensions and improvements, such that more advanced control algorithms can be implemented. The first extension to the project can be, to use a different sensor apart from optical encoders. Each axis of the robotic test bed is equipped with an encoder and these encoders measure the position of the corresponding axis. Use of another sensor such as camera, can enhance the operation of robotic test bed. A camera sensor can be used to measure the tool tip position with respect to a local frame. This enhancement has several advantages. The homing procedure currently used is to give a constant negative voltage to the DAC and allow the robot axis to move till the hardware limit point. The hardware limit points are mechanical restrictions mounted the robot itself. Hence, by using camera sensor can determine a reliable homing position of the robot. In industrial processes, position and velocity control of the end effector within work envelope is a crucial criterion in robotic control. Point to point control, path tracking and trajectory planning of the end effector can be performed efficiently. Use of LED display panels and dSPACE simulator boards can enhance the debugging process.

System identification of T1 and T2 axes of the SCARA robot is performed in this project. In a similar way, parameters of A and Z axes can be identified using ILC. In this manner, we can create a complete mathematical model of the SCARA robot.

Bibliography

[1] M. Uchiyama, "*Formation of high speed motion pattern of a mechanical arm by trial*", Transactions of SICE, Vol. 21, Issue a6, pp. 706-712, 1978

[2] S. Arimoto, S. Kawamura, and F. Miyazaki, "*Bettering operation of robots by learning*", Journal of Robotic Systems, Vol. 1, pp. 123-140, 1984.

[3] G. Casolino and G. Bartolini, "*A learning procedure for the control of movements of robotic manipulators*", Proceedings of 4th IASTED Symposium on Robotics and Automation, pp. 108-111,1984

[4] J. J. Craig, " *Adaptive control of manipulator through repeated trials*", Proceedings of 1984 American Control Conference, pp. 1566-1573, 1984

[5] R. H. Middleton, G. C. Goodwin and R.W. Longman, " *A method for improving dynamic accuracy of the robot performing a repetitive task*", International Journal of Robotic Research, Vol. 8, pp. 67-74, 1989

[6] S. Kawamura, F. Miyazaki, and S. Arimoto, "*Iterative learning control for robotic systems*", Proceedings of 1984 International Conference on Industrial Electronics, Control, and Instrumentation, pp. 393-398, 1984

[7] C.G Atkeson and J. McIntyre, "*Robot trajectory learning through practice*", IEEE conference on Robotics and Automation, pp. 1737-1742, 1986

[8] S. Arimoto, S. Kawamura and F. Miyazaki, "*Iterative learning control for robotic systems*", Proceedings of IECON'84, (Japan), pp. 393-398, 1984

[9] Y. Gu and N. Loh, "*Learning control in robotic systems*", Proceedings of IEEE International Symposium of Intelligent control, pp. 360-364, 1987

[10] T. Mita and E. Kato, "*Iterative control and its applications to motion control arm- a direct approach to servo problems*", Proceedings of IEEE International Conference on Decision and control, pp.1393-1398, 1985

[11] M. Norrlöf, and S. Gunnarsson "*Time and frequency domain convergence properties in iterative learning control*", International Journal of Control **75**(14), pp. 1114-1126, 2002

[12] D. Wang and Y. Ye , "*Design and Experiments of Anticipatory Learning control: Frequency domain approach*", IEEE/ASME Transactions on Mechatronics, Vol. 10, Issue 3, pp. 305 – 313, 2005

[13] J. Hauser, "*Learning control for a class of nonlinear systems*", 26th IEEE Conference on Decision and control, pp. 859-860, 1987

[14] Z. Bien and K.M. Huh, "*Higher-order iterative learning control algorithm*", IEE Proceedings Part D, Vol. 136, pp.105-112, 1989

[15] S. Arimoto, S. Kawamura and F. Miyazaki, "*Convergence, stability, and robustness of learning control schemes for robot manipulators*", International Symposium on Robot manipulators: Modeling, Control and Education, pp. 307-316, 1986

[16] G. Heinzinger, D. Fenwick , B. Paden and F. Miyazaki, "*Stability of learning control with disturbances and uncertain initial conditions*", IEEE Transactions on Automatic control ,Vol. 137, pp. 110-114,1992

[17] S. Arimoto, "*Robustness of learning control for robot manipulators*", Proceedings of IEEE International Conference on Robotics and Automation, pp.1528-1533, 1990

[18] D. Wang, "*On D-type and P-type designs and anticipatory approach*", International Journal of Control, Vol. 73, Issue 10, pp. 890-901, 2000

[19] B. Zhang, D. Wang and Y. Ye, "*On learning transient and cut off frequency tuning in ILC*", Proceedings of 1st IEEE Conference on Robotics, Automation and Mechatronics, Vol. 1, pp. 484 – 489, 2004

[20] Y. Ye and D. Wang, "*Better robot tracking accuracy with phase lead compensated ILC*", Proceedings of IEEE International Conference on Robotics and Automation, Vol. 3, pp. 4380 – 4385, 2003

[21] Y. Ye and D. Wang, " *Implementation of ILC batch update using a robotic experimental setup*" , Microprocessors and Microsystems, Vol. 30, Issue 5, pp. 259-267, 2006

[22] B. Zhang, D. Wang and Y. Ye, "*Experimental study of Time-frequency based ILC*", International Conference on Control, Automation, Robotics and Vision, Vol. 1, pp. 619- 624, 2004

[23] B. Zhang, D. Wang and Y. Ye, "*On learning transient and cut off frequency tuning in ILC*", Proceedings of IEEE Conference on Robotics, Automation and Mechatronics, Vol. 1, pp. 484 – 489, 2004

[24] Y. Ye and D. Wang, "*Multi-Channel Design for Iterative Learning Control*", Proceedings of 2004 American Control Conference, Vol. 6, pp. 5138- 5143, 2004

[25] Y. Ye and D. Wang, "*Multi-channel Design for ILC with Robot Experiments*", 7th International Conference on Control, Automation, Robotics and Vision (ICARCV'OZ), Vol. 2, pp. 1066- 1070, 2002

[26] B. Zhang, D. Wang and Y. Ye, "*Wavelet transform-based frequency tuning ILC*", IEEE Transactions on Systems, Man and cybernetics -Part B, Vol. 35, Issue 1, pp. 107 – 114, 2005

[27] D. Wang, Y. Ye and C. C Cheah , "*Analysis and Design of Anticipatory Learning Control*", Proceedings of 42nd IEEE Conference on Decision and Control, Vol. 5, pp. 4434- 4439, 2003

[28] M. Sun and D. Wang, "*Sampled -data iterative learning control for non linear systems with arbitrary relative degree*", Automatica(37), pp. 283-289,2001

[29] M. Sun and D. Wang, "*Iterative learning control with initial rectifying action*", Automatica (38), pp. 1177-1182, 2002

[30] M. Sun and D. Wang, "*Initial conditions issues on iterative learning control for linear systems with time delay*", International Journal of Systems Science, Vol.32, Issue 11, pp. 1365-1375, 2001

[31] S. Li, X. Xu and P. Li, "*Frequency Domain Analysis for Feedback-Assisted Iterative Learning Control*", Proceedings of 2004 International Conference on Information Acquisition, pp. 1-4, 2004

[32] C. J. Goh, "*A frequency domain analysis of learning control",* Journal of Dynamic systems, Measurement and Control, pp. 781-786, 1994

[33] Y. Chen, C. Wen, J. X. Xu and M. Sun., "*Initial state learning method for iterative learning control of uncertain time-varying systems",* Proceedings of 35th IEEE Conference on Decision and Control, pp. 3996-4001, 1996

[34] Y. Chen, C. Wen, Z. Gong and M. Sun., "*An iterative learning controller with initial state learning",* IEEE Transactions on Automatic Control, 44(2), pp. 371-376, 1999.

[35] M. Sun, D. Wang, and G. Xu, "*Initial shift problem and its ILC solution for nonlinear systems with higher relative degree*", American Control Conference, Vol. 1, pp. 277-281, 2000

[36] M. Sun, D. Wang, and Y. Wang, "*Reducing the effect of initial condition offsets using selective previous cycle data*", 8th International Conference on Control, Automation, Robotics and Vision (ICARCV),Vol. 1, pp. 613-618, 2004.

[37] J. E. Hauser, "*Learning control for a class of nonlinear systems*", Proceedings of the 26th IEEE Conference on Decision and Control, pp. 859- 860, 1987.

[38] G. Heinzinger, D. Fenwick, B. Paden, and F. Miyazaki, "*Stability of learning control with disturbances and uncertain initial conditions",* IEEE Transactions on Automatic Control, 37, pp. 110-114, 1992

[39] D. Wang and C.C. Cheah, "*An iterative learning control scheme for impedance control of robotic manipulators*", International Journal of Robotics Research, Vol. 17, pp. 1091-1104, 1998

[40] D. Wang, "*An anticipatory iterative learning control scheme: theory and experiments*", Proceedings of Iterative Learning Control Workshop and Roundtable, Tampa, FL, 1998

[41] M. Hideg, "*Stability of linear time varying multiple input multiple output continuous time learning control systems: a sufficient condition*", Proceedings of the 1994 IEEE International Symposium on Intelligent Control, pp. 285-290, 1994

[42] J. Xu and Y. Tan, "*New iterative learning control approaches for nonlinear non-affine MIMO dynamic systems",* American Control Conference, Vol. 2, pp. 896-901, 2001

[43] S.N. Huang, K.K. Tan, and T.H. Lee, "*Necessary and sufficient condition for convergence of iterative learning algorithm",* Automatica(38), pp. 1257-1260, 2002

[44] Y. Ye and D. Wang, *"Non-causal filtering based MIMO learning control laws*", 4th International Conference on Control and Automation, pp. 481-485, 2003.

[45] T.Y. Doh and J.H. Moon, "*Feedback-based iterative learning control for uncertain linear MIMO systems",* 5th Asian Control Conference, Vol. 1, pp. 198-203, 2004

[46] P. Jiang and H. Chen, "*Iterative learning control of MIMO systems with less model knowledge*", IEEE Conference on Robotics, Automation and Mechatronics, pp. 490-495, 2004

[47] A. Tayebi, "*On ILC design for MIMO-LTI systems*", American Control Conference, pp. 940-945, 2006

[48] A. Tayebi, "*Analysis of two particular iterative learning control schemes in frequency and time domains*", Automatica, 43(9), pp. 1565-1572, 2007

[49] T.Y. Doh and J.R. Ryoo, "*Feedback-based iterative learning control for MIMO LTI systems*", International Journal of Control, Automation and Systems, pp. 269-277, 2008

[50] T. Sugie and F. Sakai, "*Projection based iterative learning control with its application to continuous-time system identification*", 2nd International Conference on Informatics Research for Development of Knowledge Society Infrastructure, pp. 95-102, 2007

[51] M.C. Campi, T. Sugie and F. Sakai, "*Iterative identification method for linear continuous-time systems*", 45th IEEE Conference on Decision and Control 2006, pp. 817-822, 2006

[52] M.C Campi, T. Sugie and F. Sakai, "*An iterative identification method for linear continuous-time systems*", IEEE Transactions on Automatic Control, 53(7), pp. 1661-1669, 2008

[53] F. Sakai and T. Sugie, "*A continuous-time closed-loop identification method based on iterative learning control concepts*", 46th IEEE Conference on Decision and Control, pp. 4906-4911, 2008

[54] T. Sugie, "*Identification of linear continuous-time systems based on iterative learning control*", Lecture Notes in Control and Information Sciences, 371, pp. 205-218, 2008

[55] T. Sugie and F. Sakai, "*Noise tolerant iterative learning control for identification of continuous-time systems*", 44th IEEE Conference on Decision and Control, and the European Control Conference, pp. 4251-4256, 2005

[56] T.H. Kim and T. Sugie, "*An iterative learning control based identification for a class of MIMO continuous-time systems in the presence of fixed input disturbances and measurement noises*", International Journal of Systems Science, 38(9), pp. 737-748, 2007

[57] M. Norrlof, "*Iterative Learning Control: Analysis, Design, and Experiments*", PhD thesis, Linkopings universitet, Linkoping, Sweden, 2000 , Linkoping Studies in Science and Technology, Dissertations; 653, Download from http://www.control.isy.liu.se/publications/

[58] S. Arimoto, " *Learning control theory for robotic motion*", International Journal of Adaptive Control and Signal processing, Vol. 4, pp. 543-564, 1990

[59] J. X. Xu and Y. Tan, "*Robust optimal design and convergence properties analysis of iterative learning control approaches*", Automatica, Volume 38, Issue 11, pp. 1867-1880, 2002

[60] J. X. Xu and Y. Tan, *"Iterative Learning Control Design without a Priori Knowledge of Control Directions"*, Proceedings of American Control Conference, Vol. 4, pp. 3661-3666, 2003.

[61] M. C. Campi, T. Sugie and F. Sakai, "*Iterative identification method for linear-continuous time systems*", Proceedings of 45th IEEE Conference on Decision and Control, pp. 817-822, 2006.

[62] X. D. Li, T. W. S. Chow, J. K. L. Ho, J. Zhang, *"Iterative learning control with initial rectifying action for nonlinear continuous systems"* , Proceedings of IET Control Theory and Applications, 2008

[63] J. X. Xu and Z. Bien , "*Iterative learning control: Analysis, design, integration and applications",* Book, Kluwer Academic Publishers, Norwell MA, USA, pp. 3-140, 1998

[64] C. Wenjian, "*Design of Multivariable controllers*", Lecture Notes, Part 4, Process Control Course, pp. 1-25, Department of Electrical and Electronic Engineering, Nanyang Technological University, 2008

[65] S. Arimoto, S. Kawamura, and F. Miyazaki, "*Convergence, stability and robustness of learning control schemes for robot manipulators*", In M. J. Jamishidi, L. Y. Luh, and M. Shahinpoor, editors, *Recent Trends in Robotics: Modelling, Control, and Education*, pp. 307–316. Elsevier, 1986.

[66] G. Heinzinger, D. Fenwick, B. Paden and F. Miyazaki, *"Stability of Learning Control with Disturbances and Uncertain Initial Conditions",* IEEE transactions on Automatic Control, Vol. 37, No. 1, 1992

[67] A. De Luca, G. Paesano and G. Ulivi, "*A frequency-domain approach to learning control: implementation for a robot manipulator*", Industrial Electronics, IEEE Transactions on Volume 39, Issue 1, pp. 1 – 10, 1992

[68] J. X. Xu and Y. Tan, *"Linear and Nonlinear Iterative Learning Control",* Springer, URL: www.ece.nus.edu.sg/stfpage/elexujx/PDF/Chap1.pdf

[69] The Math Works Inc., "*Dynamic System simulation for MATLAB"*, Edition 3, 2001, "*Writing S-functions*" and "*Using SIMULINK*" (2 books supplement main book)

[70] V. Giurgiutiu and S. E. Lyshevski, "*Mechatronics - Modeling, Analysis and Design with MATLAB*", CRC press, 1998

[71] dSPACE Inc. , "*dSpace Manuals*", release 5.2, 2005

[72] N. E. Leonard and W.S. Levine , " *Using MATLAB to analyze and design Control systems*",

Appendix

A. Basic theory with respect to repetition domain

MIMO system representation in repetition domain is derived in [63] and concept of ILC for MIMO systems is explained with respect to repetition domain. Consider a time-invariant discrete-time MIMO system with disturbance,

$$x(k+1) = Ax(k) + Bu(k) + w_s(k) \tag{A.1}$$

$$y(k) = Cx(k) + w_y(k)$$

where, A, B & C are system matrices. Vectors $x(k), u(k)$ & $y(k)$ are state, control input and output vectors, respectively. Let q be number of samples and $k = 0,1, \ldots\ldots, q-1$. From Eq A.1, we can generate matrix P such that,

$$\underline{y} = P\underline{u} + \underline{w} \tag{A.2}$$

$$\underline{yy} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(p) \end{bmatrix}, \quad \underline{u} = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(p-1) \end{bmatrix}, \quad \underline{w} = \begin{bmatrix} w(1) \\ w(2) \\ \vdots \\ w(p) \end{bmatrix} \quad P = \begin{bmatrix} CB & 0 & \ldots & 0 \\ CAB & CB & \ldots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ CA^{p-1}B & \cdots & \ldots & CB \end{bmatrix} \tag{A.3}$$

If iteration number is j, backward difference operator applied to any variable $z(k)$ at time step k, can be defined as, $\boldsymbol{\delta_j z(k) = z(j) - z(j-1)}$. Applying backward difference operator to Eq A.2,

$$\delta_j \underline{y} = P\delta_j \underline{u} \tag{A.4}$$

$$\underline{y_j} = \underline{y_{(j-1)}} + P\delta_j \underline{u} \tag{A.5}$$

Eq A.5 is repetition domain representation of system. Repetition domain representation is repetition-invariant, even if system is time-variant. Repetitive disturbances are eliminated, i.e., system representation in repetition domain is independent of disturbances, if disturbances are repetitive. Assuming a learning control with linear feedback in repetition domain of form,

$$\delta_j \underline{u} = L\underline{e}_{(j-1)} \tag{A.6}$$

$$\underline{e}_j = y_d - y_j \tag{A.7}$$

Using Eq A.4, Eq A.5, Eq A.6 and A.7, error dynamics in repetition domain is,

$$\underline{e}_j = (I - PL)\underline{e}_{(j-1)} \qquad [63] \tag{A.8}$$

For a MIMO system to exhibit proper convergence eigenvalues of $(I - PL)$ must have the magnitude less than one. Asymptotic stability in repetition domain guarantees tracking error convergence in time-domain.

B. Tracking error convergence in time-domain for MIMO system

Consider a continuous time MIMO system which is represented as,

$$\dot{x}_k(t) = A(t)x_k(t) + B(t)u_k(t) \tag{B.1}$$

$$y_k(t) = C(t)x_k(t)$$

where k is the iteration number. Our objective is to design an iterative learning rule for MIMO systems which can generate the control input $u_k(t)$, such that system output $y_k(t)$ converges to desired output $y_d(t)$, when iteration k goes to infinity for all $t \in [0, T]$. To simplify the discussion, P-type ILC law used to explain ILC design on MIMO system $\boldsymbol{G_p(s)}$.

P-type update law is expressed as,

$$u_{k+1}(t) = u_k(t) + L\big(y_d(t) - y_k(t)\big) \tag{B.2}$$

$$e_k(t) = y_d(t) - y_k(t)$$

where $\boldsymbol{t \in [0, t_f]}$, $u_k(t)$ *and* $y_k(t)$ are the rx*1 dimension* input vector and px*1 dimensional* output vector respectively, with r being the number of inputs and p being the number of outputs. L is rxp dimensional learning matrix. $e_k(t)$ is px*1* dimensional error matrix. The input and output matrix are defined as

$$u_k = \begin{bmatrix} u_{1_k} \\ u_{2_k} \\ \vdots \\ u_{r_k} \end{bmatrix}, y_k = \begin{bmatrix} y_{1_k} \\ y_{2_k} \\ \vdots \\ y_{p_k} \end{bmatrix} \; and \; L = \begin{bmatrix} l_{11} & \cdots & l_{1p} \\ l_{21} & & l_{2p} \\ \vdots & \ddots & \vdots \\ l_{r1} & \cdots & l_{rp} \end{bmatrix} \tag{B.3}$$

So the update law (P-type) in this case is

$$\begin{bmatrix} u_{1_{(k+1)}}(t) \\ u_{2_{(k+1)}}(t) \\ \vdots \\ u_{r_{(k+1)}}(t) \end{bmatrix} = \begin{bmatrix} u_{1_k}(t) \\ u_{2_k}(t) \\ \vdots \\ u_{r_k}(t) \end{bmatrix} + \begin{bmatrix} l_{11} & \cdots & l_{1p} \\ l_{21} & & l_{2p} \\ \vdots & \ddots & \vdots \\ l_{r1} & \cdots & l_{rp} \end{bmatrix} \begin{bmatrix} y1_d - y_{1_k} \\ y2_d - y_{2_k} \\ \vdots \\ yd_p - y_{p_k} \end{bmatrix} \tag{B.4}$$

Consider the following condition,

$$x_{k+1}(t) - x_k(t) = \left(\int_0^t \dot{x}_{k+1}(\tau)d\tau - \int_0^t \dot{x}_k(\tau)d\tau \right) + x_{k+1}(0) - x_k(0) \tag{B.5}$$

Since the initial resetting conditions are same,

$$x_{k+1}(0) = x_k(0) = x_0 \tag{B.6}$$

$$\dot{x}_k(t) = A(t)x_k(t) + B(t)u_k(t)$$

Using Eq B.6 and B.7, Eq B.5 reduces to,

$$x_{k+1}(t) - x_k(t) = \left(\int_0^t A(\tau)\big(x_{k+1}(\tau) - x_k(\tau)\big)d\tau + \int_0^t B(\tau)(u_{k+1}(\tau) - u_k(\tau))d\tau \right) \tag{B.7}$$

We know that,

$$u_{k+1}(t) = u_k(t) + Le_k(t) \tag{B.8}$$

Substituting Eq B.8 in Eq B.7,

$$\boldsymbol{x_{k+1}(t) - x_k(t) = \left(\int_0^t A(\tau)\big(x_{k+1}(\tau) - x_k(\tau)\big)d\tau + \int_0^t B(\tau)L(\tau)e_k(\tau)d\tau \right)} \tag{B.9}$$

Applying norm to Eq B.9,

$$\boldsymbol{\|x_{k+1}(t) - x_k(t)\| \leq \left(L_a \int_0^t \|x_{k+1}(\tau) - x_k(\tau)\|d\tau + \int_0^t \|B(\tau)L(\tau)e_k(\tau)\|d\tau \right)} \tag{B.10}$$

where $L_a = sup_{\ t\in[0,T]}\ \|A(t)\|$. Consider,

$$\int_0^t \|B(\tau)L(\tau)e_k(\tau)\|d\tau = \left\| \big(B(t)L(t)e_k(t) - B(0)L(0)e_k(0)\big) - \int_0^t \frac{d}{d\tau}\big(B(\tau)L(\tau)\big)e_k(\tau)d\tau \right\| \tag{B.11}$$

Since $e_k(0) = 0$, Initial conditions are same.

$$\int_0^t \|B(\tau)L(\tau)e_k(\tau)\|d\tau = \left\| B(t)L(t)e_k(t) - \int_0^t \frac{d}{d\tau}\big(B(\tau)L(\tau)\big)e_k(\tau)d\tau \right\| \tag{B.12}$$

$$= L_1\|e_k(t)\| - L_2\int_0^t \|e_k(\tau)\|d\tau$$

Where $L_1 = sup_{\ t\in[0,T]}\ \|B(t)L(t)\|$ and $L_2 = sup_{\ t\in[0,T]}\ \left\|\frac{d}{d\tau}\big(B(\tau)L(\tau)\big)\right\|$

Substitute Eq B.12 in Eq B.11,

$$\|x_{k+1}(t) - x_k(t)\| \leq L_a \int_0^t \|x_{k+1}(\tau) - x_k(\tau)\| d\tau + L_1 \|e_k(t)\| - L_2 \int_0^t \|e_k(\tau)\| d\tau \tag{B.13}$$

Multiply Eq. B.13 by $e^{-\lambda t}$,

$$\|\boldsymbol{x_{k+1}(t) - x_k(t)}\|_\lambda \tag{B.14}$$

$$\leq \boldsymbol{L_a sup}_{\ t\in[0,T]} \left(\boldsymbol{e^{-\lambda t} \int_0^t \|x_{k+1}(t) - x_k(t)\| \, d\tau} \right)$$

$$+ \boldsymbol{L_1 \int_0^t \|e_k(\tau)\|_\lambda d\tau - sup}_{\ t\in[0,T]} \left(\boldsymbol{e^{-\lambda t} \int_0^t \|e_k(\tau)\| \, d\tau} \right)$$

Consider equation, $e_{k+1}(t) = y_d(t) - y_{k+1}(t)$ (B.15)

Eq B.15 can be rewritten as, $e_{k+1}(t) - e_k(t) = y_k(t) - y_{k+1}(t)$

From Eq B.1, $$e_{k+1}(t) - e_k(t) = -C(t)\big(x_{k+1}(t) - x_k(t)\big) \tag{B.16}$$

$$e_{k+1}(t) = e_k(t) - C(t)\big(x_{k+1}(t) - x_k(t)\big) \tag{B.17}$$

Substituting Eq B.9 in Eq B.17,

$$e_{k+1}(t) = e_k(t) - C(t)\left(\int_0^t A(\tau)\big(x_{k+1}(\tau) - x_k(\tau)\big) d\tau + \int_0^t B(\tau)L(\tau)e_k(\tau) d\tau \right) \tag{B.18}$$

$$e_{k+1}(t) = e_k(t) - \left(C(t)B(t)L(t)e_k(t) - C(t) \int_0^t \frac{d}{d\tau}(B(\tau)L(\tau))e_k(\tau) d\tau \right) \tag{B.19}$$

$$- C(t)\left(\int_0^t A(\tau)\big(x_{k+1}(\tau) - x_k(\tau)\big) d\tau \right)$$

$$\boldsymbol{e_{k+1}(t) = \big(I - C(t)B(t)L(t)\big)e_k(t) + C(t) \int_0^t \frac{d}{d\tau}\big(B(\tau)L(\tau)\big)e_k(\tau) d\tau} \tag{B.20}$$

$$\boldsymbol{- C(t)\left(\int_0^t A(\tau)\big(x_{k+1}(\tau) - x_k(\tau)\big) d\tau \right)}$$

Applying norm to Eq B.20,

$$\|e_{k+1}(t)\| = \rho^0\|e_k(t)\| + L_cL_2\int_0^t\|e_k(\tau)d\tau\| - L_cL_1\left(\int_0^t\|x_{k+1}(\tau) - x_k(\tau)\|d\tau\right) \tag{B.21}$$

Where, $L_c = sup_{\ t\in[0,T]}\ \|C(t)\|$

$$\|e_{k+1}(t)\| = \rho^0\|e_k(t)\| + L_cL_2\int_0^t\|e_k(\tau)d\tau\| - L_cL_1\left(\int_0^t\|x_{k+1}(\tau) - x_k(\tau)\|d\tau\right) \tag{B.22}$$

Multiply Eq B.22 by $e^{-\lambda t}$

$$\|\boldsymbol{e_{k+1}(t)}\|_\lambda \leq \boldsymbol{\rho^0}\|\boldsymbol{e_k(t)}\|_\lambda + \boldsymbol{L_cL_2}\,\boldsymbol{sup}_{\ t\in[0,T]}\left(\boldsymbol{e^{-\lambda t}}\int_0^t\|\boldsymbol{e_k(\tau)}\|\boldsymbol{d\tau}\right) \tag{B.23}$$

$$-\boldsymbol{L_cL_1}\,\boldsymbol{sup}_{\ t\in[0,T]}\left(\boldsymbol{e^{-\lambda t}}\int_0^t\|\boldsymbol{x_{k+1}(\tau) - x_k(\tau)}\|\boldsymbol{d\tau}\right)$$

Applying lemma B.1 to Eq B.14 and B.23,

$$sup_{\ t\in[0,T]}\left(e^{-\lambda t}\int_0^t\|x(\tau)\|d\tau\right) \leq \frac{1}{\lambda}\|x(\tau)\|_\lambda \tag{lemma B.1}$$

$$\|x_{k+1}(t) - x_k(t)\|_\lambda \leq \frac{L_a}{\lambda}\|x_{k+1}(t) - x_k(t)\|_\lambda + L_1\|e_k(\tau)\|_\lambda - \frac{L_2}{\lambda}\|e_k(\tau)\|_\lambda \tag{B.24}$$

$$\|x_{k+1}(t) - x_k(t)\|_\lambda \leq \frac{L_a}{\lambda}\|x_{k+1}(t) - x_k(t)\|_\lambda + \frac{\lambda L_1 - L_2}{\lambda}\|e_k(t)\|_\lambda \tag{B.25}$$

$$\|e_{k+1}(t)\|_\lambda \leq \rho^0\|e_k(t)\|_\lambda + \frac{L_cL_2}{\lambda}\|e_k(t)\|_\lambda - \frac{L_cL_1}{\lambda}\|x_{k+1}(t) - x_k(t)\|_\lambda \tag{B.26}$$

$$\|e_{k+1}(t)\|_\lambda \leq \frac{\rho^0\lambda + L_cL_2}{\lambda}\|e_k(t)\|_\lambda - \frac{L_cL_1}{\lambda}\|x_{k+1}(t) - x_k(t)\|_\lambda \tag{B.27}$$

Simplifying Eq B.26, Eq B.25, Eq B.27 and Eq B.24, we arrive at,

$$\|\boldsymbol{e_{k+1}(t)}\|_\lambda \leq \boldsymbol{\rho}\|\boldsymbol{e_k(t)}\|_\lambda \tag{B.28}$$

Where $\rho = \frac{\rho^0\lambda + L_cL_2}{\lambda} - \frac{\lambda L_1 - L_2}{\lambda L_a}$

If $\rho < 1$, Eq B.28 exhibits effective convergence of tracking error. We can select the value of λ to be sufficiently large and $\rho^0 < 1$, in order to guarantee that $\rho < 1$

C. Steps to choose lead time(Δ) in A-type ILC

Convergence and robustness conditions of A-type ILC are derived in chapter 4 (section 4.2.2 and 4.2.3). Based on convergence and robustness analysis, A-type ILC is implemented on T2-axis of SCARA robot. In this section, steps to choose the lead time of A-type ILC is explained in detail.

Plant model can be expressed in frequency domain as,

$$\boldsymbol{G_p(j\omega) = N_p(\omega)e^{jN_p(\omega)}} \tag{C.1}$$

where $N_p(\omega)$ is magnitude characteristic and $\theta_p(\omega)$ is phase characteristic of plant

A typical update law of A-type ILC is

$$u_{k+1} = u_k + \varphi(y_d(t+\Delta) - y_k(t+\Delta)) \tag{C.2}$$

where Δ is the lead time, φ is the learning gain, $y_d(t)$ is desired trajectory, $u_k(t)$ is control input and $y_k(t)$ is output in iteration k.

Referring to Equations 4.24, 4.25 and 4.29 in sections 4.2.2 and 4.2.3,

Convergence and robustness conditions for A-type ILC

$$\boldsymbol{\varphi N_p(\omega) < 2\cos\left(\theta_p(\omega) + \Delta\omega\right)} \tag{C.3}$$

$$\boldsymbol{\left|\theta_p(\omega) + \Delta\omega\right| < 90^0 - \epsilon} \tag{C.4}$$

where φ learning gain of ILC. We assume $\epsilon = 10^0$ to provide robustness to the system

In order to ensure that learning algorithm exhibits convergence, equations C.3 and C.4 have to be satisfied. From equation C.4, we can deduce value of lead time (Δ).

From equation, 6.54 in section 6.6, the system model (T2-axis) obtained by system identification is

$$G_{T2}(s) = \frac{22.05}{s^2 + 6.074s + 22.41} \tag{C.5}$$

Using equation C.5 and the plot of "$\theta_p(\omega) + \Delta\omega$ versus frequency" (figure C.1), lead time is chosen to maximize the frequency ω.

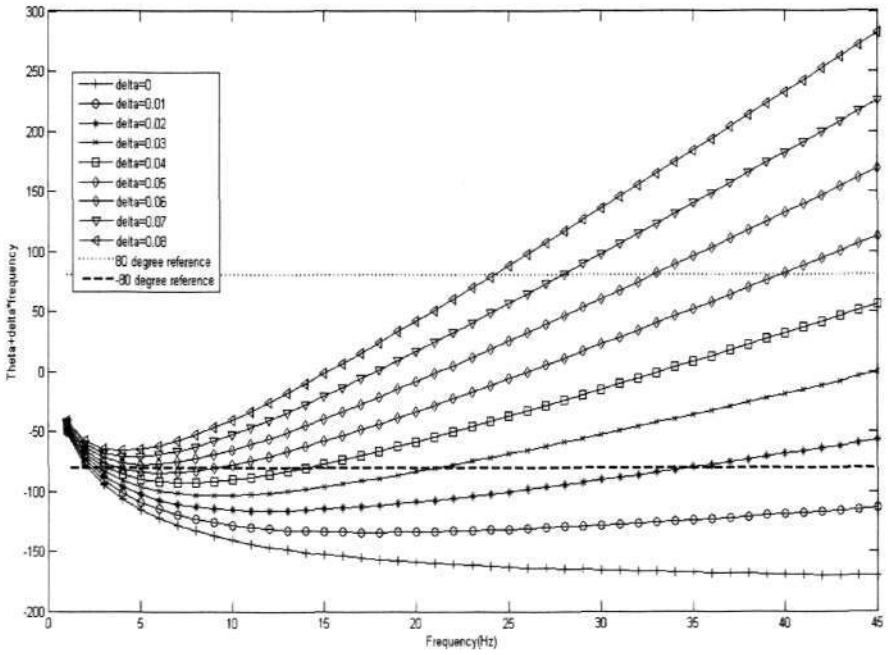


Figure C.1: Plot of $\theta_p(\omega) + \Delta\omega$ versus frequency to choose lead time

From figure C.1, we observe that, at lead time (Δ) = 0.04 sec,

$$-80^0 < \theta_p(\omega) + \Delta\omega < 80^0$$

Because from equation C.4,

$$\left|\boldsymbol{\theta_p(\omega) + \Delta\omega}\right| < \mathbf{90^0} - \epsilon \qquad \text{(C.6)}$$

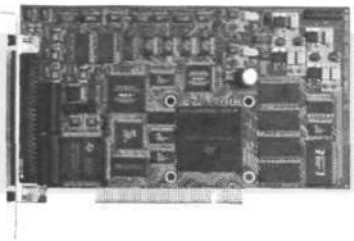
$$\left|\boldsymbol{\theta_p(\omega) + \Delta\omega}\right| < \mathbf{80^0}$$

Hence we choose lead time (Δ) to be 0.04 sec

D. Data sheets of DS1104, 25A8 power amplifier and Interface board

DS1104 R&D Controller Board

Cost-effective system for controller development



Key Features

- **Single-board PCI hardware for use in PCs**
- **Set of intelligent I/O on-board**
- **Incremental encoder interface**
- **Serial interface (UART)**

Description

Application Areas

The real-time hardware based on PowerPC technology and its set of I/O interfaces make the controller board an ideal solution for developing controllers in various fields, such as drives, robotics, aerospace and automotives.

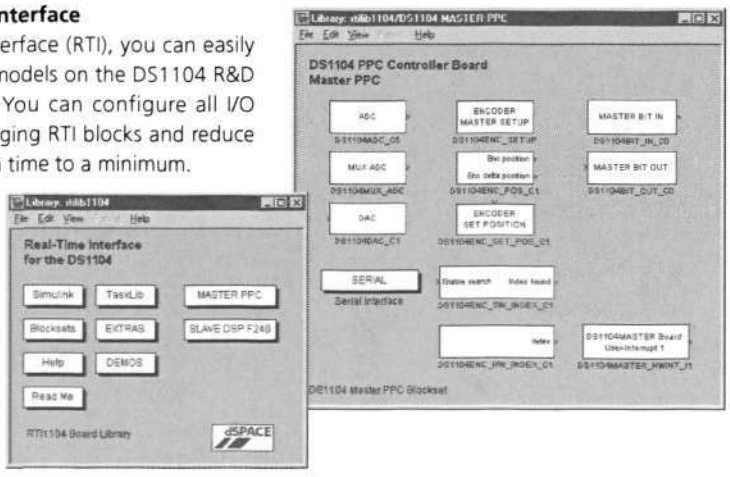
Key Benefits

The DS1104 upgrades your PC to a powerful development system for rapid control prototyping („R&D" stands for research & development). Real-Time Interface provides Simulink® blocks for graphical configuration of A/D, D/A, digital I/O lines, incremental encoder interface and PWM generation, for example. The board can be installed in virtually any PC with a free 5-V PCI slot.

Real-Time Interface

Using Real-Time Interface

With Real-Time Interface (RTI), you can easily run your function models on the DS1104 R&D Controller Board. You can configure all I/O graphically by dragging RTI blocks and reduce the implementation time to a minimum.



Technical Details

Parameter		Specification
Processor	PowerPC Type	■ PPC 603e
	CPU clock	■ 250 MHz
	Cache	■ 2 x 16 KB
Memory	Global memory	■ 32 MB SDRAM
	Flash memory	■ 8 MB
Timer	4 general-purpose timers	■ 32-bit down counter ■ Reload by hardware ■ 80-ns resolution
	1 sampling rate timer (decrementer)	■ 32-bit down counter ■ Reload by software ■ 40-ns resolution
	1 time base counter	■ 64-bit up counter ■ 40-ns resolution
Interrupt controller		■ 5 timer interrupts ■ 2 incremental encoder index line interrupts ■ 1 UART interrupt ■ 1 slave DSP interrupt ■ 1 slave DSP PWM interrupt ■ 5 A/D converter (end of conversion) interrupts ■ 1 host interrupt ■ 4 external interrupts (user interrupts)
A/D converter	Channels	■ 4 multiplexed channels equipped with one sample & hold A/D converters ■ 4 parallel channels each equipped with one sample & hold A/D converter
	Resolution	■ Multiplexed channels: 16 bit ■ Parallel channels: 12 bit
	Input voltage range	■ ±10 V
	Conversion time	■ Multiplexed channels: 2 µs [1] ■ Parallel channels: 800 ns [1]
	Offset error	■ ±5 mV
	Gain error	■ Multiplexed channels: ±0.25% ■ Parallel channels: ±0.5%ing
	Offset drift	■ 4 ppm/K
	Gain drift	■ 25 ppm/K
	Signal-to-noise ratio	■ Multiplexed channels: >80 dB ■ Parallel channels: >65 dB
D/A converter	Channels	■ 8 channels
	Resolution	■ 16-bit
	Output range	■ ±10 V
	Settling time	■ Max. 10 µs (full-scale, accuracy ½ LSB)
	Offset error	■ ±1 mV
	Gain error	■ ±0.1%
	Offset drift	■ 13 ppm/K
	Gain drift	■ 25 ppm/K
	Signal-to-noise ratio	■ >80 dB
	I_{max}	■ ±5 mA
Digital I/O	Channels	■ 20-bit parallel I/O ■ Single bit selectable for input or output
	Voltage range	■ TTL input/output levels
	I_{outmax}	■ ±5 mA

[1] Speed and timing specifications describe the capabilities of the hardware components and circuits of our products. Depending on the software complexity, the attainable overall performance figures can deviate significantly from the hardware specifications.

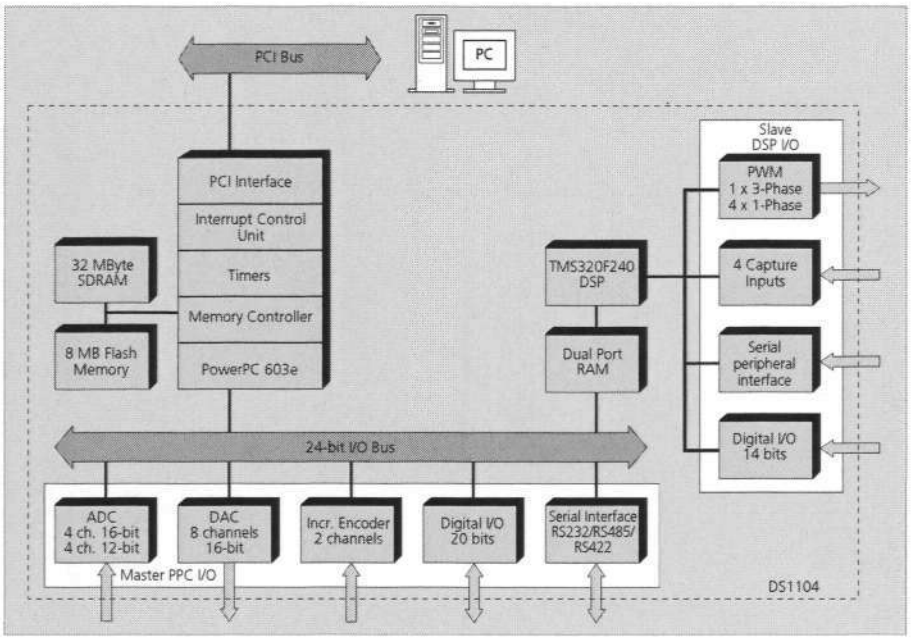
Parameter		Specification
Digital incremental encoder interface	Channels	■ 2 independent channels ■ Single-ended (TTL) or differential (RS422) input (software programmable for each channel)
	Position counters	■ 24-bit resolution ■ Max. 1.65 MHz input frequency, i.e., fourfold pulse count up to 6.6 MHz ■ Counter reset or reload via software
	Sensor supply voltage	■ 5 V/0.5 A
Serial interface	Configuration	■ Single UART (universal asynchronous receiver and transmitter) with FIFO ■ PLL-driven UART for accurate baud rate selection ■ RS232/RS422/RS485 compatibility
	Baud rate	■ Up to 115.2 Kbaud (RS232) ■ Up to 1 Mbaud (RS422/RS485)
Slave DSP	Type	■ Texas Instruments TMS320F240 DSP
	Clock rate	■ 20 MHz
	Memory	■ 64K x 16 external code memory ■ 28K x 16 external data memory ■ 4K x 16 dual-port memory for communication ■ 32 KB flash memory
	I/O channels	■ 10 PWM outputs ■ 4 capture inputs ■ 1 serial peripheral interface
	Input voltage range	■ TTL input/output level ■ A/D converter inputs: 0 ... 5 V
	Output current	■ Max. ±13 mA
Host interface		■ Requires one 33 MHz / 32-bit 5-V PCI slot
Physical characteristics	Physical size	■ 178 x 107 mm (7.0 x 4.2 in)
	Ambient temperature	■ 0 ... 55 °C (32 ... 131 °F)
	Cooling	■ Active cooling by fan
	Power consumption	■ 18.5 W
	Power supply	■ +5 V ±5%, 2.5 A ■ +12 V ±5%, 0.3 A ■ -12 V ±5%, 0.2 A

Order Information

Product	Order Number
DS1104 R&D Controller Board	■ DS1104

Software		Order Number
Included	■ DS1104 Real-Time Library	–
	■ Experiment and Platform Manager for hardware management	–
Required	■ Microtec C Compiler (p. 139)	■ CCPPPC
	■ Real-Time Interface (p. 120)	■ RTI
Optional	■ ControlDesk Standard – Operator Version (p. 140)	■ CS_O
	■ ControlDesk Standard – Developer Version (p. 140)	■ CS_D
	■ MLIB/MTRACE (p. 178)	■ MLIB/MTRACE
	■ CLIB (p. 177)	■ CLIB

Hardware		Order Number
Optional	■ Connector Panel (p. 260)	■ CP1104
	■ Combined Connector/LED Panel (p. 260)	■ CLP1104



Block Diagram

Induction Motor Control

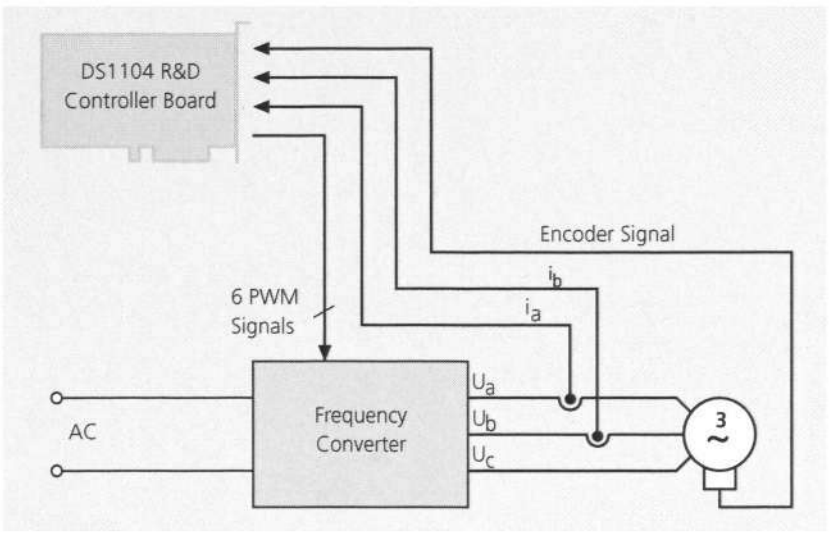
Drive Control

In this use case, an induction motor controller is developed with the DS1104. The slave DSP system was designed for applications in drive control, and the PowerPC's calculation power supports convenient simulation and a smooth development process. In combination with Simulink, the board makes it easy to verify and optimize control algorithms and parameters.

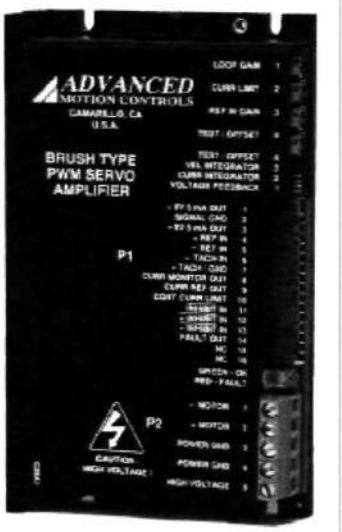
Determining Values

One of the board's incremental encoder interfaces picks up the encoder signal of the motor, while two A/D converters are required to analyze the motor currents. The controller board calculates the control algorithm on the basis of the measured values and determines the corresponding pulse width modulation (PWM). The three-phase PWM signals are generated on the board's DSP subsystem and determine the converter's output voltage and frequency.

Use Case



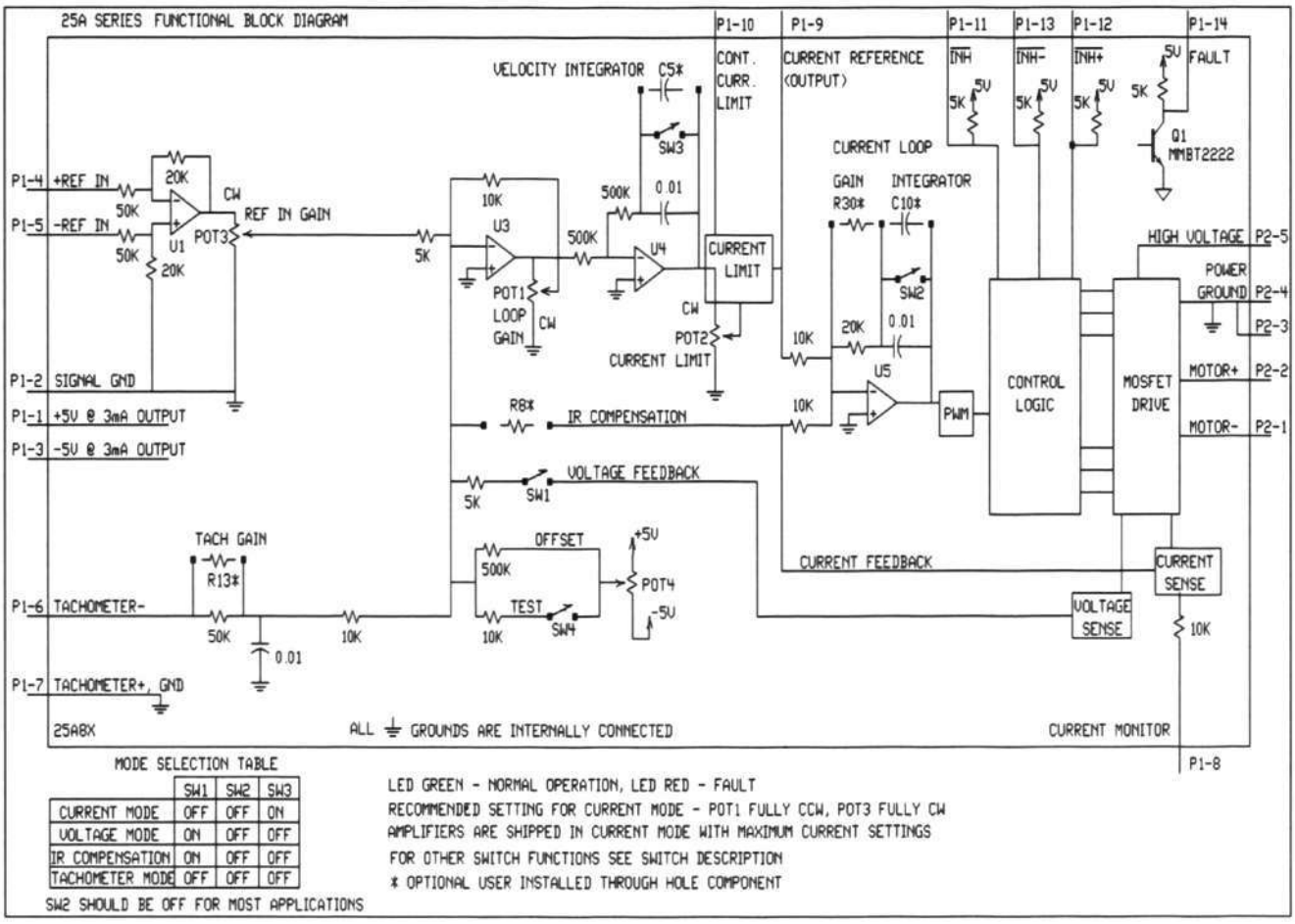
SERIES 25A SERVO AMPLIFIERS
Models: 12A8, 25A8, 20A14, 20A20
Miniature Series



FEATURES:

- Surface-mount technology
- Small size, low cost, ease of use
- DIP switch selectable: current, voltage, velocity, analog position loop
- Four quadrant regenerative operation
- Agency Approvals:

BLOCK DIAGRAM:



MODE SELECTION TABLE

	SW1	SW2	SW3
CURRENT MODE	OFF	OFF	ON
VOLTAGE MODE	ON	OFF	OFF
IR COMPENSATION	ON	OFF	OFF
TACHOMETER MODE	OFF	OFF	OFF

SW2 SHOULD BE OFF FOR MOST APPLICATIONS

LED GREEN - NORMAL OPERATION, LED RED - FAULT
RECOMMENDED SETTING FOR CURRENT MODE - POT1 FULLY CCW, POT3 FULLY CW
AMPLIFIERS ARE SHIPPED IN CURRENT MODE WITH MAXIMUM CURRENT SETTINGS
FOR OTHER SWITCH FUNCTIONS SEE SWITCH DESCRIPTION
* OPTIONAL USER INSTALLED THROUGH HOLE COMPONENT

ADVANCED MOTION CONTROLS
3805 Calle Tecate, Camarillo, CA 93012 Tel: (805) 389-1935, Fax: (805) 389-1165

DESCRIPTION: The 25A Series PWM servo amplifiers are designed to drive brush type DC motors at a high switching frequency. A single red/green LED indicates operating status. All models are fully protected against over-voltage, over-current, over-heating and short-circuits across motor, ground and power leads. All models interface with digital controllers or can be used as a stand-alone drive. They require only a single unregulated DC power supply. Loop gain, current limit, input gain and offset can be adjusted using 14-turn potentiometers. The offset adjusting potentiometer can also be used as an on-board input signal for testing purposes when SW4 (DIP switch) is ON.

SPECIFICATIONS:

	MODELS			
POWER STAGE SPECIFICATIONS	**12A8**	**25A8**	**20A14**	**20A20**
DC SUPPLY VOLTAGE	20 - 80 V	20 - 80 V	40 - 140 V	40 - 190 V
PEAK CURRENT (2 sec. Max., internally limited)	± 12 A	± 25 A	± 20 A	± 20 A
MAXIMUM CONTINUOUS CURRENT (internally limited)	± 6 A	± 12.5 A	± 10 A	± 10 A
MINIMUM LOAD INDUCTANCE*	200 µH	200 µH	250 µH	250 µH
SWITCHING FREQUENCY	36 kHz	22 kHz ± 15%		
HEATSINK (BASE) TEMPERATURE RANGE	0 ° to + 65°C, disables if > 65°C			
POWER DISSIPATION AT CONTINUOUS CURRENT	24 W	50 W	70 W	100 W
OVER-VOLTAGE SHUT-DOWN (self-reset)	86 V	86 V	142 V	195 V
BANDWIDTH (load dependent)	2.5 kHz			

MECHANICAL SPECIFICATIONS	
POWER CONNECTOR	Screw terminals
SIGNAL CONNECTOR	Molex connector
SIZE	5.09 x 2.98 x 0.99 inches 129.3 x 75.8 x 25.1 mm
WEIGHT	10 oz. 0.28 kg

* Low inductance motors ("pancake" and "basket-wound") require external inductors.

PIN FUNCTIONS:

CONNECTOR	PIN	NAME	DESCRIPTION / NOTES	I/O
P1	1	+5V OUT	Internal DC-to-DC converter, outputs regulated voltages of ±5 V @ 3 mA for customer use. Short circuit protected.	O
	2	SIGNAL GND		GND
	3	-5V OUT		O
	4	+REF IN	Differential analog input, maximum ±15 V, 50K input resistance.	I
	5	-REF IN		
	6	-TACH IN	Maximum ± 60 VDC, 60K input resistance.	I
	7	+TACH (GND)		
	8	CURRENT MONITOR OUT	This signal is proportional to the actual current in the motor leads. Scaling is 2A/V for 12A8 and 4 A/V for 25A8, 20A14 and 20A20.	O
	9	CURRENT REFERENCE OUT	Command signal to the internal current-loop. The maximum peak current rating of the amplifier always equals 7.25V at this pin. See current limit adjustment information below.	O
	10	CONTINUOUS CURRENT LIMIT	Can be used to reduce the factory-preset maximum continuous current limit.	I
	11	INHIBIT	This TTL level input signal turns off all four power devices of the "H" bridge drive when pulled to ground. This inhibit will cause a FAULT condition and a red LED. For inverted inhibit inputs; see section "G".	I
	12	+INHIBIT	Disables the amplifier for the "+" direction only. This inhibit will not cause a FAULT condition or a red LED.	I
	13	-INHIBIT	Disables the amplifier for the "-" direction only. This inhibit will not cause a FAULT condition or a red LED.	I
	14	FAULT OUT (red LED)	TTL compatible output. It becomes high during output short-circuit, over-voltage, over-heating, inhibit, and during "power-on reset". Fault condition indicated by a red LED.	O
	15	NC	Not connected	
	16			
P2	1	-MOTOR	Motor minus connection.	O
	2	+MOTOR	Motor plus connection.	O
	3	POWER GROUND	Power Ground.	GND
	4	POWER GROUND	Power Ground.	GND
	5	HIGH VOLTAGE	DC voltage input.	I

SWITCH FUNCTIONS:

SWITCH	FUNCTION DESCRIPTION	SETTING	
		ON	OFF
1	Internal voltage feedback	On	Off
2	It is recommended to leave SW2 in OFF position.	Shorts out the current loop integrator capacitor.	Current loop integrator operating
3	This capacitor normally ensures "error-free" operation by reducing the error-signal (output of summing amplifier) to zero.	Shorts out the outer velocity/voltage loop integrator capacitor	Velocity/Voltage integrator operating
4	Offset / test. Controls sensitivity of the "offset" pot. Used as an on-board reference signal in test mode.	Test	Offset

POTENTIOMETER FUNCTIONS:

POTENTIOMETER	DESCRIPTION	TURNING CW
Pot 1	Loop gain adjustment in voltage & velocity modes. Turn this pot fully ccw in current mode.	Increases loop gain
Pot 2	Current limit. It adjusts both continuous and peak current limit by maintaining their ratio (50 %).	Increases current limit
Pot 3	Reference gain. It adjusts the ratio between input signal and output variables (voltage, current, or velocity).	Increases reference input gain
Pot 4	Offset / test. Used to adjust any imbalance in the input signal or in the amplifier. When SW4 (DIP switch) is ON, the sensitivity of this pot is greatly increased thus it can be used as an on-board signal source for testing purposes. See section "G".	N/A

TEST POINTS FOR POTENTIOMETERS: See section "G"

SET-UP: See section "G" for engineering and installation notes.

OPERATING MODE SELECTION:

These modes can be selected by the DIP-switches according to the chart in the functional block diagram:

- Current Mode
- Voltage Mode
- IR Compensation Mode*
- Tachometer Mode

APPLICATION NOTES:

For IR compensation mode, a resistor must be added to location R8. See the functional block diagram above and section "G" for more information. The combination of the resistor addition and the switches set for voltage mode will configure the amplifier for IR compensation mode. See section "G" for more information.

See section G for more information on analog position loop mode.

CURRENT LIMIT ADJUSTMENTS:

These amplifiers feature separate peak and continuous current limit adjustments.

The current limit adjusting Pot 2 adjusts both peak and continuous current limit at the same time. It has 12 active turns plus 1 inactive turn at each end and is approximately linear. Thus, to adjust the current limit, turn the potentiometer fully counter-clockwise, then turn clockwise to the appropriate value. If the desired limit is, for example, 10 amperes, and the servo amplifier peak current is 20 amperes, turn the potentiometer 7 turns clockwise from the fully counter-clockwise position.

Pin P1-9 is the input to the internal current amplifier stage. Since the output current is proportional to P1-9, the adjusted current limit can easily be observed at this pin. Note that a command signal must be applied to the reference inputs to obtain a reading on P1-9. The maximum peak current value equals 7.25 V at this pin and the maximum continuous current value equals 3.63 V at this pin.

The actual current can be monitored at pin P1-8.

The continuous current can be reduced without affecting the peak current limit by connecting an external current limiting resistor R-lmt between P1-10 and P1-2. See table below.

Current Limit Resistor (Ω)	15K	6.6K	3.4K	2.1K	1.2K	810	500	250	0
Continuous Current Limit %	90 %	80 %	70 %	60 %	50 %	40%	30 %	20 %	10 %

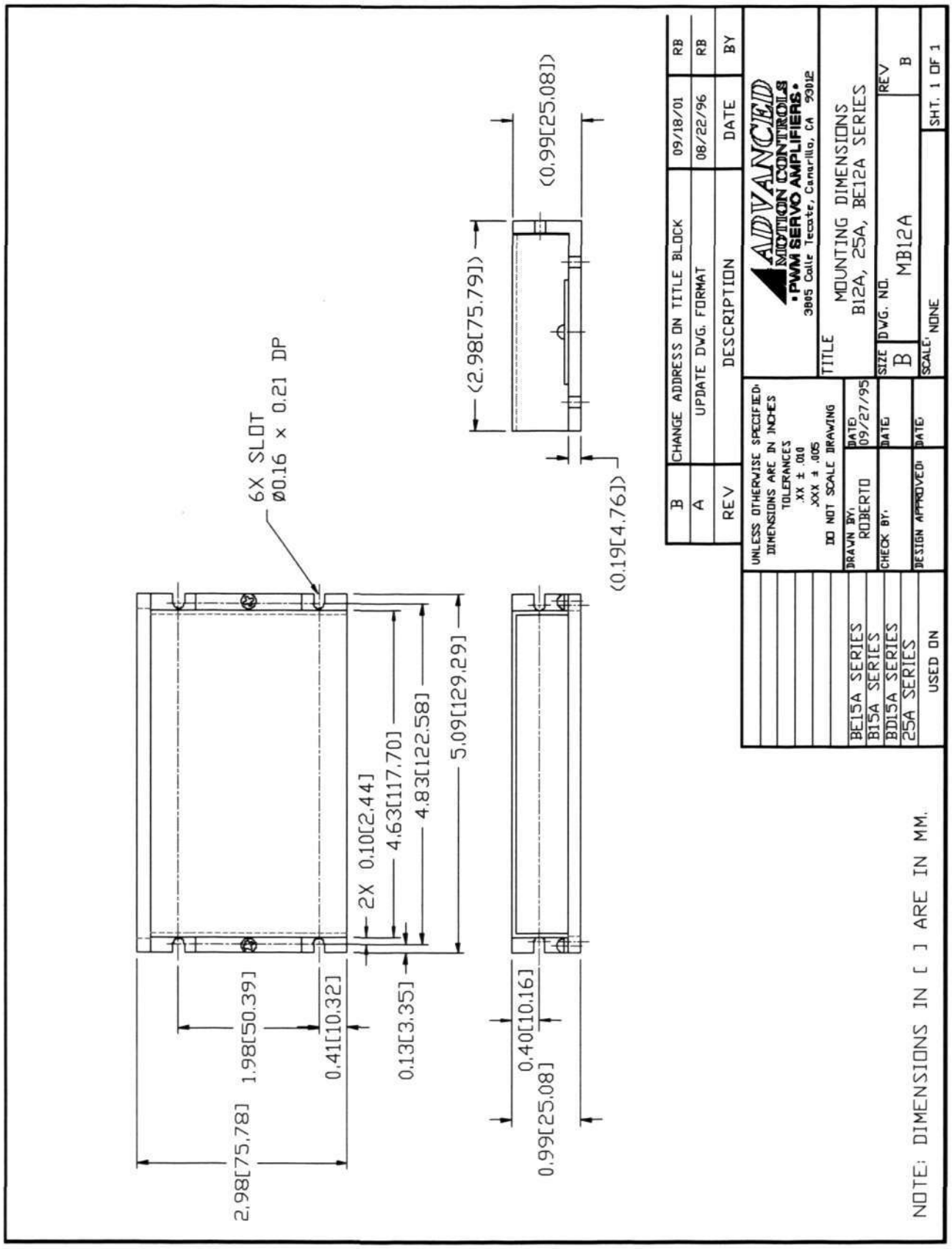
TYPICAL SYSTEM WIRING: See section "G".

ORDERING INFORMATION:

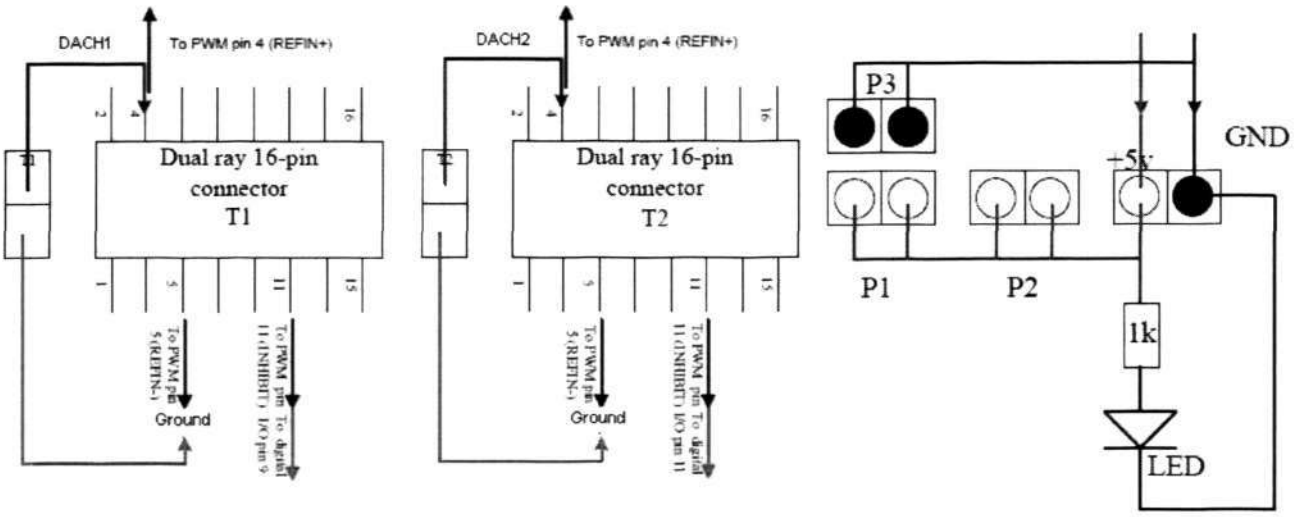
Models: 12A8X, 25A8X, 20A14X, and 20A20X

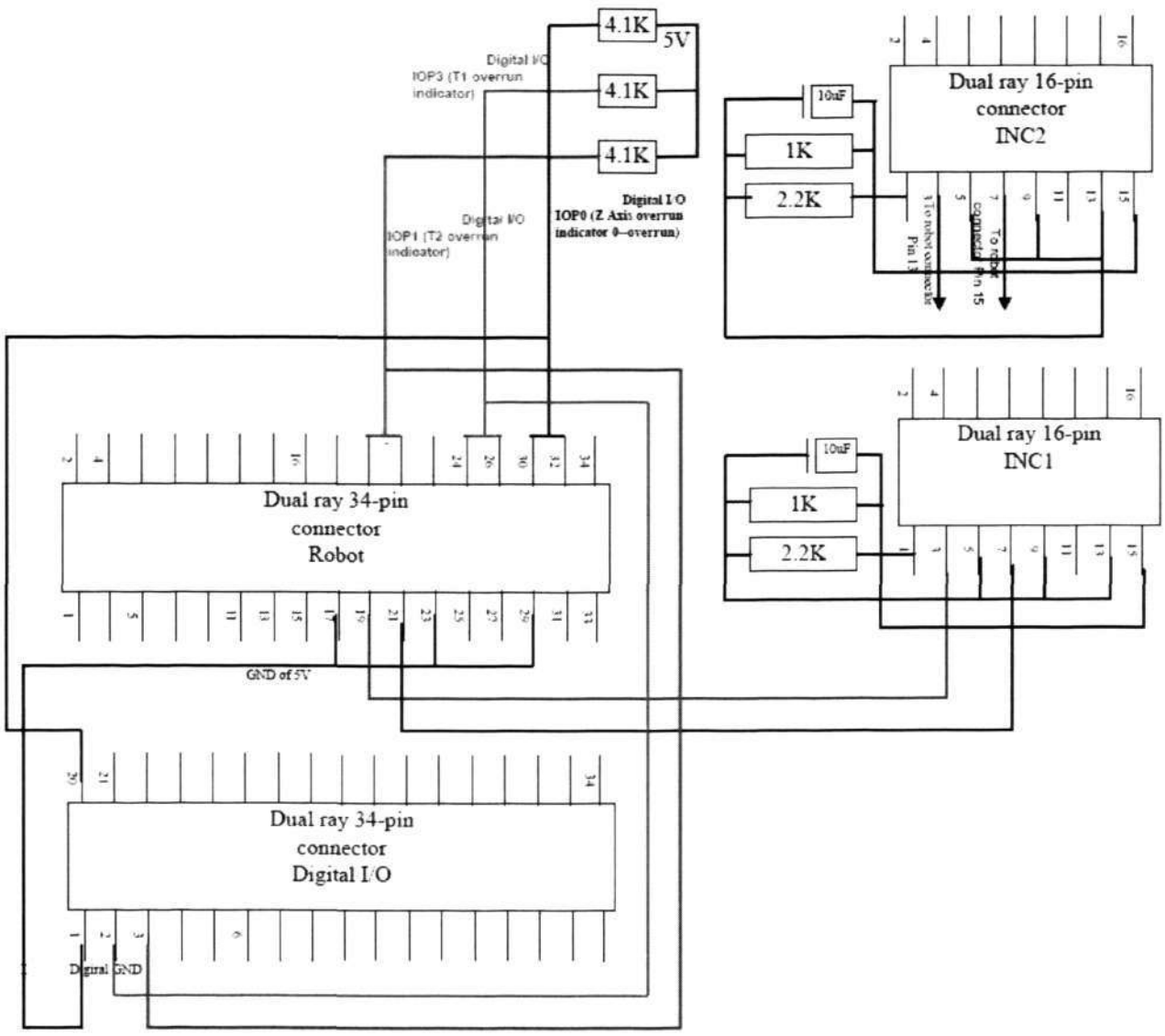
X indicates the current revision letter.

MOUNTING DIMENSIONS: See page F-7.



The detail connection of Robot





External Connections