

# INVESTIGATIONS INTO ITERATIVE LEARNING IN FUZZY CONTROL SYSTEMS

Analysis, Design, Simulations and Experiments

Suhail Ashraf



This thesis is submitted in partial fulfilment of  
the requirement for the degree of  
PhD

Supervised by: Dr. Ejaz Muhamamd

College of Electrical and Mechanical Engineering  
National University of Science and Technology  
Rawalpindi, Pakistan

2008

## ABSTRACT

The most important aspect of human behaviour is learning. One of the learning methodologies applied by humans is learning through iteration. This human capability has recently been used by control engineers to design Iterative Learning Controls (ILC). The problem with ILC is that it is designed for a specific system and a specific desired response. Moreover, the number of iterations is high, especially if the system dynamics are not known. Our research work aims at reducing the number of iterations for convergence and evolving a design mechanism that can adapt for changing systems and varying desired responses, without the need to redesign the ILC. This thesis develops a number of Iterative Learning Controllers to meet these requirements. Stability and convergence criteria of these controllers are also established.

Fuzzy control is another emerging control methodology focusing on human perception and fuzzy thinking. The problem with fuzzy design is the uncertainties associated with the design of membership functions and rule base. Moreover, controlled design requirements are generally given in the form of steady state error, percentage overshoot etc. These requirements need to be translated into fuzzy design. The work also establishes a number of fuzzy controllers combined with ILC to overcome these shortcomings in fuzzy design. The designs are tested through simulations and practical setups. For the practical setups, a Six Degree of Freedom Hexapod, a DC motor kit by Quanser and a custom built Two Degree of Freedom Tracker were used. Stability and convergence of these Iterative Learning Fuzzy Controllers are also discussed.

The research concludes that in order to reduce uncertainties associated with fuzzy logic based design we have to incorporate learning. This hybrid approach can open up a new era of controller design.

## ACKNOWLEDGMENTS

I thank God for giving me the strength and knowledge to complete this research. Thank you (my family) for the patience. And very special thanks to my supervisor, Dr. Ejaz, who gave me endless hours from his very busy schedule. Without his guidance, support and time to time encouragement, this would not have been possible.

Thanks to Dr. Mo. Jamshadi (University of Texas, USA), Dr. Yangquan Chen (Utah State University, USA), Dr. D. H. Owens (University of Sheffield, UK) and Dr. K.L Moore (Utah State University, USA) for knowledgeable e-discussions on concepts of iterative learning control and fuzzy logic especially during the early days of my research. Dr. Robert Parkin (Loughborough University, UK), Dr. Amin Al- Habaibeh (Nottingham Trent University, UK) for their time to time advice.

Thanks to Dr. Tasleem for his support and special thanks to Farooq Rashid (PAEC) for his tips and assistance, especially during the construction of S-101.

## TABLE OF CONTENTS

<b>1. INTRODUCTION</b> .....	1
1.1 Iterative Learning Control.....	2
1.2 Fuzzy Logic / Fuzzy Control .....	4
1.3 Scope and organisation of this thesis.....	8
1.3.1 Scope of research.....	9
1.3.2 Organisation of the thesis.....	9
<b>2. TWO DIMENSIONAL APPROACH TO ITERATIVE LEARNING CONTROL</b> .....	10
2.1 Two Dimensional Learning Process .....	10
2.2 Mathematical Foundation .....	11
2.3 How Humans Learn? .....	15
2.4 How many gains?.....	17
2.5 One Sample At A Time Iterative Learning Controller (OSATILC).....	19
2.5.1 Mathematical background.....	20
2.5.2 Two-Dimensional model of the proposed learning control system.....	21
2.5.3 Stability and convergence analysis using error equations .....	22
2.5.3.1 Case 1: When $K_1 = 0$ .....	24
2.5.3.2 Case 2: When $K_1 \neq 0$ .....	26
2.5.4 Simulation results.....	27
2.6 Multiple Samples at a Time Iterative Learning Controller (MSATILC).....	31
2.7 Modified Multiple Samples At a Time Iterative Learning Controller (MMSATILC).....	33
2.8 Comparative Results .....	36
2.9 Two Degree of Freedom Tracking Platform.....	36
2.9.1 Simulation results for 2DOFTP .....	41
2.10 Experimental Setup.....	43
2.10.1 Six Degree of Freedom Hexapod.....	43
2.10.2 Proposed Approach.....	45
2.10.3 Results.....	48
2.11 Summary.....	50
<b>3. INTELLIGENT CONTROLLERS USING ADAPTIVE ITERATIVE LEARNING</b> .....	51
3.1 Adaptive Learning Controllers .....	52
3.2 When System is Known (Approach-1).....	54
3.2.1 Gradient descent for adaptive gain(s) .....	57
3.2.2 Simulation results.....	60
3.3 When System is Partially Known (Approach-2) .....	65
3.3.1 Gradient descent for adaptive gain(s) .....	67
3.3.2 Simulation results.....	69
3.4 When System is Completely Unknown (Approach-3) .....	73
3.4.1 Identification.....	75
3.4.2 Gradient descent for adaptive gain(s) .....	78
3.4.3 Convergence analysis.....	81

3.4.3.1	Convergence of iterative learning control law .....	81
3.4.3.2	Convergence for adaptive gain .....	82
3.4.4	Simulation results.....	84
3.4.4.1	A Simple System .....	84
3.4.4.2	Car Suspension System.....	88
3.4.4.3	A Non-Linear System .....	91
3.4.5	Discussion and comparison.....	93
3.5	Cost Functions .....	95
3.5.1	Difference of input (Approach-4).....	95
3.5.1.1	Simulation results.....	97
3.6	Iterative Learning Control with an Iterative Learning Gain (Approach-5) .....	100
3.6.1	Convergence analysis.....	101
3.6.2	Simulation results.....	102
3.6.3	Discussions and comparison .....	107
3.6.4	Experimental setup and results .....	108
3.6.5	Real time tracking using iterative learning control with an iterative learning gain.....	110
3.6.5.1	Simulation results.....	111
3.6.5.2	Real time tracking using an experimental set up .....	114
3.7	Iterative Learning Control with an Iterative Learning Gain and Adaptive Step Size .....	117
3.7.1	Adaptive step size .....	119
3.7.2	Simulation results.....	119
3.7.3	Discussion and comparison.....	123
3.8	Summary .....	123
<b>4.</b>	<b>SELF LEARNING FUZZY CONTROLLERS USING ITERATIVE LEARNING TUNER .....</b>	<b>125</b>
4.1	Problems in Fuzzy Logic Based Design .....	125
4.2	Basics of Fuzzy Control.....	129
4.3	Supporting Work.....	131
4.4	Iterative Learning Fuzzy Tuner (ILFT) .....	139
4.4.1	Simulations and results .....	148
4.4.1.1	Conventional Proportional Controller.....	149
4.4.1.2	Iterative Learning Fuzzy Tuner .....	153
4.4.1.2.1	DC Motor .....	153
4.4.1.2.2	A non-linear system .....	158
4.4.2	Stability and convergence.....	162
4.4.3	Tracking a desired trajectory in real time .....	165
4.4.4	Effect of considering derivative of error.....	168
4.4.5	Stability using linguistic trajectory .....	172
4.5	A Real Time Tracker Using ILT.....	175
4.5.1	Simulation results.....	179
4.5.2	Experimental setup.....	185
4.5.2.1	Real time tracking system.....	186
4.5.2.2	The S-101 .....	186
4.5.2.3	Target Simulation Board (TSB).....	191

4.5.2.4	Interface card .....	191
4.5.2.5	Scanner coordinate system.....	192
4.5.2.6	Control software.....	192
4.5.2.7	The complete setup .....	193
4.5.2.8	Experiments using 3 input and 3 output MFs .....	194
4.5.2.9	Experiments using 7 input and 7 output MFs .....	200
4.6	Summary .....	206
<b>5.</b>	<b>ITERATIVE LEARNING FUZZY GAIN SCHEDULER.....</b>	<b>208</b>
5.1	Introduction.....	208
5.2	Proposed Approach.....	210
5.2.1	Procedure for the up gradation of parameters.....	215
5.2.2	Stability .....	217
5.2.3	Simulations and results .....	219
5.2.3.1	Motor Speed Control.....	219
5.2.3.2	Zeigler-Nichols controller vs. proposed approach.....	223
5.2.3.3	Tracking trajectories in real time .....	229
5.3	Experimental Setup and Results .....	233
5.3.1	Stability using Linguistic Trajectory .....	237
5.4	Summary .....	238
<b>6.</b>	<b>FUZZY ITERATIVE LEARNING CONTROLLER (FILC).....</b>	<b>240</b>
6.1	Proposed Approach.....	240
6.1.1	Simulation Results .....	242
6.1.1.1	Case 1: With no initial guess .....	243
6.1.1.2	Case 2: With initial guess .....	248
6.2	Stability .....	250
6.3	Summary .....	251
<b>7.</b>	<b>CONCLUSION AND RECOMMENDATIONS.....</b>	<b>252</b>
7.1	Conclusion .....	253
7.2	Recommendations.....	257
	<b>REFERENCES.....</b>	<b>259</b>
	<b>Appendix A.....</b>	<b>273</b>
A.1	A Simple System (SS).....	273
A.2	Cruise Control System (CCS).....	273
A.3	Car Suspension System (CSS).....	275
A.4	Non-Linear System (NLS).....	276
A.5	Motor Speed Control System (MSCS).....	276
A.6	Inverted Pendulum (INVPL) .....	277
A.7	Desired Trajectory.....	278
	<b>Appendix B.....</b>	<b>279</b>
	<b>Appendix C.....</b>	<b>281</b>
	<b>Appendix D.....</b>	<b>2814</b>



Humans have a remarkable capability to perform a wide variety of physical and mental tasks without any apparent difficulty. Tasks like parking a car, driving in traffic, playing cricket, understanding speech and summarizing a story. All these and other variety of tasks are being performed by the same controller, the brain. Though modern day technology has accomplished great feats like landing on the moon, sophisticated flight control systems, robots that can paint cars, yet we are still unable to assemble and control machines that can talk like humans, build robots that can drive in heavy traffic, prepare programs that can summarize non trivial stories. Also, these controllers need to be redesigned for every new task in hand. Current technologies have still not been able to satisfactorily solve these problems, which seem quite easy for humans.

The key components of human capability lie in learning and forming perceptions. Humans learn form experience, from trials and from repeatedly performing a task. If we want to design controllers with human-like capabilities, we should be designing controllers that are able to iteratively learn from previous attempts. Those controllers should also be able to treat quantities like speed, distance, temperature etc. as perceptions and should be able to function on their imprecise representations rather than their crisp values, just as humans do. For example, actions should be taken on “high speed”, or “low temperature” rather than “greater than 100m/h speed” or “less than 10 degree Celsius” temperature.

With the ever increasing complexity of practical systems and the demand for diverse functionalities, we need new technologies and intelligent systems that can combine knowledge, techniques and methodologies from various sources. These intelligent systems should possess human like capabilities, should be able to adapt themselves and should do better in changing environment. Combining knowledge is the way forward. As in the words of J.S.R. Jang, C.T. Sun and E. Mizutani:-



“It is frequently advantageous to use several computing techniques synergistically rather than exclusively, resulting in construction of complementary hybrid intelligent systems.”  
[75]

This research has focused on combining Iterative Learning and Fuzzy Logic to achieve robust, adaptive and simple-to-design controllers. Before indulging in more technical discussions, a short survey of developments in iterative learning control and fuzzy logic is presented.

## **1.1 Iterative Learning Control**

Iterative Learning Control (ILC) is an approach to improve the performance of a system, operating repetitively, by suitably changing the input to the system. This change in input is done through learning or training. Consider an example of a child throwing stones at a log submerged in a stream. At first, the stone misses the target because the refractive index of the water gives a misleading sense of its location in the water. The child learns to compensate for this effect by throwing the stones slightly off from the required mark. This is not done by changing any fundamental structure of the sensory system, which still observes the log to be at the wrong place. Instead, the child changes the command to the muscles of his arm, telling them to throw at a different mark. The key is to find that change in command or input to the system. Iterative learning control aims at finding such an input command.

The phrase “learning” often causes misunderstanding. This is especially true, given the current interest in artificial neural networks. “Learning” is a broad concept which means different things to different people. In a general sense, learning refers to the action of a system to adapt and change its behaviour based on input/output observations. Many systems have this ability, including adaptive control systems and neural network based systems. In ILC, learning means to learn the next input based on the error signal [83]. This is a fundamental change in control philosophy compared to conventional

thinking where change in input is not a primary goal. Main landmarks of ILC as developed by leading researchers are presented in Appendix B.

For a general introduction on ILC, [83, 84, 120] are good references. Though in the 1980's researchers were primarily concerned with the question of finding convergence conditions, it has come a long way since then. Some of the main areas where ILC researchers have mainly focused are Direct Learning ILC [77, 79, 100, 129, 130], frequency domain analysis and development of ILC techniques [1, 33, 69, 98, 122], multivariable systems [81, 109], non-minimal phase systems [112], feedback systems [10], norm optimal ILC [114, 139], time delay systems [78, 97, 99], time invariant systems [32, 125, 149], time variant systems [125, 140], two dimensional systems [15, 21, 65, 146], repetitive control [13, 124, 142], convergence [24, 85, 97, 150], robustness [111, 138], higher order ILC [30, 149] and neural networks [14, 147, 153].

Iterative Learning Control has found practical applications in almost all the major fields, like robotics [5, 6, 80, 129, 130, 131], automotive vehicles [12, 27, 86], chemical processing [8], mechanical systems control [11, 67, 155], hard drives [61] and even in nuclear reactor [141].

As regards the future scope of work in ILC, researchers have been pointing out in different directions. Some of the directions are mentioned below:-

- (a) We need to explore new ILC paradigm: variable structure iterative learning control. Since there is no coupling between any two consecutive iterations, we can let the ILC mechanism switch its structure from one to another during iterations. The simplest way could be to change the learning gains, and the general one could be to change ILC algorithm during iterations [148].
- (b) Adaptive control has attracted extensive research efforts and has found successful applications. The ILC scheme combined with the adaptive control will be attractive in future [145].
- (c) The actual way we should go is, in fact, in a framework of the 2-D system theory [23].
- (d) The iterative learning controllers plugged into the existing robust controllers should give quite interesting results [64].

- (e) Developing ILC algorithms with lower level of tracking error is theoretically challenging [117].
- (f) If we already have several learned desired inputs for the desired trajectories, how to utilize them is an interesting problem [128].
- (g) Currently, in ILC research, the learning gain is designed based on the ILC convergence condition which may not lead to a good design in terms of knowledge assumed. Therefore, systematic design method is in great desire [145].
- (h) It is well known that output tracking is much more complicated than state tracking and is still an open research area even for linear systems [148].
- (i) From practical point of view, perfect resetting especially the initial state resetting could hardly be achieved. Therefore, it is very important and interesting to investigate the conditions under which resetting requirement can be removed [148].

More recently, N. Amann, D. H. Owens, E. Rogers, M. Norrlof, Mo. Jamshadi, Jian-Xin Xu, Yangquan Chen, L.X. Wang and K.L Moore have been actively pursuing ILC research [15, 28, 34, 41, 44, 52, 53, 60, 77, 78, 79, 83, 84, 85, 103, 105, 106, 113, 115, 116, 127, 144, 145, 151, 152].

The conclusion is simple “Repetition improves skill, for either man or machine.” [145]

## **1.2 Fuzzy Logic / Fuzzy Control**

Fuzzy logic [43,46] was first introduced by Lotfi A. Zadeh [50], Professor of Systems Theory at the University of California, Berkeley, USA, in a publication in 1965 [94]. However, during its early years, it was met with a lot of criticism, some of which were from Prof. Zadeh's colleagues themselves.

Rudolph E. Kalman [55] had this to say in 1972:-

"I would like to comment briefly on Prof. Zadeh's presentation. His proposals could be severely, ferociously, even brutally criticized from a technical point of view. This would be out of place here. But a blunt question remains: Is Prof. Zadeh presenting important ideas or is he indulging in wishful thinking? No doubt Prof. Zadeh's enthusiasm for fuzziness has been reinforced by the prevailing climate in the US - one of unprecedented permissiveness. 'Fuzzification' is a kind of scientific permissiveness; it tends to result in socially appealing slogans unaccompanied by the discipline of hard scientific work and patient observation."

Similarly, his esteemed and brilliant colleague Prof. William Kahan [47,51] whose Evans Hall office is a few doors from Zadeh's, stated the following in 1975:

"Fuzzy theory is wrong, wrong, and pernicious. I cannot think of any problem that could not be solved better by ordinary logic. What Zadeh is saying is the same sort of things: Technology got us into this mess and now it can't get us out. Well, technology did not get us into this mess. Greed and weakness and ambivalence got us into this mess. What we need is more logical thinking, not less. The danger of fuzzy theory is that it will encourage the sort of imprecise thinking that has brought us so much trouble."

Even in the 1990s when hundreds of successful applications of fuzzy logic were being developed, some scientists still condemned the concept, like Jon Konieki who stated in 1991:

"Fuzzy logic is based on fuzzy thinking. It fails to distinguish between the issues specifically addressed by the traditional methods of logic, definition, and statistical decision-making." [47]

Criticisms however did not stop the spread of fuzzy logic. Fuzzy logic, invented in the US, was engineered to perfection in Europe, mass-marketed in Japan, and only then recently returned to US. It can be argued that fuzzy logic has its roots from logic science [101]. A brief historical overview of fuzzy logic is presented in Appendix C.

The full scale operation of fuzzy logic controlled Sendai subway [39] in 1986-87 by Hitachi performed better than any human operator. The subway train, in fact, had a better on-time schedule history, used less energy, and ran smoother than the same train operated by a human. After the success of Sendai subway project hundreds of fuzzy logic based products were produced in Japan, like Mitsubishi's fuzzy logic transmission, Canon's auto focusing mechanism, Minolta's subject tracking system (Maxxum 7xi) and Panasonic Electronic Image Stabilizer.

Theoretically fuzzy controllers should run slower than conventional controllers because of additional computations involved in fuzzifiers, defuzzifiers and inference engines. But the introductions of fuzzy micro chips by companies like Omron, which perform thousands of fuzzy inferences per second, have made this difference irrelevant. In U.S. software companies began offering tools to create fuzzy systems for families of microprocessors. The first was by Togai Infralogic [56] in Irvine, California and the second was by Apronix Inc.[48] in San Jose, California. Fuzzy logic has not been restricted to engineering disciplines, it has found its way into almost all the fields of human interaction, like intelligent project management, project risk assessment, financial statement analyser, forecasting, fleet container management, database retrieval system, abuse detection system [17], to name a few.

Fuzzy control is the most widely used application of fuzzy logic [73]. Fuzzy logic controller (FLC) provides a method to construct controller algorithms in a user friendly way, mimicking human thinking and perception. FLC has successfully outperformed the traditional control systems (like PID controllers) in many areas [18, 19]. But still there are many sources of uncertainties facing the FLC in dynamic real world environment [31]. Uncertainties are independent of the kind of Fuzzy System (FS) or methodology one uses to handle them. Some sources of uncertainties facing the FLC design are as follows:

- (a) Uncertainties in inputs to the FLC, which translate into uncertainties in the antecedent's memberships as the sensor measurements are effected by noises from various sources.
- (b) Uncertainties in control output, which translate into uncertainties in the consequent's membership functions of FLC. Such uncertainties can result from

change of actuator characteristics, which can be due to wear, tear, environment changes etc.

- (c) Linguistic uncertainties as words mean different things to different people [73]. A survey of experts will usually lead to a number of possibilities for the antecedents and consequent of rules. These variations represent uncertainty. J. M. Mendel describes this uncertainty as,  
“Uncertainty about the antecedent and consequent membership functions as experts do not agree on one membership function end-points.” [73]
- (d) Uncertainties associated with the change in operating conditions of the controller. Such uncertainties can also be translated into uncertainties in the membership functions.
- (e) Uncertainties associated with the use of noisy training data that could be used to learn, tune or optimise the FLC.

As regards the problem areas and future work in fuzzy, researchers have been pointing out in different directions. Some of the directions being hinted are annotated below:-

- (a) Fuzzy has progressed a lot during the last decades and the research is going on in the fields like, fuzzy mathematics, fuzzy systems, fuzzy control, image processing, stability analysis, information retrieval, prediction etc. but the two problem areas are the convergence analysis and handling uncertainties [103].
- (b) Possible future work includes adaptation of membership functions so that higher degree of flexibility in search of optima can be achieved [75].
- (c) Conventional fuzzy logic (now called “type-1” fuzzy logic) has limited capabilities to directly handle data uncertainties [71].
- (d) For dynamic unstructured environments and many real world applications, there is a need to cope with large amounts of uncertainties. The traditional fuzzy logic control can not directly handle such uncertainties to produce a better performance [31].
- (e) Unfortunately it (type-1) has completely ignored the uncertainties associated with

the two end points of a membership function [71].

- (f) It is anticipated that by using more general fuzzy logic formulation (now called “type-2” fuzzy logic) [59], it will be possible to capture higher order uncertainties about words. Much remains to be done [71].
- (g) Employment of type-2 fuzzy sets usually increases the computational complexity in comparison with type-1 fuzzy sets due to additional dimension of having to compute secondary grades for each primary membership [22].
- (h) Choosing rules, membership functions are in general still done by hand [119].
- (i) L. A. Zadeh [89] presents a powerful argument for the use of fuzzy logic to manipulate perceptions. His argument is that  
“Perceptions reflect finite ability of sensory organs and the brain to resolve detail and store information. We have partial knowledge, partial understanding, partial certainty, partial belief and accept partial solutions, partial truth and partial causality.” [89]

Perceptions (e.g. perceptions of size, speed, temperature etc.) cannot be modelled by traditional mathematical techniques and that fuzzy logic is more useful in these regards. He also mentions terms like “Computing with words” and “Computing with perceptions” as future research areas.

Current number of researchers mushrooming around the world shows the attractiveness of the fuzzy theory. “Computer world” in its August, 2004 issue reported that there were over 10,000 active fuzzy researchers in China alone [49].

The conclusion is that fuzzy logic has given us a more natural way of looking at the problems and a new way of designing our controllers.

### **1.3 Scope and organisation of this thesis**

We now discuss the scope and organisation of this thesis.

### **1.3.1 Scope of research**

The main aim of the research is to combine Iterative Learning Control and Fuzzy Logic to evolve new methodologies for designing intelligent controllers. These controllers should lower the number of iterations to learn, reduce dependency on knowledge about system dynamics and eliminate uncertainty in the design process. To accomplish this, various ILC techniques and Fuzzy Logic based schemes need to be investigated. Also, their implication in terms of performance parameters like peak overshoot and steady state error should be determined.

### **1.3.2 Organisation of the thesis**

Chapter 2 starts with the development of the mathematical basis for ILC design. Three ILCs are developed and their stability and convergence criteria are established. Chapter 3 takes the research further and presents a number of adaptive ILCs. These ILCs reduced the number of iterations and enhanced adaptability against changes in the system and the desired response. Stability and convergence criteria are also derived. These ILCs were tested using simulations and practical experimental setups. Chapter 4 deals with “Learning Fuzzy” based approach with an emphasis on eliminating uncertainties associated with fuzzy design. Here also, stability and convergence criteria are presented. To test the performance of this controller, a two degree of freedom tracking device was designed and constructed. The robustness of the proposed controller is demonstrated using this tracking device. Chapter 5 combines ILC and Fuzzy Logic to schedule gain values in conventional proportional (P), proportional-integral (PI) and proportional-integral-derivative (PID) controllers. This unique combination of ILC and Fuzzy performed much better than the conventional controllers. Chapter 6 presents a Fuzzy based ILC controller, in which fuzzy logic was used to update the underlying learning law as opposed to the method adopted in chapter 4, where learning was used to update fuzzy controller parameters. Chapter 7 summarises the inferences of this research and also proposes a number of recommendations for future work.



## 2 TWO DIMENSIONAL APPROACH TO ITERATIVE LEARNING CONTROL

This chapter formulates a mathematical framework for developing Iterative Learning Controllers (ILCs). Based on this framework, three controllers are presented. Stability and convergence analysis is also discussed in detail. The controllers are validated through a number of simulations and an experimental setup.

### 2.1 *Two Dimensional Learning Process*

Researchers have used different ways to mathematically represent iterative learning controllers. Some have used continuous time domain [68, 107, 108], some, discrete time domain [95], most have used one dimensional representation [153], while there are some who have used two dimensional representation [146, 112]. Two dimensional (2-D) approach seems a natural way to represent iterative learning processes as we can learn from iteration to iteration and also from sample to sample basis.

A 2-D learning process is one in which inputs, outputs and system states depend on two independent variables i.e. its dynamics are propagated along two independent directions. In fact these are two dynamic processes. One process, indicated by the variable 'k', reflects the dynamics of the system in terms of time history. The other process, indicated by the variable 'j', reflects the learning iteration and resultant performance improvement in terms of learning times [143, 21, 151]. For example,  $u_j(k)$  expresses the  $k^{\text{th}}$  item of the input in the  $j^{\text{th}}$  execution cycle ( $j^{\text{th}}$  learning iteration). Here  $k = 1 \dots N$  and  $j = 1 \dots M$ , where N and M are finite integers. A good introduction of ILC is given in [83].

## 2.2 Mathematical Foundation

Following variables are defined:-

$u_j(k)$  = Input at current iteration or trial

$u_{j+1}(k)$  = Input at next iteration or trial

$\Delta u_j(k)$  = Change in input calculated at current iteration

$y_j(k)$  = Output at current iteration or trial

$y_j(k+1)$  = Next output at current iteration or trial

$e_j(k)$  = Error at current iteration

$y_d_j(k)$  = Desired output at current iteration

Appendix B presents a brief history of ILC with a list of definitions as given by different researchers. It can be concluded from these definition that, the main aim of ILC is to change next input so as to reduce error. This means that after several learning cycles,  $u_j(k)$  is modified to approach a desired control input which can generate the desired output. Mathematically the modification can be expressed as

$$u_{j+1}(k) = u_j(k) + \Delta u_j(k) \quad (2.1)$$

The equation states that the next input to the system for sample  $k$  is equal to the current input plus  $\Delta u_j(k)$ , where  $\Delta u_j(k)$  is in general a function of the error between the system's actual output  $y_j(k)$  and the desired output  $y_d_j(k)$ . Here  $j+1$  means that the new control input is for the next execution or learning cycle.

The error is given by

$$e_j(k) = y_d_j(k) - y_j(k) \quad (2.2)$$

A general structure adapted for the proposed iterative learning controls, similar to the one given in [83], is shown in figure 2.1.

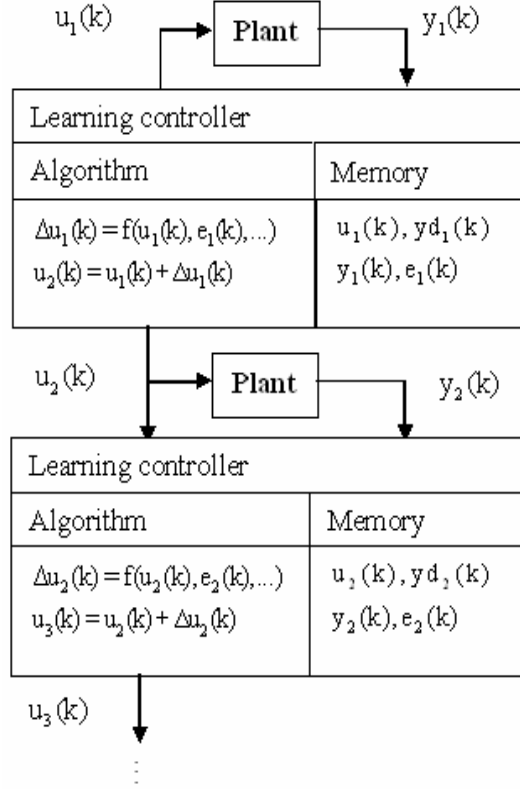


Figure 2.1: A general structure of an iterative learning control system.

Input at iteration 1,  $u_1(k)$ , for  $k = 1 \dots N$ , is applied to the plant. This input produces an output  $y_1(k)$ . According to the equation (2.1),  $u_1(k)$  needs to be modified. This modification,  $\Delta u_1(k)$ , can be a function of  $u_1(k)$ ,  $e_1(k)$ ,  $y_1(k)$ ,  $yd_1(k)$  etc. As explained before, the purpose of learning process is to generate a new control input,  $u_2(k)$ , in this case, that can reduce or eliminate the error  $e_2(k)$ . This change in input,  $\Delta u_1(k)$ , and error at first iteration,  $e_1(k)$ , must be related. Values of  $u_1(k)$ ,  $e_1(k)$ ,  $y_1(k)$  and  $yd_1(k)$  are stored in memory, to be used in future iterations. The new input,  $u_2(k)$  is applied to the plant again, resulting in  $y_2(k)$ . The process is repeated again and again until the error comes down to acceptable limit.

To modify  $u_j(k)$  into  $u_{j+1}(k)$ , error information  $e_j(k)$ ,  $e_j(k+1)$ ,  $e_j(k-1)$  etc. are used. The following equation for change in input is suggested.

$$\Delta u_j(k) = K_0 e_j(k) + K_1 e_j(k+1) \quad (2.3)$$

Where  $K_0$  and  $K_1$  are gain values. There can be a separate value of  $K_0$  and  $K_1$  for each sample or one gain value for all the samples. If a separate gain value is required for each sample,  $K_0$  and  $K_1$  are of the form

$$\mathbf{K}_0 = \mathbf{K}_0(k) = [K_0(1) \quad K_0(2) \quad \dots \quad K_0(N)] \text{ and}$$

$$\mathbf{K}_1 = \mathbf{K}_1(k) = [K_1(1) \quad K_1(2) \quad \dots \quad K_1(N)]$$

Where  $K_0(1), K_0(2), \dots, K_0(N)$  and  $K_1(1), K_1(2), \dots, K_1(N)$  are the gains associated with samples 1, 2, ..., N. For example, for sample 3, equation (2.3) becomes

$$\Delta u_j(3) = K_0(3)e_j(3) + K_1(3)e_j(4)$$

Here  $K_0(3)$  is the gain assigned to sample number 3. This structure has a problem associated with its last sample. For example, for sample N, the value of  $K_1 e_j(N+1)$  can not be resolved, as  $e_j(N+1)$  is not available. Different methods, like using rate of change of error, average of error were used. It was found that the best method, considering execution time and final results, is to use the previous error value for  $e_j(N+1)$ . That is,  $e_j(N+1) = e_j(N)$ . Through out this thesis,  $e_j(N+1)$  error value is taken as,  $e_j(N)$  (error value at sample N).

The general block diagram of the proposed schemes is given in figure 2.2.

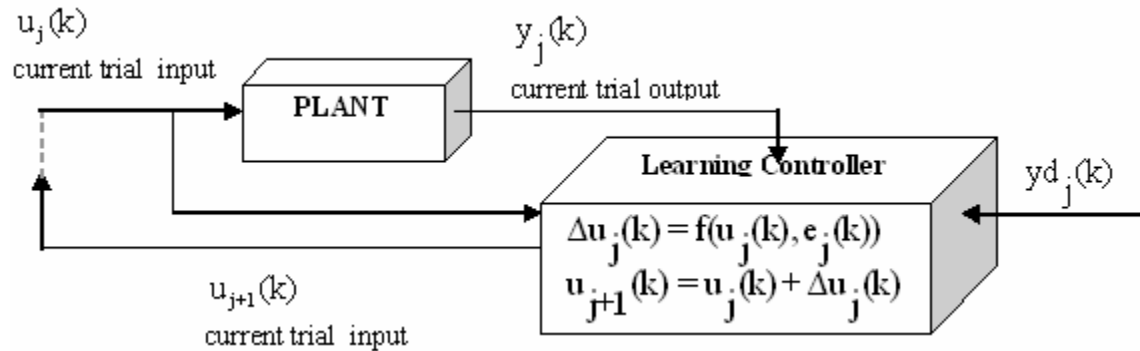


Figure 2.2: Learning control configuration of the proposed schemes using 2-D theory.

The Learning controller calculates  $\Delta u_j(k)$  based on information on error and current input.

Putting (2.3) in (2.1), the learning process proceeds as

$$u_{j+1}(k) = u_j(k) + K_0 e_j(k) + K_1 e_j(k+1) \quad (2.4)$$

Many learning control algorithms proposed by different researchers are of the form similar to (2.1). Several interesting facts concerning these algorithms were observed. For example:-

- (a) Most current methods adopt fixed learning laws. Only control input sequence is modified. Although this type of learning laws can be easily implemented they are unable to cope with changing requirements, uncertainties in model of the system and change in model parameters. This type of control is most useful where the tasks always have the same desired trajectory.
- (b) In order to find the learning gains,  $K_0$  and  $K_1$ , some specific knowledge of controlled system is needed. In case of model based learning, an explicit expression of system inverse dynamic model should be available.
- (c) The learning performance depends on the accuracy of the inverse model used in the design.

The development of the ILCs has focused at following aims:-

- (a) They should require minimum knowledge about the plant.
- (b) They should converge even if there is uncertainty in plant model.
- (c) They should converge even if the resetting is not perfect.
- (d) They should be able to handle desired trajectory changes. The knowledge learnt from one task should be utilized for other tasks as well.
- (e) They should be able to adjust for small changes in the controlled system due to wear, tear and aging.
- (f) They should be usable for non-linear systems.
- (g) The number of iterations should be reduced to an acceptable level.

In order to develop such an approach, the behaviour of  $K_0$  and  $K_1$  were first studied. Throughout this thesis, simulation results from different linear and non-linear systems are presented. The systems are named Simple System (SS), Cruise Control System (CCS), Car Suspension System (CSS), Non-linear System (NLS), Motor Speed Control System (MSCS) and Inverted Pendulum System (INVPL). The SS, CCS, CSS and MSCS will also be referred by  $G_1(z)$ ,  $G_2(z)$ ,  $G_3(z)$  and  $G_4(z)$  in this thesis. These systems are described in Appendix A.

### 2.3 How Humans Learn?

The iterative learning control tries to mimic the most important aspect of human behaviour, i.e. learning from experience. To get a quantitative measure of how a human will learn an input, that will make a motor run at a predefined speed, interactive software was developed. The software gave the human operator control over the input. This input, the corresponding output and the error were continuously displayed to help the human operator select the next input. The software also varied the desired speed at run time, to record human operator's response to change in requirements. The responses were then analysed. A snap shot of one such test performed by one human is given in figure 2.3.

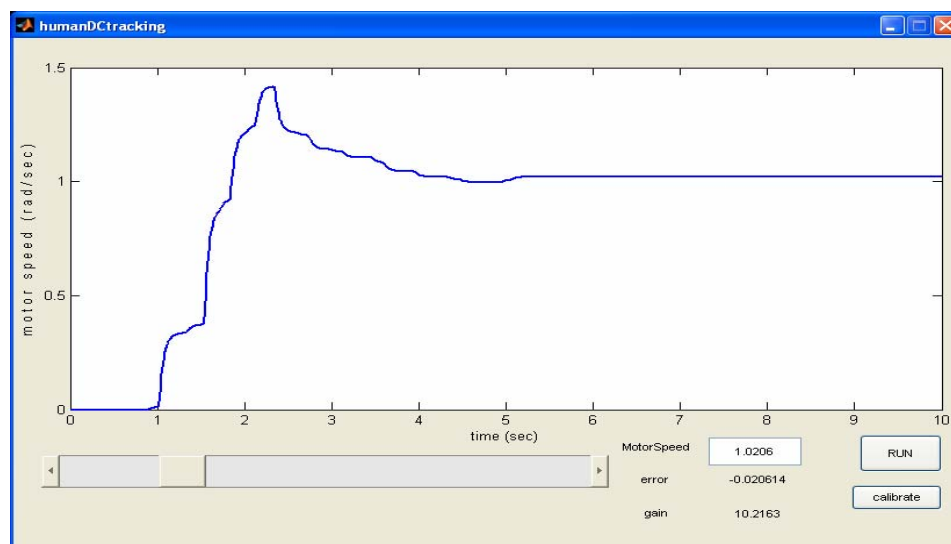


Figure 2.3: Graphical Interface for human operators to control input for a motor.

The plot shows motor speed as input is applied by the human operator using the slide bar. The required speed is 1 radians/second. Gain was also recorded for better data analysis.

Humans from varying age groups took the test. A plot of input applied by three such humans, at their third iteration, to achieve a speed of 1 radians/second is shown in figure 2.4.

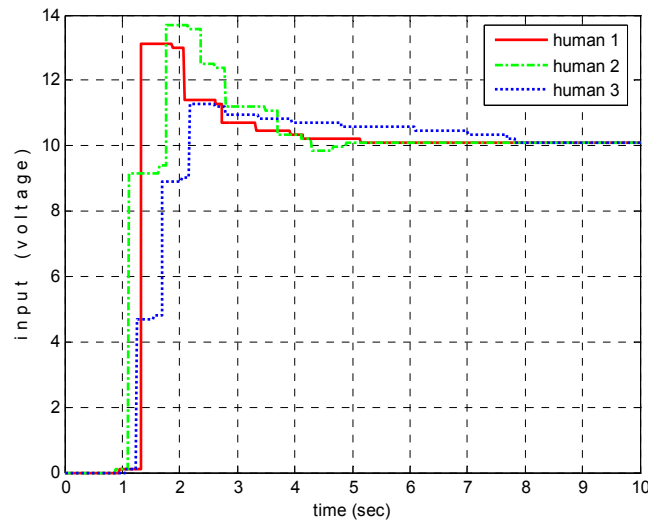


Figure 2.4: Different inputs applied by human operators to attain a speed of 1 radian/second.

The similarity of human response is quite evident. A few interesting points noted were:-

- (a) Human operator was, almost, never able to achieve desired response at first go. They needed repetition.
- (b) Initially, human operator tends to give higher gains, partially because of higher error at the start and partially to get a feeling of the system response.
- (c) The experience of the previous inputs helps in determining the next input.
- (d) The next input applied is mainly a function of error.
- (e) Human response is usually under damped.
- (f) Human operator usually waits to see the output of the system before guessing the next required input.

- (g) When following a predefined shape, human operator tends to break the desired shape in smaller pieces or intervals. The more the human operator concentrates, to reduce error, the smaller the interval. The concept is that if error is reduced in those small intervals, the overall error is also reduced.

The result of these simulations contributed in the formulation of learning control laws, presented in this chapter.

## **2.4 How many gains?**

The control law of equation (2.4) can be extended to

$$u_{j+1}(k) = u_j(k) + K_0 e_j(k) + K_1 e_j(k+1) + \dots + K_{N-1} e_j(k+N-1)$$

This law requires N gains. As discussed in section 2.2, each gain can be a separate value for each sample or one gain value for all the samples. All these gains are not only heavy in terms of computation but also complicate the stability of the control law. There was a need to investigate, “How many minimum numbers of gains are sufficient to achieve convergence?” Simulation results from different kinds of systems, using control law of equation (2.4), revealed following interesting facts:-

- (a) The possible minimum numbers of iterations were different for different systems. These possible minimum numbers of iterations are called optimal number of iterations in this thesis.
- (b) There was always a range of values of  $K_0$  and  $K_1$ , for each system, that produced optimal number of iterations; meaning that, different combinations of  $K_0$  and  $K_1$  can give same results.
- (c) The term  $K_1 e_j(k+1)$  in equation (2.4), some times introduced oscillations and even



instability.

(d) Considering  $K_0$  alone can also achieve optimal number of iterations.

These observations are demonstrated in results from one such simulation on  $G_1(z)$ . Figure 2.5 gives a plot of number of iterations vs. values of  $K_0$  and  $K_1$ .

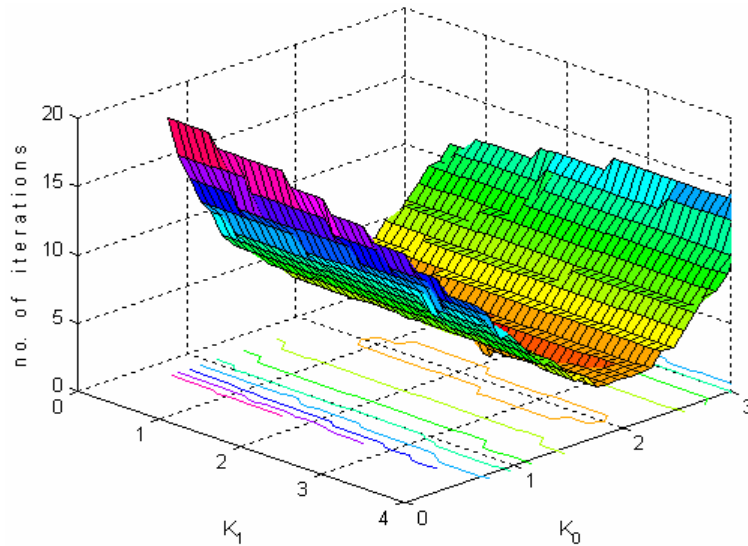


Figure 2.5: Surface and contour plot of  $K_0$  and  $K_1$  vs. number of iterations for  $G_1(z)$ .

As can be seen there is a range of values of  $K_0$  and  $K_1$  that give optimal iterations. The minimum achievable iterations for this system were 4. For the same system, figure 2.6 gives a plot of number of iterations as  $K_0$  is varied, keeping  $K_1 = 0$ .

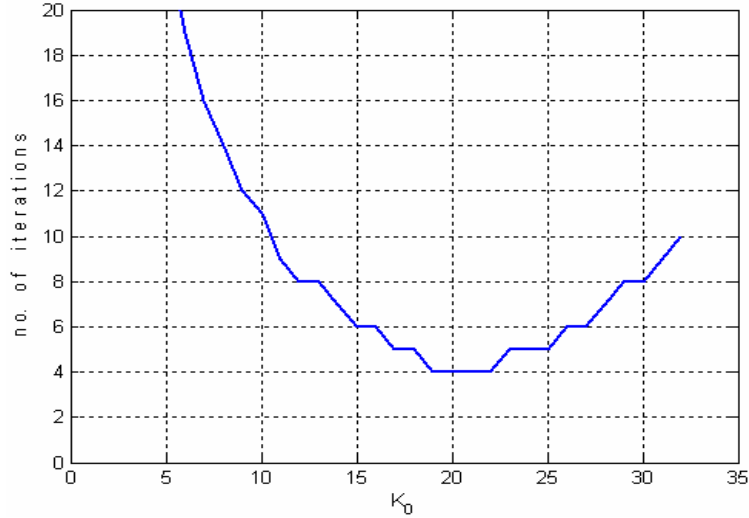


Figure 2.6: Number of iterations against  $K_0$  for  $G_1(z)$

Again the minimum number of iterations achieved was 4, though with higher values of  $K_0$ . Similar simulations on different systems concluded that:

- (a) Tune the control algorithm initially, by using  $K_0$  alone.
- (b) Consider  $K_1$  and higher gains only, if the requirements are not met.

This procedure of selection of gains is followed through out this thesis.

## **2.5 One Sample At A Time Iterative Learning Controller (OSATILC)**

A survey of ILC research already published revealed, that the corrective input was updated for all the samples, for all iterations, until the desired response was achieved. This means that even if most part of the desired response was achieved, the whole input was changed. In other words all the samples were modified at all the iterations. One

Sample At a Time (OSAT) approach focuses on converging one sample at a time. In this approach each sample is given a priority. The order of priority is from sample 1 to N. This means that if samples 1 to k have converged i.e. their output match the desired output, no correction in input is made for these samples. The rules for OSATILC are stated below:-

- (1) Samples should be given a priority from sample 1 to N, where sample 1 has the highest priority and sample N has the least priority.
- (2) There should be no gap between the samples, i.e. samples 1 to 10 should converge before sample 11 is considered for convergence.
- (3) The input changes for only that sample which is under consideration.
- (4) Once a sample input has converged i.e. the input value of  $k^{\text{th}}$  sample is calculated, the value of that input should be applied as a starting value for  $k+1^{\text{th}}$  sample. i.e.

$$u_j(k+1) = u_j(k) \quad (2.5)$$

- (5) Once the input for a sample has been calculated, no change in that input will be made in any subsequent iterations i.e.

$$u_{j+1}(1...k_c) = u_j(1...k_c) \quad (2.6)$$

Where,  $k_c$  represents the number of converged samples.

We now develop the mathematical frame work for OSATILC and other ILCs presented in this chapter.

### 2.5.1 Mathematical background

This section develops the mathematical basis required to formulate Iterative Learning Controllers. As our ILCs are discrete controllers, we take the discrete time representation of a system, given by equation:-

$$y(k+1) = Ay(k) + Bu(k) \quad (2.7)$$

Where current output, current input and next output are represented by  $y(k)$ ,  $u(k)$  and  $y(k+1)$ . The system coefficients are represented by A and B.

In 2-D format this representation can be written as

$$y_j(k+1) = Ay_j(k) + Bu_j(k) \quad (2.8)$$

Where  $y_j(k+1)$  is the next output at  $j^{\text{th}}$  iteration.

In discrete terms, the effect of input sample  $k$  is observed at sample  $k+n$ , at the output. Where  $n$  is the delay of the system. Therefore, the number of samples for  $y$  is from  $1 \dots (N+n)$ . Putting it another way, this means that the input at sample  $(k+5)$  will have no effect at output sample  $(k+5)$  and lower. Using the delay of the system the error equation (2.2) is written, in a more generalized form as

$$e_j(k) = yd_j(k) - y_j(k+n) \quad (2.9)$$

Which means that the error at sample  $k$  is the difference between the desired output at sample  $k$ , minus the actual output delayed by  $n$  samples.

## 2.5.2 Two-Dimensional model of the proposed learning control system

A 2-D state space representation of the ILC scheme will be developed in this section.

Error equation (2.9), for next sample, can be extended to

$$e_j(k+1) = yd_j(k+1) - y_j(k+1+n) \quad (2.10)$$

Putting the values of  $e_j(k)$  and  $e_j(k+1)$  from (2.9) and (2.10) in (2.4)  $\Rightarrow$

$$\begin{aligned} u_{j+1}(k) &= u_j(k) + K_0(yd_j(k) - y_j(k+n)) + K_1(yd_j(k+1) - y_j(k+1+n)) \\ u_{j+1}(k) &= u_j(k) + K_0 yd_j(k) - K_0 y_j(k+n) + K_1 yd_j(k+1) - K_1 y_j(k+1+n) \end{aligned} \quad (2.11)$$

Discrete representation of a 2-D system from (2.8) can be extended, for a system with  $n$  sample delay, to

$$y_j(k+1+n) = Ay_j(k+n) + Bu_j(k) \quad (2.12)$$

Using (2.12) in (2.11) gives

$$\begin{aligned} u_{j+1}(k) &= u_j(k) + K_0 y_d_j(k) - K_0 y_j(k+n) + K_1 y_d_j(k+1) - K_1 (Ay_j(k+n) + Bu_j(k)) \\ u_{j+1}(k) &= u_j(k) + K_0 y_d_j(k) - K_0 y_j(k+n) + K_1 y_d_j(k+1) - K_1 Ay_j(k+n) - K_1 Bu_j(k) \\ u_{j+1}(k) &= -K_0 y_j(k+n) - K_1 Ay_j(k+n) + u_j(k) - K_1 Bu_j(k) + K_0 y_d_j(k) + K_1 y_d_j(k+1) \end{aligned} \quad (2.13)$$

$$u_{j+1}(k) = (-K_0 - K_1 A)y_j(k+n) + (I - K_1 B)u_j(k) + K_0 y_d_j(k) + K_1 y_d_j(k+1) \quad (2.14)$$

Equation (2.12) and (2.14) in compact matrix form give

$$\begin{bmatrix} y_j(k+1+n) \\ u_{j+1}(k) \end{bmatrix} = \begin{bmatrix} A & B \\ -K_0 - K_1 A & I - K_1 B \end{bmatrix} \begin{bmatrix} y_j(k+n) \\ u_j(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ K_0 & K_1 \end{bmatrix} \begin{bmatrix} y_d_j(k) \\ y_d_j(k+1) \end{bmatrix} \quad (2.15)$$

This is the 2-D state space representation of the proposed ILC.

### 2.5.3 Stability and convergence analysis using error equations

Stability and convergence issues were studied using the Roesser model given by R. P. Roesser [123]. Roesser presents a two-dimensional discrete state-space model in which the state of the system is divided into “horizontal” and a “vertical” state. The model is given by

$$\begin{bmatrix} x^h(i+1, j) \\ x^v(i, j+1) \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} x^h(i, j) \\ x^v(i, j) \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u(i, j) \quad (2.16)$$

where  $i$  and  $j$  are non negative integers, named horizontal and vertical coordinates and  $x^h \in \mathbb{R}^{n1}$ ,  $x^v \in \mathbb{R}^{n2}$  are local states of the system which are propagated horizontally and vertically by the first order difference equations respectively. Input vector is represented by  $u \in \mathbb{R}^f$  and output vector by  $y \in \mathbb{R}^m$ . Matrices  $A_1, A_2, A_3, A_4, B_1, B_2, C_1$  and  $C_2$  have appropriate dimensions. It is well known that Roesser’s model is the most general 2-

D state space model. For a special case of Roesser model where  $A_2 = 0$  and  $A_3 = 0$ , the necessary conditions for a system, of form (2.16), to be stable are

$$\rho(A_1) < 1, \rho(A_4) < 1 \quad (2.17)$$

Here  $\rho$  is the 1-D spectral radii [133]. The spectral radius of a matrix or a bounded linear operator is the supremum among the moduli of the elements in its spectrum [40]. That is, if  $A_1$  is some complex or real element with eigenvalues  $\lambda_1, \dots, \lambda_n$ . Then the spectral radius  $\rho(A_1)$  of  $A_1$  is

$$\rho(A_1) = \max_{1 \leq i \leq n} |\lambda_i|$$

Equation (2.17) means that for the model in (2.16) with  $A_2$  and  $A_3$  set to zero, if spectral radii of  $A_1$  and  $A_4$  is less than 1, the 2-D system is stable. Hence if any 2-D approach can be represented in Roesser model format, the stability and convergence criteria can be applied.

By letting iteration  $j$  increase to  $j+1$ , equation (2.10) and (2.12)  $\Rightarrow$

$$e_{j+1}(k+1) = yd_{j+1}(k+1) - y_{j+1}(k+1+n) \quad (2.18)$$

$$y_{j+1}(k+1+n) = Ay_{j+1}(k+n) + Bu_{j+1}(k) \quad (2.19)$$

Putting (2.19) in (2.18)  $\Rightarrow$

$$e_{j+1}(k+1) = yd_{j+1}(k+1) - (Ay_{j+1}(k+n) + Bu_{j+1}(k)) \quad (2.20)$$

$$e_{j+1}(k+1) = yd_{j+1}(k+1) - Ay_{j+1}(k+n) - Bu_{j+1}(k)$$

Adding  $Ayd_{j+1}(k)$  on both sides

$$Ayd_{j+1}(k) + e_{j+1}(k+1) = Ayd_{j+1}(k) + yd_{j+1}(k+1) - Ay_{j+1}(k+n) - Bu_{j+1}(k)$$

$$e_{j+1}(k+1) = yd_{j+1}(k+1) - Ayd_{j+1}(k) + Ayd_{j+1}(k) - Ay_{j+1}(k+n) - Bu_{j+1}(k)$$

$$e_{j+1}(k+1) = yd_{j+1}(k+1) - Ayd_{j+1}(k) + A(yd_{j+1}(k) - y_{j+1}(k+n)) - Bu_{j+1}(k)$$

$$e_{j+1}(k+1) = yd_{j+1}(k+1) - Ayd_{j+1}(k) + Ae_{j+1}(k) - Bu_{j+1}(k) \quad (2.21)$$

Putting (2.12) in (2.10)  $\Rightarrow$

$$e_j(k+1) = yd_j(k+1) - (Ay_j(k+n) + Bu_j(k)) \quad (2.22)$$

$$e_j(k+1) = yd_j(k+1) - Ay_j(k+n) - Bu_j(k) \quad (2.23)$$

Adding  $Ayd_j(k)$  on both sides of equation (2.23)  $\Rightarrow$

$$Ayd_j(k) + e_j(k+1) = Ayd_j(k) + yd_j(k+1) - Ay_j(k+n) - Bu_j(k)$$

$$e_j(k+1) = yd_j(k+1) - Ayd_j(k) + Ayd_j(k) - Ay_j(k+n) - Bu_j(k)$$

$$e_j(k+1) = yd_j(k+1) - Ayd_j(k) + A(yd_j(k) - y_j(k+n)) - Bu_j(k)$$

$$e_j(k+1) = yd_j(k+1) - Ayd_j(k) + Ae_j(k) - Bu_j(k) \quad (2.24)$$

Subtracting (2.24) from (2.21)  $\Rightarrow$

$$e_{j+1}(k+1) - e_j(k+1) = yd_{j+1}(k+1) - yd_j(k+1) - Ayd_{j+1}(k) + Ayd_j(k) + \dots \quad (2.25)$$

$$Ae_{j+1}(k) - Ae_j(k) - Bu_{j+1}(k) + Bu_j(k)$$

Assuming that the desired output does not change, i.e.

$$yd_{j+1}(k+1) = yd_j(k+1) \text{ and } yd_{j+1}(k) = yd_j(k)$$

Equation (2.25) becomes

$$e_{j+1}(k+1) - e_j(k+1) = Ae_{j+1}(k) - Ae_j(k) - Bu_{j+1}(k) + Bu_j(k) \quad (2.26)$$

We now discuss two cases one when  $K_1 = 0$  and the other when  $K_1 \neq 0$ .

### 2.5.3.1 Case 1: When $K_1 = 0$

Defining

$$\tilde{\lambda}_j(k) = e_{j+1}(k) - e_j(k) \quad (2.27)$$

and

$$\tilde{\lambda}_j(k+1) = e_{j+1}(k+1) - e_j(k+1) \quad (2.28)$$

Putting (2.28) in (2.26)  $\Rightarrow$

$$\tilde{\lambda}_j(k+1) = Ae_{j+1}(k) - Ae_j(k) - Bu_{j+1}(k) + Bu_j(k) \quad (2.29)$$

As  $K_1 = 0$ , equation (2.4) can be rewritten as

$$u_{j+1}(k) = u_j(k) + K_0 e_j(k) \quad (2.30)$$

Using (2.30) in (2.29)  $\Rightarrow$

$$\begin{aligned} \hat{\lambda}_j(k+1) &= A e_{j+1}(k) - A e_j(k) - B(u_j(k) + K_0 e_j(k)) + B u_j(k) \\ \hat{\lambda}_j(k+1) &= A e_{j+1}(k) - A e_j(k) - B u_j(k) - K_0 B e_j(k) + B u_j(k) \end{aligned} \quad (2.31)$$

$$\hat{\lambda}_j(k+1) = A e_{j+1}(k) - A e_j(k) - K_0 B e_j(k)$$

$$\hat{\lambda}_j(k+1) = A(e_{j+1}(k) - e_j(k)) - K_0 B e_j(k)$$

Using definition in (2.27)  $\Rightarrow$

$$\hat{\lambda}_j(k+1) = A \hat{\lambda}_j(k) - K_0 B e_j(k) \quad (2.32)$$

Rearranging (2.27)  $\Rightarrow$

$$e_{j+1}(k) = \hat{\lambda}_j(k) + e_j(k) \quad (2.33)$$

Writing (2.32) and (2.33) in Roesser model form

$$\begin{bmatrix} \hat{\lambda}_j(k+1) \\ e_{j+1}(k) \end{bmatrix} = \begin{bmatrix} A & -K_0 B \\ I & I \end{bmatrix} \begin{bmatrix} \hat{\lambda}_j(k) \\ e_j(k) \end{bmatrix} \quad (2.34)$$

Using the Roesser model convergence criteria in (2.17), following convergence theorem is presented.

**Theorem 1:**

The Roesser model in (2.16), its convergence criteria in (2.17) and the Roesser model of the proposed scheme in (2.34)  $\Rightarrow$

$$\rho(A_1) < 1 \Rightarrow \rho(A) < 1 \quad (2.35)$$

Inequality (2.35) requires that the original system should be stable i.e. its poles should be with in the unit circle.

And



$$A_2 = 0 \Rightarrow K_0 B = 0 \quad (2.36)$$

Equation (2.36) states that if the value of  $K_0$  is chosen such that,  $K_0 B = 0$ , the resulting 2-D error system is stable and the learning process will converge.

### 2.5.3.2 Case 2: When $K_1 \neq 0$

Defining

$$\hat{\lambda}_j(k) = e_{j+1}(k) - e_j(k) + K_1 B e_j(k) \quad (2.37)$$

and

$$\hat{\lambda}_j(k+1) = e_{j+1}(k+1) - e_j(k+1) + K_1 B e_j(k+1) \quad (2.38)$$

Putting (2.4) in (2.26)  $\Rightarrow$

$$e_{j+1}(k+1) - e_j(k+1) = A e_{j+1}(k) - A e_j(k) - B(u_j(k) + K_0 e_j(k) + K_1 e_j(k+1)) + B u_j(k)$$

$$e_{j+1}(k+1) - e_j(k+1) = A e_{j+1}(k) - A e_j(k) - B u_j(k) - K_0 B e_j(k) - K_1 B e_j(k+1) + B u_j(k)$$

$$e_{j+1}(k+1) - e_j(k+1) + K_1 B e_j(k+1) = A e_{j+1}(k) - A e_j(k) - B u_j(k) - K_0 B e_j(k) + B u_j(k)$$

$$e_{j+1}(k+1) - e_j(k+1) + K_1 B e_j(k+1) = A e_{j+1}(k) - A e_j(k) - K_0 B e_j(k)$$

Adding  $K_1 A B e_j(k)$  on both sides and rearranging

$$e_{j+1}(k+1) - e_j(k+1) + K_1 B e_j(k+1) = A e_{j+1}(k) - A e_j(k) + K_1 A B e_j(k) - K_1 A B e_j(k) - K_0 B e_j(k)$$

$$e_{j+1}(k+1) - e_j(k+1) + K_1 B e_j(k+1) = A(e_{j+1}(k) - e_j(k) + K_1 B e_j(k)) - K_1 A B e_j(k) - K_0 B e_j(k)$$

$$e_{j+1}(k+1) - e_j(k+1) + K_1 B e_j(k+1) = A(e_{j+1}(k) - e_j(k) + K_1 B e_j(k)) - (K_1 A B + K_0 B) e_j(k) \quad (2.39)$$

Using (2.37) and (2.38) in (2.39)

$$\hat{\lambda}_j(k+1) = A \hat{\lambda}_j(k) - (K_1 A B + K_0 B) e_j(k) \quad (2.40)$$

Rearranging (2.37)  $\Rightarrow$

$$e_{j+1}(k) = \hat{\lambda}_j(k) + e_j(k) - K_1 B e_j(k)$$

$$e_{j+1}(k) = \hat{\lambda}_j(k) + (I - K_1 B) e_j(k) \quad (2.41)$$

Writing (2.40) and (2.41) in Roesser model form

$$\begin{bmatrix} \tilde{\lambda}_j(k+1) \\ e_{j+1}(k) \end{bmatrix} = \begin{bmatrix} A & -(K_1AB + K_0B) \\ I & I - K_1B \end{bmatrix} \begin{bmatrix} \tilde{\lambda}_j(k) \\ e_j(k) \end{bmatrix} \quad (2.42)$$

Using the Roesser model convergence criteria in (2.17), following convergence theorem is presented

**Theorem 2:**

The Roesser model in (2.16) the convergence criteria in (2.17) and the Roesser model of the proposed scheme  $\Rightarrow$

$$\rho(A_1) < 1 \Rightarrow \rho(A) < 1 \quad (2.43)$$

This means that the original system should be stable i.e. its poles should be within the unit circle. Criteria

$$A_2 = 0 \Rightarrow (K_1AB + K_0B) = 0 \quad (2.44)$$

And criteria

$$\rho(A_4) < 1 \Rightarrow I - K_1B < 1 \quad (2.45)$$

Inequality (2.43) and equations (2.44) and (2.45) state that:

If the system is stable and the values of  $K_0$  and  $K_1$  are chosen such that  $(K_1AB + K_0B) = 0$  and  $I - K_1B < 1$ , then the resulting 2-D error system is stable and the learning process will converge.

To test the OSATILC, a numbers of simulations were carried out. Some of the results are presented in the following section.

**2.5.4 Simulation results**

This section presents simulation results from  $G_1(z)$ ,  $G_2(z)$ ,  $G_3(z)$  and NLS systems. The following results were taken with values of  $K_0 = 0.1$ ,  $K_1 = 0$  and sampling time  $(T_s) = 0.1$ . The system convergence criteria was

$$\|e_j(k)\| < 0.01 \tag{2.46}$$

Figure 2.7 shows the learning behaviour for the non-linear system (NLS). The dotted lines are the desired response and the solid lines represent the actual output of the system. The system converged in 877 iterations.

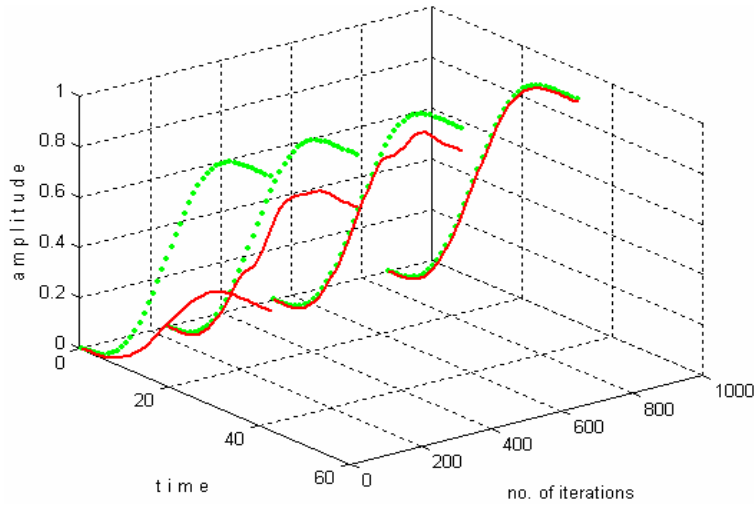


Figure 2.7: A non linear system using OSATILC.

The same system when tested on classical ILC technique [125] showed mixed results. The performance was very good in the beginning as error decreased sharply but it started increasing later on and then became unstable. The response is shown in figure 2.8.

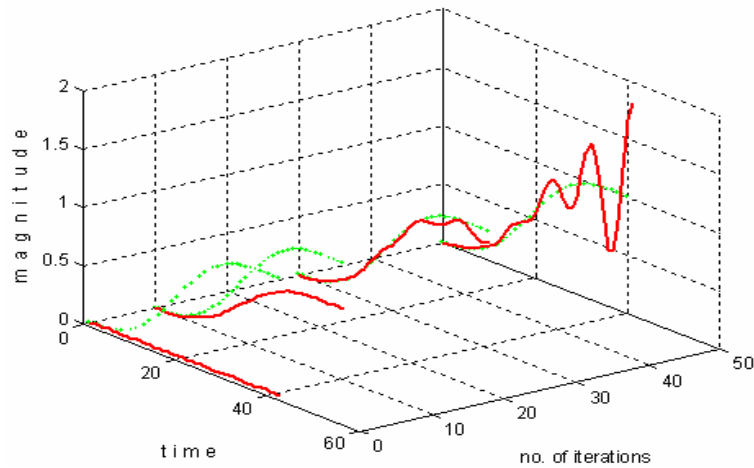


Figure 2.8: Performance of the approach using the classical ILC given by Arimoto.

For the classical ILC, the norm of error as iterations were increased is plotted in figure 2.9.

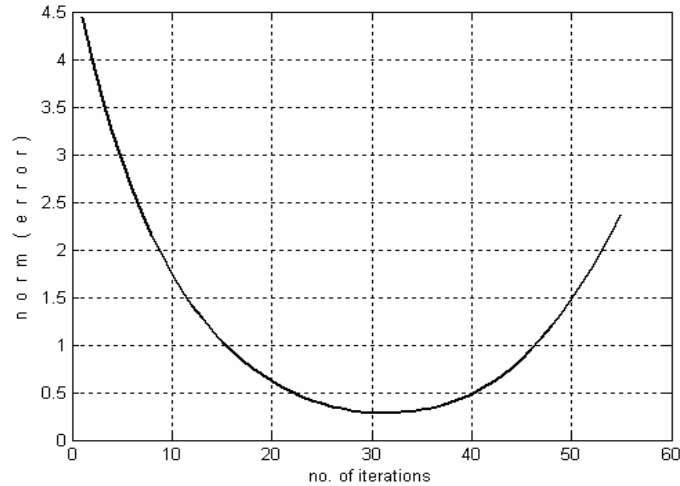


Figure 2.9: Error vs. number of iterations.

The error decreases sharply but then at about 35<sup>th</sup> iteration starts to increase. The system would not have diverged if the convergence criteria was relaxed to  $\|e_j(k)\| < 0.4$ . In fact most approaches have this problem of divergence if the convergence criterion is tightened i.e. the permissible error is reduced beyond a certain limit. With the proposed scheme the output reaches the desired output and does not alter as the numbers of iterations are increased further or permissible error is reduced. The four systems, converged in 2086, 686007, 5083 and 877 iterations, respectively. The behaviour of OSATILC for the four systems is presented in figure 2.10.

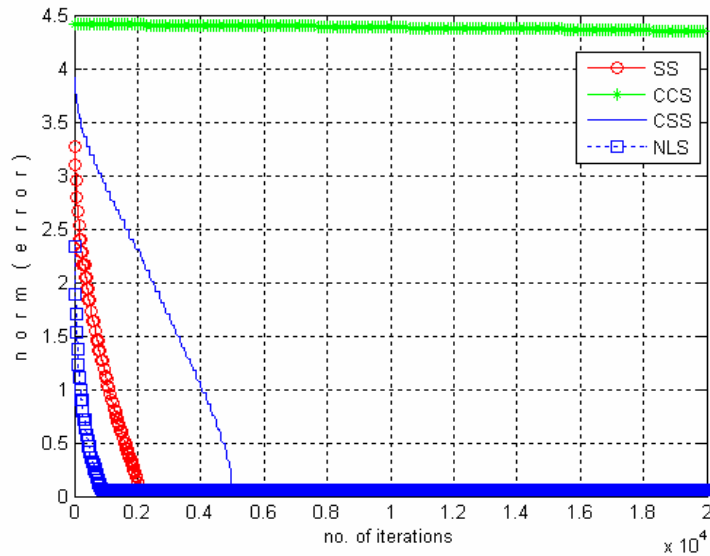


Figure 2.10: Behaviour of error for SS, CCS, CSS and NLS

A highly non-linear problem of an inverted pendulum on a cart (Appendix A) was also used to test OSATILC. The pendulum was required to track a sigmoid trajectory and then maintain an angle of -1 radians. Figure 2.11 shows the angle of the rod (pendulum) and the input learnt to achieve the desired path.

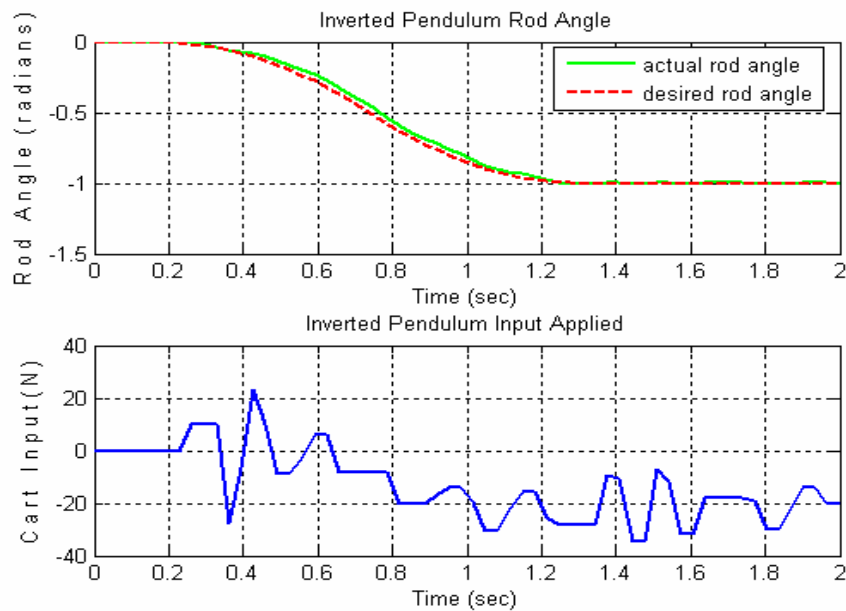


Figure 2.11: The pendulum maintaining a desired angle with the learnt input.

OSATILC showed better performance than the classical ILC but the convergence time needed to be improved.

## **2.6 Multiple Samples at a Time Iterative Learning Controller (MSATILC)**

To increase the convergence rate, more samples needed to be taken in to account. This resulted in the modification of rule 3 given for OSATILC. The rules for MSATILC are:-

- (1) Samples are given a priority from sample 1 to N, where sample 1 has the highest priority and sample N has the least priority.
- (2) There should be no gap between the samples, i.e. samples 1 to 10 should converge before sample 11 is considered for convergence.
- (3) The input changes for all those samples that have not converged.
- (4) Once a sample input has converged i.e. the input value of  $k^{\text{th}}$  sample is calculated, the value of that input should be applied as a starting value for  $k + 1^{\text{th}}$  sample. i.e.

$$u_j(k+1) = u_j(k)$$

- (5) Once the input for a sample has been calculated, no change in that input will be made in any subsequent iterations i.e.

$$u_{j+1}(1...k_c) = u_j(1...k_c)$$

Where,  $k_c$  represents the number of converged samples.

Rule (3) is the heart of MSATILC. Instead of, only changing the input of the sample under consideration, all the sample inputs that have not converged are changed.

Results from same four systems, as were used for OSATILC, are presented for comparison. The results in this section were taken with values of  $K_0 = 0.1$ ,  $K_1 = 0$  and  $T_s = 0.1$ . Figure 2.12 shows the learning behaviour for a simple system,  $G_1(z)$ . The

dotted lines are the desired response and the solid lines represent the actual output of the system.

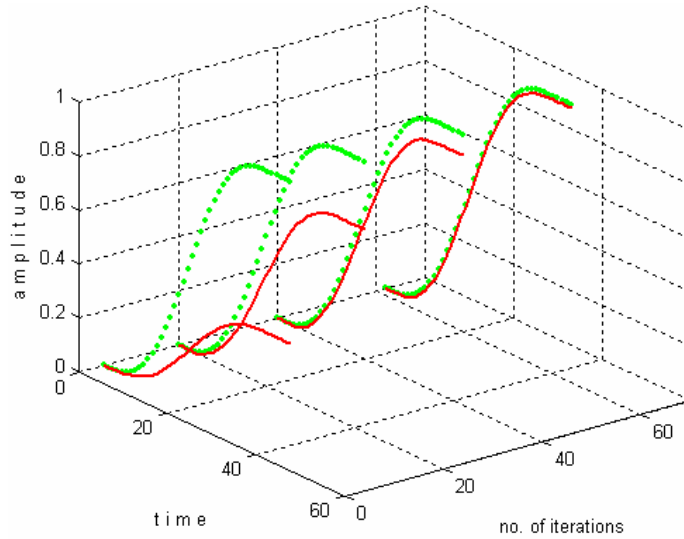


Figure 2.12: Desired output and actual output as iterations increase for  $G_1(z)$ .

The system converged in 63 iterations. MSATILC performed better than OSATILC as it took more samples into account. The key in both schemes (OSATILC and MSATILC) is that the input for samples that have converged is not changed, unlike other ILC schemes presented in literature. This guarantees that once convergence is achieved, for a range of samples (in priority order) it never diverges. Figure 2.13 shows the error plot for  $G_1(z)$ ,  $G_2(z)$  and  $G_3(z)$  as iterations increase.

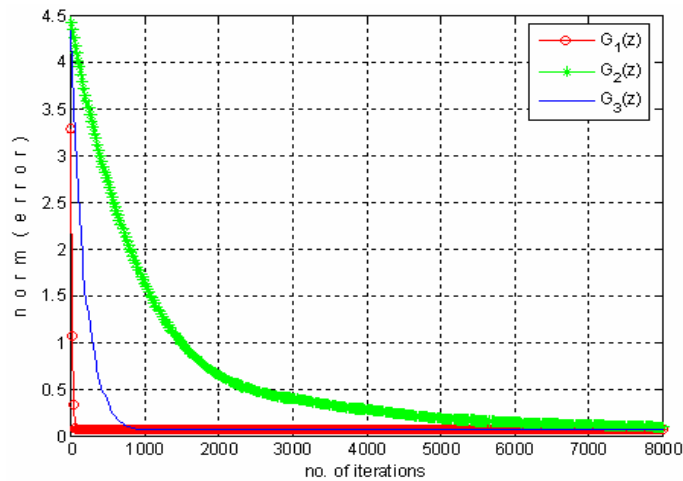


Figure 2.13: Error curves for  $G_1(z)$ ,  $G_2(z)$  and  $G_3(z)$ .

All the systems show continuous decrease in error as iterations increase. The error behaviour for NLS is exhibited in figure 2.14.

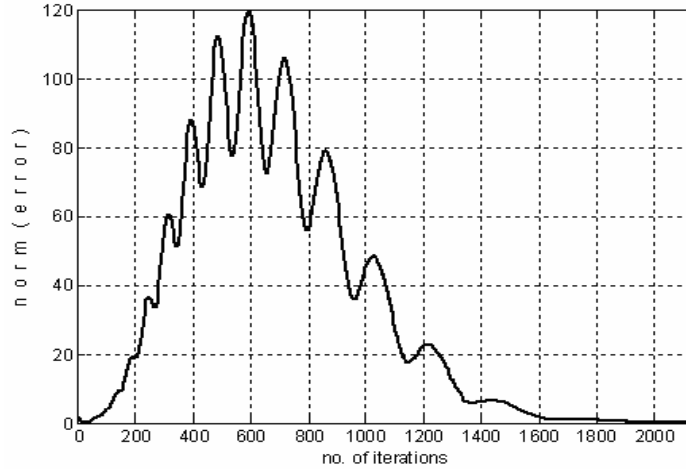


Figure 2.14: Behaviour of error for NLS.

Changing inputs for more samples have an unwanted effect of temporarily increased error, similar to the effect seen in conventional ILC. This increase is catered for by the priority assigned and by not changing the inputs of samples which have converged. Therefore the error is pulled back in subsequent iterations. It took 63, 11629, 886 and 2141 iterations to converge for the four systems.

Both OSATILC and MSATILC used a randomly chosen value of  $K_0$  and  $K_1$ . If we can make an initial guess at these values, just as the human operator experiment revealed, we can reduce the number of iteration further.

## **2.7 Modified Multiple Samples At a Time Iterative Learning Controller (MMSATILC)**

This modified approach makes an initial guess at the values of  $K_0$  and  $K_1$ . The initial value of  $K_0$  is taken as the DC gain of the system while the initial value of  $K_1$  is 10% of the DC gain of the system.



For sampled systems the DC gain is given by

$$\lim_{z \rightarrow 1} zG(z) = G(1)$$

If the system is not known, a trial input can be given to the system to calculate its DC gain. Results from the same four systems, as used in the previous section, are presented to see the effect of this guess.

The performance of this modified approach as iterations increased for  $G_2(z)$  is given in figure 2.15. The results in this section were taken with values of  $K_0 = 0.1$ ,  $K_1 = 0$  and  $T_s = 0.1$ .

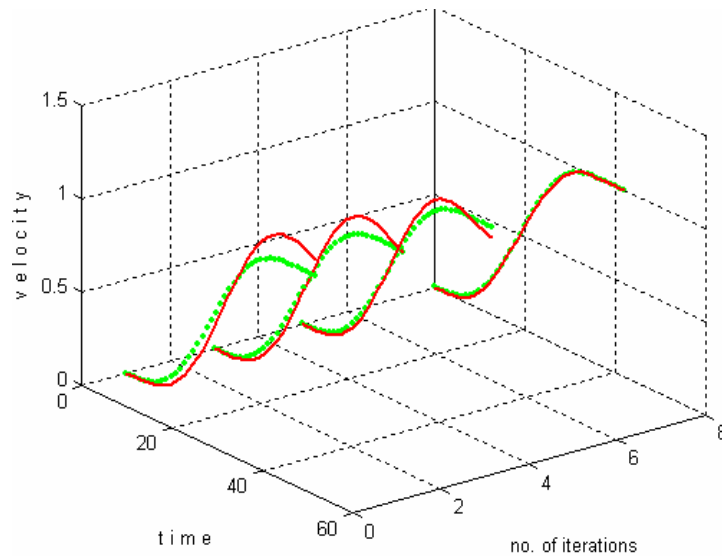


Figure 2.15: Output trying to track the desired output for  $G_2(z)$ .

The error performance of the four systems under consideration is given in figure 2.16 and figure 2.17.

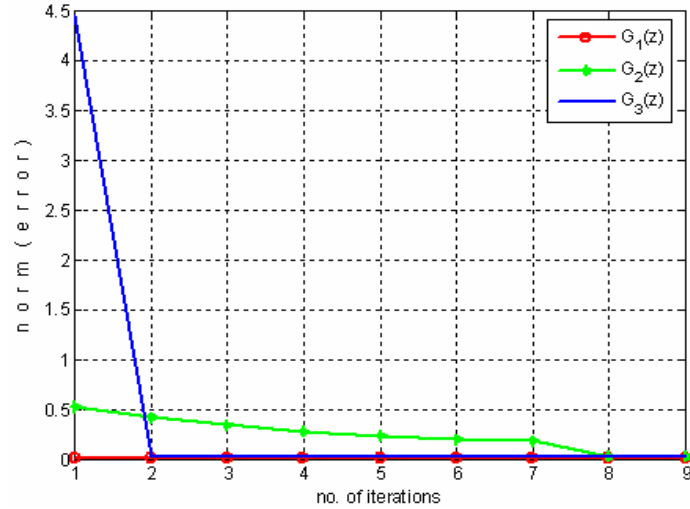


Figure 2.16: Behaviour of error  $G_1(z)$ ,  $G_2(z)$  and  $G_3(z)$ .

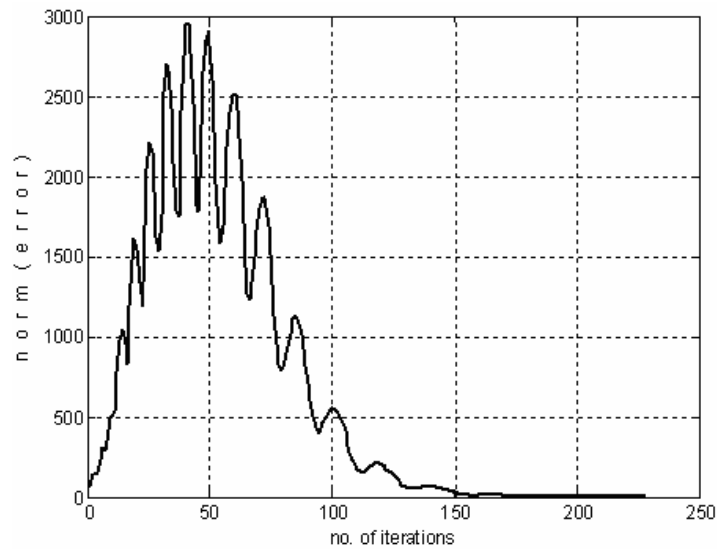


Figure 2.17: Error performance with NLS.

The figures show marked improvement in performance over OSATILC. The rate of convergence is much better. To see this improvement in performance, a comparison of results from the three schemes is made.

## 2.8 Comparative Results

The results obtained by applying the classical ILC, OSATILC, MSATILC and MMSATILC on the four selected systems are presented in this section for quick comparison. Table 2.1 presents these results for convergence criteria of  $\|e_j(k)\| < 0.01$ . Here DNC stands for ‘Did Not Converge’.

Approach System	Classical ILC (iterations)	OSAT ILC (iterations)	MSAT ILC (iterations)	MMSAT ILC (iterations)
SS	146	2086	63	1
CCS	DNC	686007	11629	8
CSS	1335	5083	886	2
NLS	DNC	877	2141	228

Table 2.1: A quick comparison of results.

All the results were taken under similar conditions. Classical ILC was not able to converge for CCS and NLS. Though OSATILC converged for all systems, numbers of iterations were very high. These iterations were significantly reduced in MSATILC and further reduced in MMSATILC. OSATILC gives us the power to apply ILC technique to systems, in real time as each sample can be treated as one trial. MSATILC and MMSATILC can divide an input into smaller regions for better control and convergence. The other advantage of the proposed schemes is that the desired error can be made as small as possible, though at the cost of more iterations.

## 2.9 Two Degree of Freedom Tracking Platform

To test OSATILC, MSATILC and MMSATILC for real time tasks, a simple mathematical model of a two degree of freedom platform was developed. The platform

carries a camera mounted on it. The camera takes the pictures of a scene at regular intervals and thus tracks an object of interest. The model was named Two Degree of Freedom Tracking Platform (2DOFTP).

A few assumptions were made for developing a model of this 2DOFTP.

- (a) The two DC motors have rigid rotor and shaft.
- (b) There is no backlash and gear slip.
- (c) The pixels of camera are continuously distributed i.e. no gap exists between two consecutive pixels.
- (d) The camera has a pixel resolution of 320x240.
- (e) The two motors are uncoupled.
- (f) The viewing area is at a fixed distance from the platform.

Both the motors, called the pitch motor and the yaw motor, were assumed to be of the same specifications. The motors parameters are given below.

J (moment of inertia of motor) =3.2284E-6;

b (damping ratio of the mechanical system) =3.5077E-6;

K (electromotive force constant) =0.0274;

R (electric resistance) =4;

L (electric inductance) =2.75E-6;

The motor's transfer function is as given below

$$H(s) = \frac{0.0274}{8.878 \times 10^{-12} s^2 + 1.291 \times 10^{-5} s + 0.0007648s} \quad (2.47)$$

The entire setup, showing its degrees of freedom, is exhibited in figure 2.18.

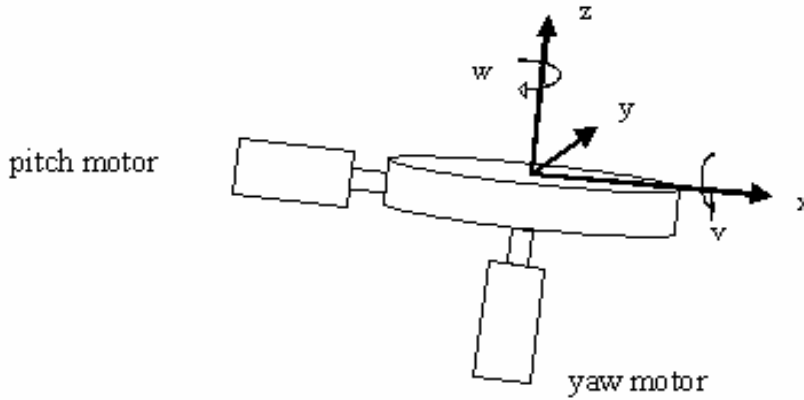


Figure 2.18: Movement angles of the platform.

In figure 2.18,  $x, y, z$  are the axis of the Cartesian coordinate system. The two degrees of freedom are the pitch ( $\nu$ ) and yaw ( $w$ ) angles.

The camera was assumed to have a viewing area of 2m x 2m. This means that there are 240 pixels to view 2m in vertical direction and 320 pixels to view 2m in horizontal direction. For this proposed viewing area, each pixel represents approx. =  $1/120 = 0.0083\text{m}$  in vertical direction and approx. =  $1/160 = 0.0063\text{m}$  in horizontal direction.

Another assumption made was that the object can move in this defined viewing area of 2x2 meter only. The pertinent geometry is explained in figure 2.19.

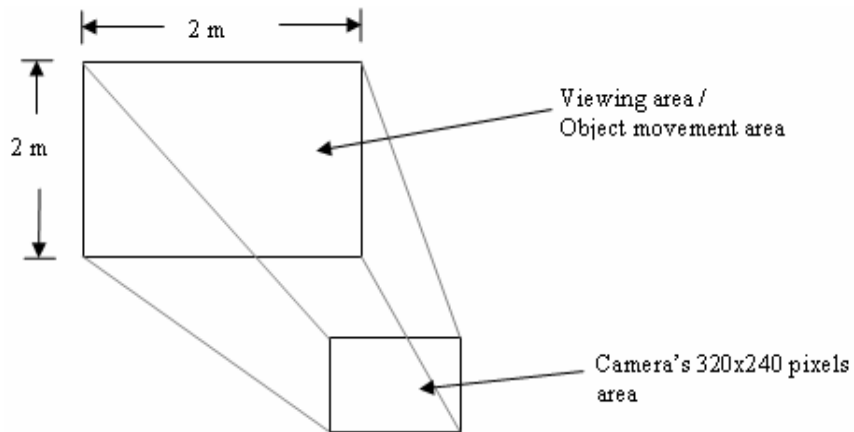


Figure 2.19: Mapping of the viewing area with the camera's pixel area.

The camera pixels were divided into four regions with (0,0) at the centre of camera as shown in figure 2.20.

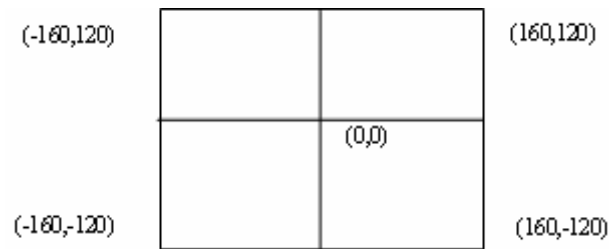


Figure 2.20: Division of camera's resolution into platform resolution.

The aim is to have the object at (0,0) of platform resolution or (160,120) of camera resolution. A conceptual view of an object being viewed by a camera, 1m away from the object, is described in figure 2.21.

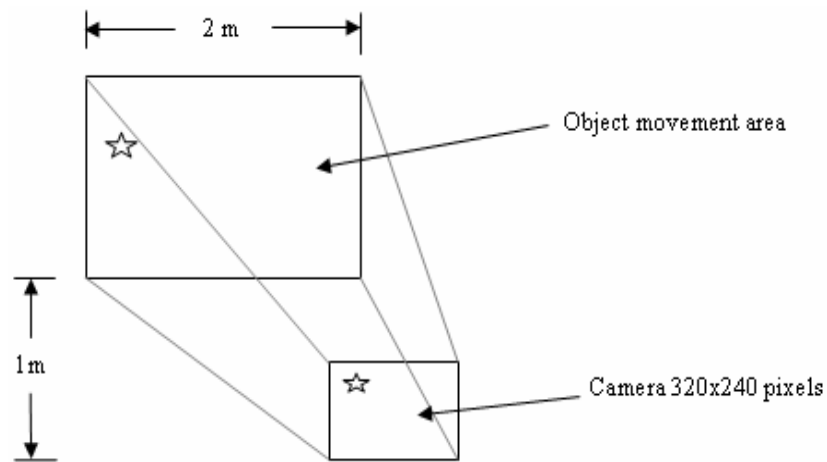


Figure 2.21: An object as viewed by the camera.

An object in viewing area, represented by a star, occupies some pixels in the camera. The position of the object is reckoned by the centre of the object, calculated by an image processing module. Defining the current x and y positions of the object to be px (pixels) and py (pixels), the error can be calculated as

$$ex_j(k) = 160 - px \tag{2.48}$$

$$ey_j(k) = 120 - py \tag{2.49}$$

The location (160,120) is the (0,0) of the platform resolution and is the centre of the camera. Figure 2.22 shows the trigonometric representation of an object at upper left corner of the viewing area.

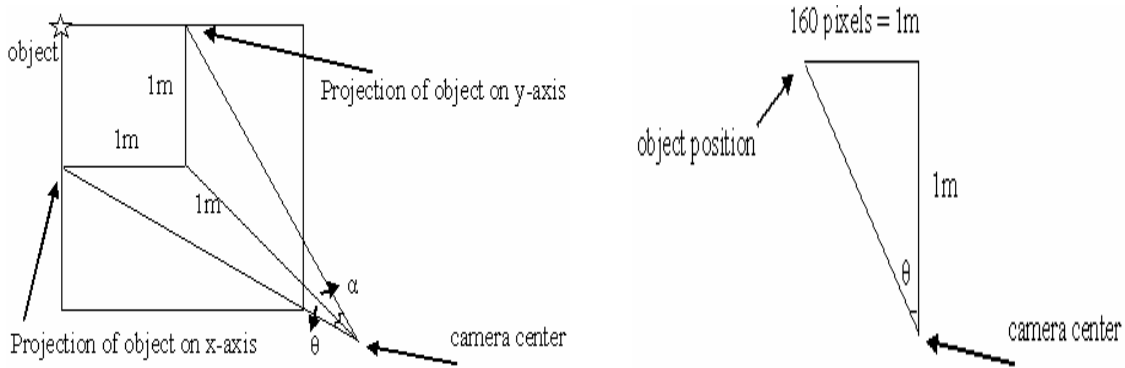


Figure 2.22: Trigonometric representation of object projected on x,y axis (left) and trigonometric representation of object projected on x axis (right).

The motor movements are labelled  $\alpha$  rad/sec and  $\theta$  rad/sec, for the pitch and yaw motors respectively. From the figure, tangent of the angle  $\theta$  can be written as

$$\tan(\theta) = \frac{1}{1}$$

giving

$$\theta = \tan^{-1}(1) = 0.7854 \text{ radians} = 45\text{degrees}$$

This means that the camera can view  $\pm 45^\circ$  in the yaw directions. Using this trigonometric representation and its equivalent mathematical formulation, objects can be moved in the viewing area, and the required motor movements can be calculated. For example, for an object at (140,120) camera coordinates, the error is computed as

$$ex_j(k) = 160 - 140 = 20 \text{ pixels} = 20 \times 0.0063 = 0.126\text{m}$$

$$ey_j(k) = 120 - 120 = 0 \text{ pixels} = 0 \times 0.0083 = 0 \text{ m}$$

Meaning that, the object is 0.126m away from the centre in x direction and 0 m away from the centre in y direction.

Therefore, the yaw motor has to be moved  $\theta = \tan^{-1}(0.125) = 0.1253$  radians. Hence the motor should move left to position the object at camera centre. The appropriate inputs to the motors were calculated by the proposed ILCs.

Once the movements have taken place, the new platform origin is recalculated by the following equations.

$$\text{new\_platform\_origion\_x} = 160 - \text{pixels moved because of yaw motor} \quad (2.50)$$

$$\text{new\_platform\_origion\_y} = 120 - \text{pixels moved because of pitch motor} \quad (2.51)$$

Using this 2DOFTP, the real time performance of any scheme can be tested. The results recorded using OSATILC are presented in next section.

### 2.9.1 Simulation results for 2DOFTP

The results from one of the proposed controller OSATILC are presented in this section. MSATILC and MMSATILC showed similar results. The behaviour of OSATILC while tracking an object at (61,1) is demonstrated in figure 2.23. For this performance the values of gains assumed were  $K_0 = 0.1$  and  $K_1 = 0$ . In this figure the object does not move.

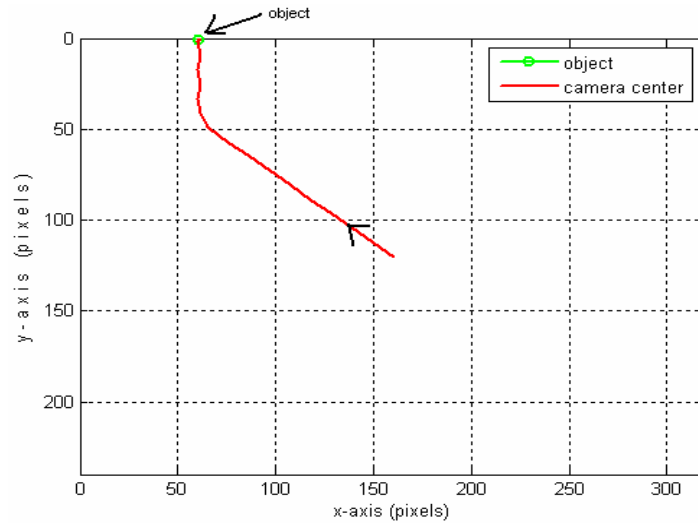


Figure 2.23: Tracking behaviour as an object at (61,0) is being tracked.



The frame rate of the camera was assumed to be 5 frames per second. The frame rate can be increased or decreased (if hardware allows) as per requirement. A typical camera can take 20 frames per second, easily now days. The result of another such simulation, with gain,  $K_0 = 0.1$ , and the object moving, is shown in figure 2.24.

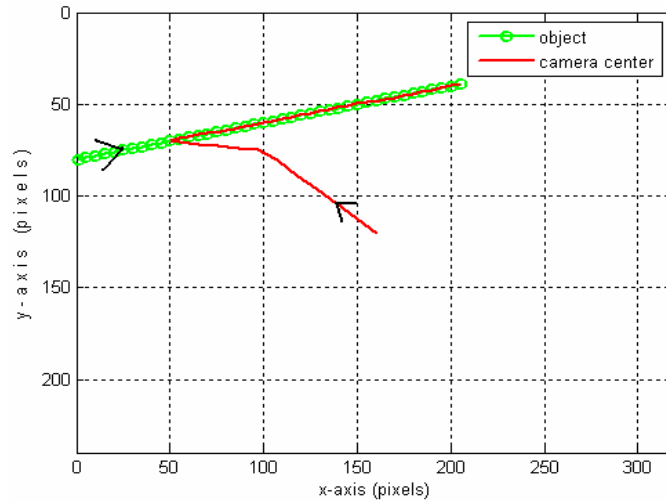


Figure 2.24: A moving object being tracked using the Two degree of freedom model.

The object started its journey at (1,80) pixels i.e. (159,40) pixels away from the camera centre. It then slowly moved upwards. The error performances, for this moving object, in both x and y axis is demonstrated in figure 2.25.

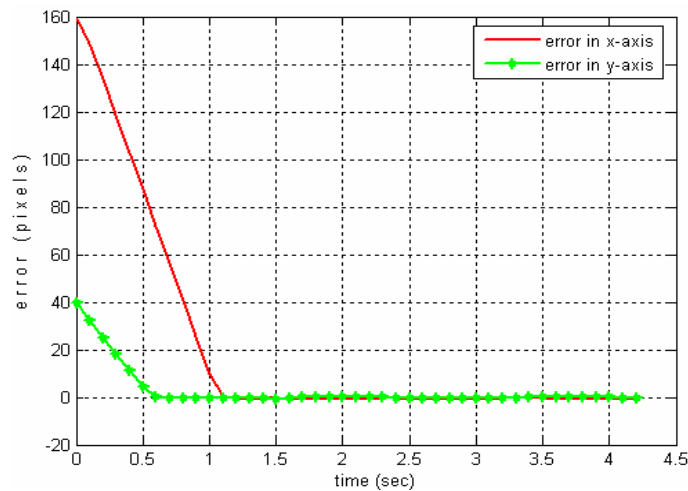


Figure 2.25: Error behaviour as moving object is tracked.

As can be seen from plots (2.24) and (2.25), both static and moving objects were successfully tracked using the OSATILC. The speed of tracking can be improved by increasing the frame rate.

Once the scheme was tested through simulations and fine-tuned, it was ready to be tested through a practical setup.

## **2.10 Experimental Setup**

The practical setup was similar in functionality to the 2DOFTP model. The platform also had a camera mounted on it and had the capability to move in different degrees of freedom with great accuracy. The platform was \$60,000 equipment with very high accuracy. The platform was M-850 Hexapod.

### **2.10.1 Six Degree of Freedom Hexapod**

The M-850 Hexapod is a 6-axis positioning system by Physik Instrumente (PI). The M-850 hexapod provides motion in 3 translation axes and 3 rotational axes. A picture of the hexapod is presented in figure 2.26.



Figure 2.26: Six degree of freedom Hexapod by PI.

Repeatability for a six-axis move is  $\pm 1 \mu\text{m}$  in Z direction and  $\pm 2 \mu\text{m}$  in X and Y directions respectively. Repeatability for rotational axes are  $\pm 10 \mu\text{rad}$ . The three translational movements X,Y and Z are performed on a straight path. The three rotational movements produce three rotational angles u , v , w called rotation angle, pitch angle and yaw angle respectively. All these movements are shown in figure 2.27.

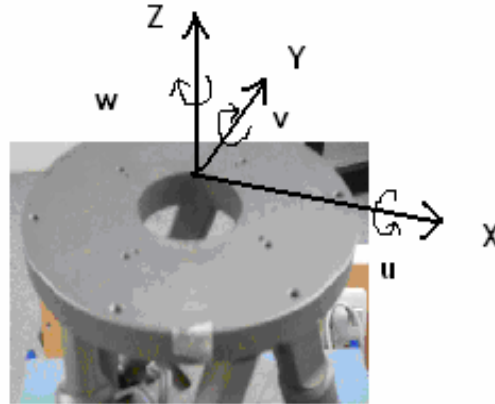


Figure 2.27: Hexapod coordinates and movements.

M-850 can lift a load of 200Kg. The hexapod system consists of Hexapod mechanism, a movable platform supported by six linear actuators, the control electronics and the connecting cables. Movements in all six degrees of freedom can be accomplished using the DC-motor-driven linear actuators, which extend and contract the struts of the Hexapod platform. The actuators have a backlash-free spindle combined with a backlash-free gear head. The Hexapod can be controlled by a PC based 6-axis DC motor-controller. The micrometer accuracy of the platform makes it an ideal choice for testing and implementing high performance tracking and control algorithms. The components are mounted free of backlash which gives the mechanical system exceptional stiffness and excellent positioning repeatability. The material and lubricants used also assure long term operation in different conditions and temperature ranges.

To create a link between the Hexapod and the PC, control software was first developed using the Borland C compiler. It was fine tuned and then developed in MATLAB for ease of implementation of the proposed schemes. A Graphical user interface (GUI) was also developed to monitor the performance in real time.

The complete setup with the GUI is shown in figure 2.28.

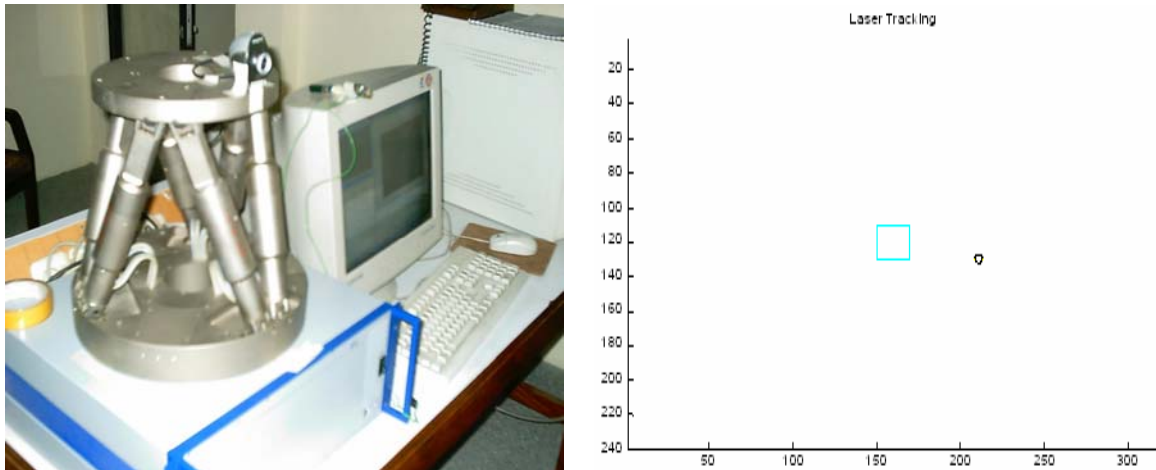


Figure 2.28: The complete Hexapod set up (left) and the GUI (right).

The figure on the left shows the camera mounted on the hexapod facing the target area. The GUI on the right is a simple way of looking at the object, as seen by the camera and gauging control algorithm performance. The centre rectangular box, on the right picture, is the locking area. The object is considered locked if with in this area. The circle is the object spotted by the camera and the image processing module.

Using OSATILC we now propose a scheme to control the Hexapod in real time [126].

### 2.10.2 Proposed Approach

The proposed scheme is shown in figure 2.29.

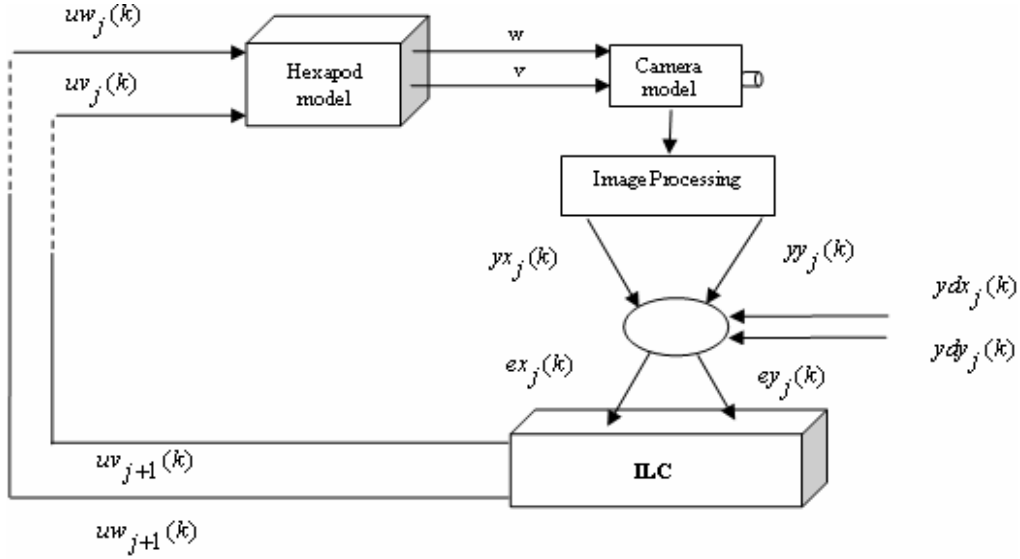


Figure 2.29: Block diagram representation of the proposed scheme.

Here current input for the pitch angle ( $uv_j(k)$ ) and current input for the yaw angle ( $uw_j(k)$ ), are supplied to the hexapod. The hexapod platform moves, moving the mounted camera with it. The camera was taking snapshots of the scene regularly. Every snapshot goes through an image processing module for recognition and positioning of the target object. The desired x,y position in an x,y coordinate system are labelled  $ydx_j(k)$  and  $ydy_j(k)$ . This desired position for tracking purposes is the centre of the camera. The output of the hexapod in terms of the x,y coordinate system are measured as  $yx_j(k)$  and  $yy_j(k)$ . Using the desired platform position and the current hexapod position the error is calculated as  $ex_j(k)$  and  $ey_j(k)$ . These values are used by the proposed ILC to calculate the next pitch and yaw inputs ( $uv_{j+1}(k)$ ,  $uw_{j+1}(k)$ ). The input modification equation is re written for the two degrees of freedom as

$$uv_{j+1}(k) = uv_j(k) + \Delta uv_j(k) \quad (2.52)$$

$$uw_{j+1}(k) = uw_j(k) + \Delta uw_j(k) \quad (2.53)$$

The difference between the desired and actual object positions are given by the error equations

$$ex_j(k) = ydx_j(k) - yx_j(k) \quad (2.54)$$

$$ey_j(k) = ydy_j(k) - yy_j(k) \quad (2.55)$$

The errors, the desired output and the actual output are in pixels. The camera could have different resolution as per accuracy requirements. For a 320x240 camera resolution the target plane was divided into four regions, as in the development of 2DOFTP. Those regions are shown in figure 2.20. The aim was to have the target as close to (0,0) as possible. Therefore, an object at coordinate (0,0) is perfectly tracked while an object at (60,0) is 60 pixels away in x direction. For the presented experiment the system operated in two modes; the learning mode and the tracking mode. The learning mode learns the gains iteratively. The hexapod is made to go to its initial position at every iteration i.e.  $yx_j(1) = yx_1(1)$  and  $yy_j(1) = yy_1(1)$ . Here  $(yx_1(1), yy_1(1))$  is the initial position of the hexapod at first iteration. The proposed gain modification scheme is given below

$$Kv_{j+1}(k) = Kv_j(k) + \mu ex_j(k) \quad (2.56)$$

$$Kw_{j+1}(k) = Kw_j(k) + \mu ey_j(k) \quad (2.57)$$

$$Kv1_{j+1}(k) = Kv1_j(k) + \mu ex_j(k+1) \quad (2.58)$$

$$Kw1_{j+1}(k) = Kw1_j(k) + \mu \Delta ey_j(k+1) \quad (2.59)$$

Where  $\mu$  is the step size parameter. The  $Kv$ ,  $Kv1$  are the pitch gains and  $Kw$ ,  $Kw1$  are the yaw gains. The next inputs to the hexapod were calculated using

$$uv_{j+1}(k) = uv_j(k) + Kvs_j(k)ex_j(k) + Kv1s_j(k)ex_j(k+1) \quad (2.60)$$

$$uw_{j+1}(k) = uw_j(k) + Kws_j(k)ey_j(k) + Kw1s_j(k)ey_j(k+1) \quad (2.61)$$

Here  $Kvs$ ,  $Kvls$ ,  $Kws$  and  $Kwls$  are the stored pitch and yaw gains during the previous learning cycle. The tracking mode used the learnt input values, to track the laser. The initial state of one learning cycle is the final state of the previous learning cycle i.e.  $y_{j+1}(1) = y_j(N)$  and  $yy_{j+1}(1) = yy_j(N)$ .

### 2.10.3 Results

A  $CO_2$  laser beam spot was chosen as a potential target. In the learning mode the values of  $Kv_j(k)$  and  $Kw_j(k)$  were learnt against different object positions from the centre. One such learnt set of values for  $Kw_j(k)$  are shown in figure 2.30.

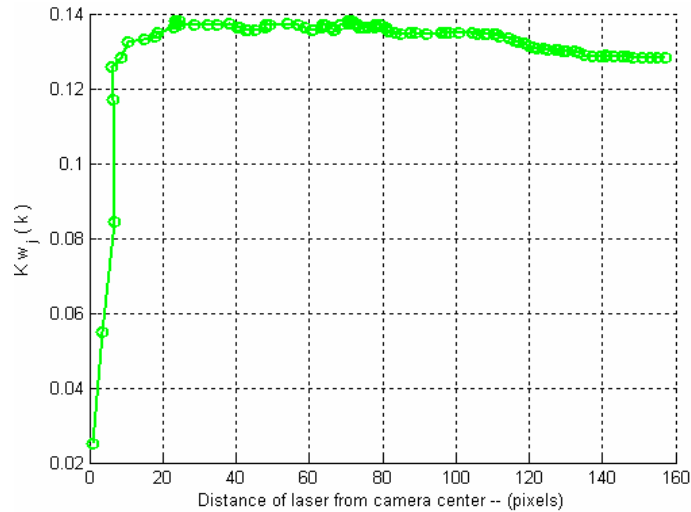


Figure 2.30: Learnt values of  $Kw_j(k)$  in the learning phase.

The small variations in the shape of the plot in figure 2.30 are due to flickering of laser beam. This produced small variations in the illuminated spot and hence caused small error in recognizing the centre of the spot by the image processing module. In almost all cases the error is not more than  $\pm 1$  pixel. These learnt values of the pitch and yaw gains were stored in memory. Using these stored values, the Hexapod was made to track and

follow different moving targets. A xy plot of one such target being chased is indicated in figure 2.31.

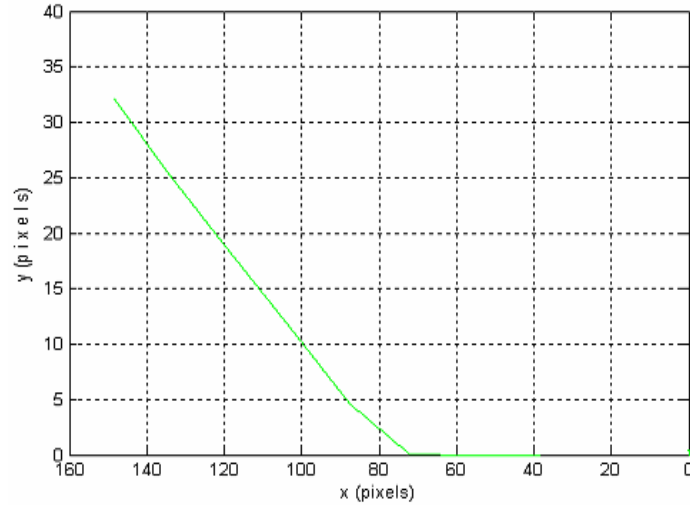


Figure 2.31: A xy plot of a target being chased.

The target was initially at (150,32) pixels. It was successfully tracked in 10 frames. The error generated during this tracking is described in figure 2.32. Here number of iterations can be thought of as number of times the input was adjusted.

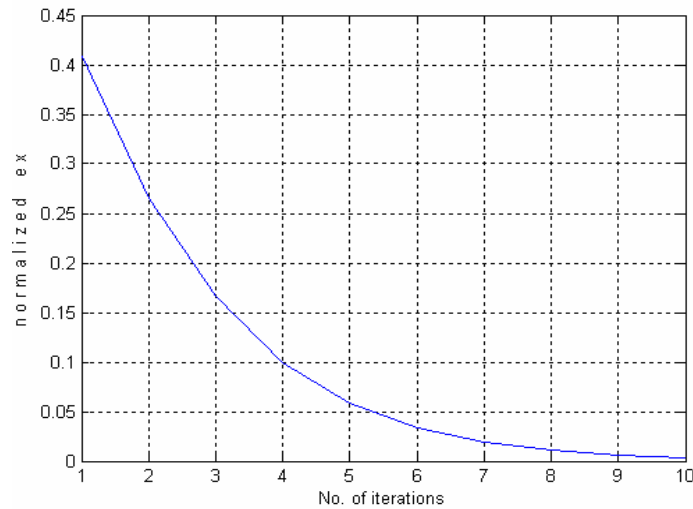


Figure 2.32: Normalized error as the laser is tracked.



The plots show the laser being successfully tracked. For a 20 frames per second camera the object was tracked with in 0.5 seconds.

## **2.11 Summary**

Basic framework for Iterative Learning Control (ILC) is developed in this chapter. Characteristics of human leaning, gathered from experimental results, helped in the development of this frame work and consequently, three iterative learning controllers were deigned and presented in this chapter. Stability and convergence criteria for these ILCs are also established. The three controllers OSATILC, MSATILC and MMSATILC are tested through simulations. These controllers can also be used in real time applications. One such application in which a Six Degree of Freedom Hexapod tracks a laser spot is also presented.

Most ILC controllers are still based on the classical ILC presented by Arimoto. All the three controllers presented in this chapter performed better than the classical ILC. Research shows that there are always some optimal values of gains that give us best results. We need to have a procedure to find these optimal values. Also, in real world applications there are usually changes in system dynamics, due to wear and tear. The desired response requirements also vary from time to time. Both of these cases require a change in the learning law for optimal performance. Hence the Iterative Learning scheme needs to be adaptive.

The next chapter develops such schemes.

### **3 INTELLIGENT CONTROLLERS USING ADAPTIVE ITERATIVE LEARNING**

In recent years, a lot of research efforts have been directed towards self-learning and adaptable systems. One of the very promising methodology for self-learning control systems is Iterative Learning Control (ILC) which is an algorithm capable of tracking a desired trajectory, within a specific period of time. Conventional ILC algorithms have the problem of relatively slow convergence rate and lack of adaptability. Most of the current ILC methods adopt fixed learning laws where only control input scheme is modified during learning iterations. Since only control input is changed during learning for a specified trajectory, it is difficult to generalize the knowledge learned from one desired trajectory to another, even if they are similar. Therefore, this type of learning control is restrictive to repetitive tasks which have same desired trajectory. A survey of current literature indicates that ILC schemes in tandem with adaptive control are growing into a very promising research area [145]. This chapter continues from the research work described in chapter 2, to evolve adaptive ILCs.

We begin with an introduction of the theory and present a 2-D learning process similar to the one given in [152]. This process considers time horizon and iteration axis simultaneously, giving us more dimensions for control [148]. Based on this theory, a specific mathematical framework is formulated upon which different ILC methodologies are developed. The methodologies makes use of system identification technique, steepest descent method, different conventional and custom built cost functions and a learning gain method to introduce adaptivity. The problem of slow convergence is resolved by adaptive control laws. The model dependency of the current ILC schemes is dealt by an identification approach. Simulations of linear and non-linear systems are presented to illustrate the design procedure and to confirm the effectiveness and robustness of the algorithms. A Quanser's DC motor kit is also used to demonstrate the usefulness of these adaptive schemes for real time applications. The optimal gain values are calculated using the steepest descent approach. Convergence and stability conditions are also derived.

Diminishing cost and increasing computational power of computers and embedded systems make the implementation of such schemes highly feasible.

The main features of such a scheme are

- (1) It does not require an inverse dynamic model of the controlled system for designing the control scheme as many other ILC schemes do. Instead, it tries to estimate the learning gains by using input and output data. Both learning law and the control input sequence are modified to improve the tracking performance of the system.
- (2) No prior knowledge of the system is required. Therefore, the same scheme can be used for different systems as the control algorithm will adapt to the system and the control gain(s) will be learned automatically.
- (3) The approach can be used in learning tasks where the desired trajectory changes during operation.
- (4) The knowledge learnt from one particular task can be utilized in similar tasks reducing the learning time.

We now develop basic framework for these adaptive controllers.

### **3.1 Adaptive Learning Controllers**

A Single Input Single Output (SISO) discrete system is shown in figure 3.1.

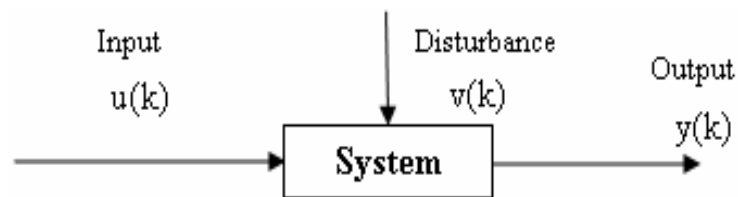


Figure 3.1: A SISO system with disturbances.

The input  $u(k)$ , output  $y(k)$  and disturbance  $v(k)$  are taken as

$$\mathbf{u}(k) = [u(1) \quad u(2) \quad \dots \quad u(N)]$$

$$\mathbf{y}(k) = [y(1) \quad y(2) \quad \dots \quad y(N)]$$

$$\mathbf{v}(k) = [v(1) \quad v(2) \quad \dots \quad v(N)]$$

Where  $N$  is the total number of samples.

Input and output are observed while disturbance in most cases is not observed. The disturbance is considered as generated by filtered white noise. Taking  $z$  as a shift operator and assuming disturbance to be generated by filtered white noise  $\varepsilon(z)$  a SISO system can be represented as

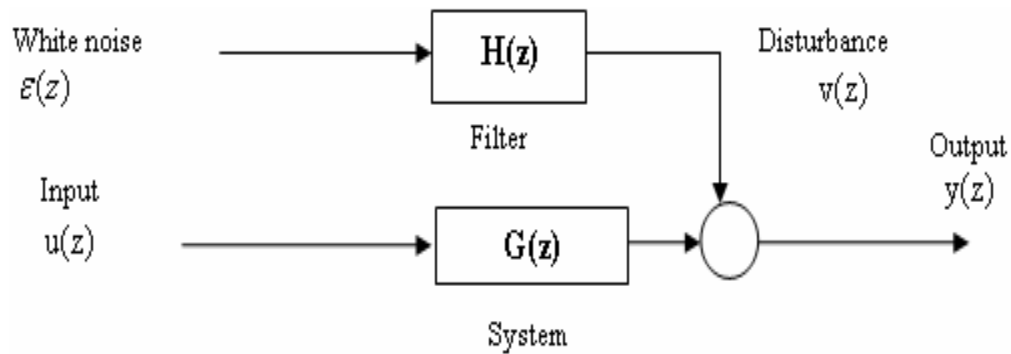


Figure 3.2: A SISO system with disturbances.

This can be written in  $z$  transform form as

$$Y(z) = G(z)U(z) + H(z)\varepsilon(z) \tag{3.1}$$

There are many models available to represent this SISO system. One of the models using Autoregressive with exogenous variables (ARX) is

$$G(z) = \frac{B(z^{-1})}{A(z^{-1})} \tag{3.2}$$

$$H(z) = \frac{1}{A(z^{-1})} \quad (3.3)$$

Where

$$A(z^{-1}) = 1 + a_1 z^{-1} + \dots + a_{na} z^{-na} \quad (3.4)$$

$$B(z^{-1}) = b_1 z^{-1} + b_2 z^{-2} + \dots + b_{nb} z^{-nb} \quad (3.5)$$

Putting (3.2) and (3.3) in (3.1)  $\Rightarrow$

$$Y(z) = \frac{B(z^{-1})}{A(z^{-1})} U(z) + \frac{1}{A(z^{-1})} \varepsilon(z) \quad (3.6)$$

Using values of  $A(z^{-1})$  and  $B(z^{-1})$  from (3.4) and (3.5) and considering no extra delay and zero disturbances gives

$$Y(z) = \frac{b_1 z^{-1} + b_2 z^{-2} + \dots + b_{nb} z^{-nb}}{1 + a_1 z^{-1} + \dots + a_{na} z^{-na}} U(z) \quad (3.7)$$

This in difference equation format can be written as

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) - \dots - a_{na} y(k-n_a) + b_1 u(k-1) + b_2 u(k-2) + \dots + b_{nb} u(k-n_b) \quad (3.8)$$

Using this representation of the system we now develop different adaptive iterative learning controllers.

### **3.2 When System is Known (Approach-1)**

We now consider the case when complete knowledge about the system is available. For this approach zero initial conditions are assumed i.e.  $u(k) = 0$  for  $k < 0$  and  $y(k) = 0$  for  $k < 1$ , with at least a single sample system delay.

Observations  $[y(1) \ y(2) \ \dots \ y(N)]$  generated by applying an input sequence  $[u(0) \ u(1) \ \dots \ u(N-1)]$  can be written as

$$y(1) = -a_1 y(0) + b_1 u(0)$$

$$y(1) = b_1 u(0) \quad (3.9)$$

$$y(2) = -a_1 y(1) - a_2 y(0) + b_1 u(1) + b_2 u(0)$$

$$y(2) = -a_1 b_1 u(0) + b_2 u(0) + b_1 u(1)$$

$$y(2) = (-a_1 b_1 + b_2) u(0) + b_1 u(1) \quad (3.10)$$

$$y(3) = -a_1 y(2) - a_2 y(1) - a_3 y(0) + b_1 u(2) + b_2 u(1) + b_3 u(0)$$

$$y(3) = -a_1 (-a_1 b_1 u(0) + b_2 u(0) + b_1 u(1)) - a_2 b_1 u(0) + b_1 u(2) + b_2 u(1) + b_3 u(0)$$

$$y(3) = a_1 a_1 b_1 u(0) - a_1 b_2 u(0) - a_1 b_1 u(1) - a_2 b_1 u(0) + b_1 u(2) + b_2 u(1) + b_3 u(0)$$

$$y(3) = a_1 a_1 b_1 u(0) - a_1 b_2 u(0) - a_2 b_1 u(0) + b_3 u(0) - a_1 b_1 u(1) + b_2 u(1) + b_1 u(2)$$

$$y(3) = (a_1 a_1 b_1 - a_1 b_2 - a_2 b_1 + b_3) u(0) + (-a_1 b_1 + b_2) u(1) + b_1 u(2) \quad (3.11)$$

⋮

Or

$$\begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ y(4) \\ \vdots \end{bmatrix} = \begin{bmatrix} b_1 & 0 & 0 & 0 & \dots \\ -a_1 b_1 + b_2 & b_1 & 0 & 0 & \dots \\ a_1 a_1 b_1 - a_1 b_2 - a_2 b_1 + b_3 & -a_1 b_1 + b_2 & b_1 & 0 & \dots \\ \vdots & a_1 a_1 b_1 - a_1 b_2 - a_2 b_1 + b_3 & -a_1 b_1 + b_2 & b_1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \dots \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ u(3) \\ \vdots \end{bmatrix} \quad (3.12)$$

This can be written in compact form as

$$\mathbf{Y} = \mathbf{G}\mathbf{U} \quad (3.13)$$

Where

$$\mathbf{Y}^T = [y(1) \quad y(2) \quad \dots \quad y(N)] \quad (3.14)$$

$$\mathbf{U}^T = [u(0) \quad u(1) \quad \dots \quad u(N-1)] \quad (3.15)$$

And

$$\mathbf{G} = \begin{bmatrix} b_1 & 0 & 0 & 0 & \dots \\ -a_1 b_1 + b_2 & b_1 & 0 & 0 & \dots \\ a_1 a_1 b_1 - a_1 b_2 - a_2 b_1 + b_3 & -a_1 b_1 + b_2 & b_1 & 0 & \dots \\ \vdots & a_1 a_1 b_1 - a_1 b_2 - a_2 b_1 + b_3 & -a_1 b_1 + b_2 & b_1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (3.16)$$

For the next iteration we can also write (3.13) as

$$\mathbf{Y}_{j+1} = \mathbf{G}\mathbf{U}_{j+1} \quad (3.17)$$

Assuming that the desired output sequence,  $[yd(1) \quad yd(2) \quad \dots \quad yd(N)]$ , is given.

The input sequence applied and the output sequence generated because of it, give rise to a sequence of residuals.

$$e(1) = yd(1) - y(1)$$

$$e(2) = yd(2) - y(2)$$

$\vdots$

$$e(N) = yd(N) - y(N)$$

Or

$$\mathbf{E} = \mathbf{Y}_d - \mathbf{Y} \quad (3.18)$$

Where

$$\mathbf{E}^T = [e(1) \quad e(2) \quad \dots \quad e(N)] \quad (3.19)$$

$$\mathbf{Y}_d^T = [yd(1) \quad yd(2) \quad \dots \quad yd(N)] \quad (3.20)$$

The two dimensional vector notation for input, output, error and desired output are  $\mathbf{U}_j(k)$ ,  $\mathbf{Y}_j(k)$ ,  $\mathbf{E}_j(k)$  and  $\mathbf{Y}_d_j(k)$ . For ease of mathematical representation these vectors will also be written as  $\mathbf{U}$ ,  $\mathbf{Y}$ ,  $\mathbf{E}$  and  $\mathbf{Y}_d$ .

In the next section an adaptive law is developed to find the optimal value of gain. For this development a novel cost function is proposed.

### 3.2.1 Gradient descent for adaptive gain(s)

Gradient descent approach was used to find an adaptive mechanism for estimating gains. For the approaches in this chapter, it is assumed, without any loss of generality, that  $K_0 = K$  and  $K_1 = 0$ . An important modification in the cost function is made to assist in the development of the ILC algorithms i.e. instead of eliminating error in the current iteration, elimination of error in the next iteration is proposed. Taking  $K$  as an unknown parameter the cost function  $J(K)$  for minimizing sum of error square for next iteration can now be written as

$$J(K) = \sum_{k=1}^N \left[ e_{j+1}(k) \right]^2 = \mathbf{E}_{j+1}^T \mathbf{E}_{j+1} \quad (3.21)$$

Where  $\mathbf{E}_{j+1}$  is the error vector for the next iteration and is given by equation

$$\mathbf{E}_{j+1} = \mathbf{Y}_d - \mathbf{Y}_{j+1} \quad (3.22)$$

As the aim is to reduce error in the next iteration, we can write

$$\lim_{j \rightarrow \infty} \left\| \mathbf{E}_{j+1} \right\| \rightarrow 0$$

Using (3.22) and (3.17) in (3.21),  $\Rightarrow$

$$J(K) = (\mathbf{Y}_d - \mathbf{G}\mathbf{U}_{j+1})^T (\mathbf{Y}_d - \mathbf{G}\mathbf{U}_{j+1}) \quad (3.23)$$

Here  $\mathbf{U}_{j+1}$  is the input vector, calculated for next iteration. Using equation (2.4) and

neglecting  $K_1$ , this input vector can be written as :-

$$\mathbf{U}_{j+1} = \mathbf{U} + \mathbf{K}\mathbf{E} \quad (3.24)$$

Expanding equation (3.23)  $\Rightarrow$

$$J(K) = \mathbf{Y}_d^T \mathbf{Y}_d - \mathbf{Y}_d^T \mathbf{G}\mathbf{U}_{j+1} - \mathbf{U}_{j+1}^T \mathbf{G}^T \mathbf{Y}_d + \mathbf{U}_{j+1}^T \mathbf{G}^T \mathbf{G}\mathbf{U}_{j+1} \quad (3.25)$$

As  $\mathbf{Y}_d^T \mathbf{G}\mathbf{U}_{j+1} = \mathbf{U}_{j+1}^T \mathbf{G}^T \mathbf{Y}_d$  and every term here is a scalar.

$$J(K) = \mathbf{Y}_d^T \mathbf{Y}_d - 2\mathbf{U}_{j+1}^T \mathbf{G}^T \mathbf{Y}_d + \mathbf{U}_{j+1}^T \mathbf{G}^T \mathbf{G}\mathbf{U}_{j+1} \quad (3.26)$$



Putting value of  $U_{j+1}$  from equation (3.24), equation (3.26)  $\Rightarrow$

$$J(K) = Y_d^T Y_d - 2(U + KE)^T G^T Y_d + (U + KE)^T G^T G (U + KE) \quad (3.27)$$

$$J(K) = Y_d^T Y_d - 2U^T G^T Y_d - 2K^T E^T G^T Y_d + U^T G^T G U + K^T E^T G^T G U + U^T G^T G E K + K^T E^T G^T G E K \quad (3.28)$$

As  $K^T E^T G^T G U = U^T G^T G E K$  and every term is a scalar.

$$J(K) = Y_d^T Y_d - 2U^T G^T Y_d - 2K^T E^T G^T Y_d + U^T G^T G U + 2K^T E^T G^T G U + K^T E^T G^T G E K \quad (3.29)$$

Plotting cost function  $J$  against different values of  $K$  gave the error performance curves for each system. This error performance curve for  $G_1(z)$  is shown in figure 3.3.

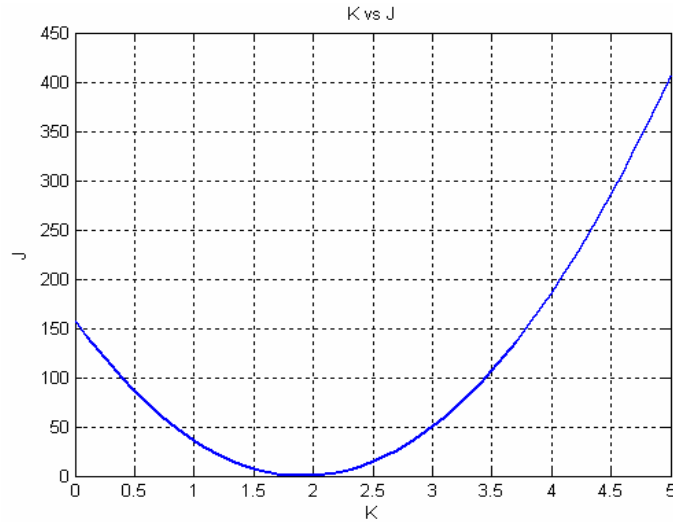


Figure 3.3: Cost function against different values of  $K$ .

The figure shows that the cost function has a minimum for certain values of  $K$ . Similar plots were obtained for other systems also.

Applying the  $\nabla$  operator to the cost function  $J(K)$

$$g = \nabla J(K) = -2E^T G^T Y_d + 2E^T G^T G U + 2K^T E^T G^T G E \quad (3.30)$$

$$\nabla J(\mathbf{K}) = -2\mathbf{E}^T \mathbf{G}^T (\mathbf{Y}_d - \mathbf{G}\mathbf{U} - \mathbf{K}^T \mathbf{G}\mathbf{E}) \quad (3.31)$$

$$\nabla J(\mathbf{K}) = -2\mathbf{E}^T \mathbf{G}^T (\mathbf{Y}_d - \mathbf{G}(\mathbf{U} + \mathbf{K}^T \mathbf{E})) \quad (3.32)$$

$$\nabla J(\mathbf{K}) = -2\mathbf{E}^T \mathbf{G}^T (\mathbf{Y}_d - \mathbf{G}\mathbf{U}_{j+1}) \quad (3.33)$$

Using steepest decent to find  $\mathbf{K}$  which results in minimum number of iterations.

$$\mathbf{K}_{j+1} = \mathbf{K}_j - \frac{1}{2} \mu (\nabla J(\mathbf{K})) \quad (3.34)$$

Or

$$\mathbf{K}_{j+1} = \mathbf{K}_j - \frac{1}{2} \mu \mathbf{g} \quad (3.35)$$

Where  $\mathbf{K}_j$  is the gain for current iteration,  $\mathbf{K}_{j+1}$  is the gain to be calculated for next iteration and  $\mu$  is the step size parameter. Putting value of  $\mathbf{g}$  from equation (3.33) in equation (3.35) we get

$$\mathbf{K}_{j+1} = \mathbf{K}_j - \frac{1}{2} \mu (-2\mathbf{E}^T \mathbf{G}^T (\mathbf{Y}_d - \mathbf{G}\mathbf{U}_{j+1})) \quad (3.36)$$

$$\mathbf{K}_{j+1} = \mathbf{K}_j + \mu \mathbf{E}^T \mathbf{G}^T (\mathbf{Y}_d - \mathbf{G}\mathbf{U}_{j+1}) \quad (3.37)$$

Using this learning law for gain, the proposed approach is presented in block diagram format in figure 3.4.

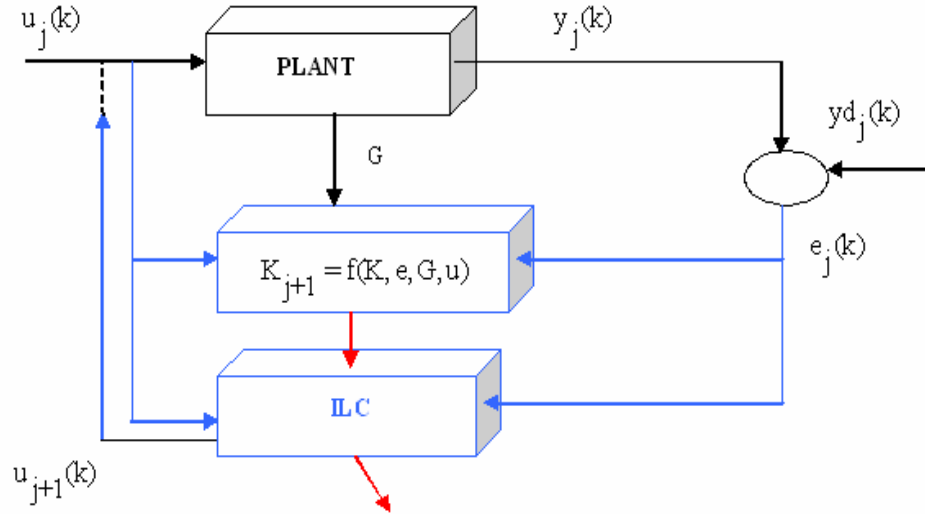


Figure 3.4: Block diagram of the scheme for Approach-1.

Input  $u_j(k)$  is applied to the plant. This input results in an output sequence  $y_j(k)$ . Using output and the desired output sequence  $y_d_j(k)$ , error  $e_j(k)$  is calculated. This error, the knowledge about previous values of  $K$ , the present input and the knowledge about the plant  $G$  is used to calculate the next value of  $K$  i.e.  $K_{j+1}$ . This value will be used during the next iteration by the ILC. The ILC calculates the next input to the plant,  $u_{j+1}(k)$ , which is applied to reduce error. This process continues until the desired output is achieved.

### 3.2.2 Simulation results

A number of simulations were performed to test this approach. The results of one such simulation using  $G_1(z)$  are discussed. The results were obtained with starting values of  $K = 0.1$  and  $\mu = 0.01$ . As discussed in chapter 2, there is always a range of values of  $K$  which converge with minimum number of iterations. They are called the optimal values of  $K$ . For this system, a plot of values of  $K$  against number of iterations it took to converge is given in figure 3.5 below.

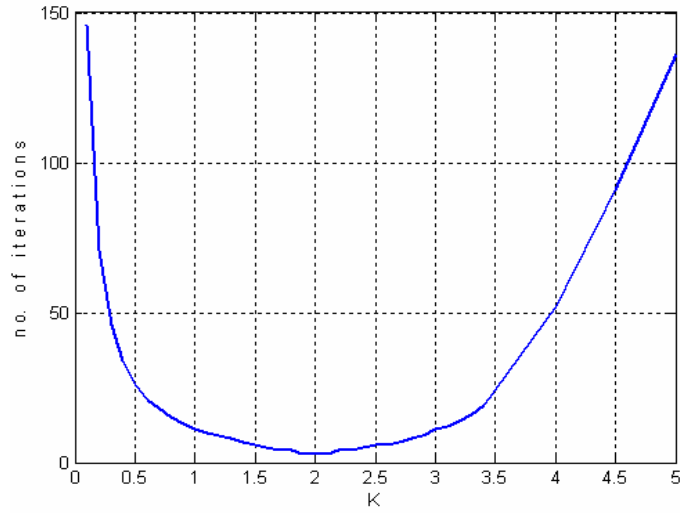


Figure 3.5: Number of iterations taken to converge for different values of  $K$ .

The optimal values of  $K$  are approximately between 1.7 – 2.3. Hence if equation (3.37) can find a value between this range, convergence in terms of finding value of  $K$  will be achieved.

The system,  $G_1(z)$ , in difference equation form can be written as

$$y(k+1) = 0.8187y(k) + 0.09063u(k) \quad (3.38)$$

Here  $a_1 = 0.8187$  and  $b_1 = 0.09063$

As this system has only  $a_1$  and  $b_1$ , equation (3.8) reduces to

$$y(k+1) = a_1y(k) + b_1u(k) \quad (3.39)$$

Output sequences generated for different values of  $k$  can be written as

$$y(1) = b_1u(0) \quad (3.40)$$

$$y(2) = a_1b_1u(0) + b_1u(1) \quad (3.41)$$

$$y(3) = a_1a_1b_1u(0) + a_1b_1u(1) + b_1u(2) \quad (3.42)$$

$$y(4) = a_1a_1a_1b_1u(0) + a_1a_1b_1u(1) + a_1b_1u(2) + b_1u(3) \quad (3.43)$$

$$y(5) = a_1 a_1 a_1 a_1 b_1 u(0) + a_1 a_1 a_1 b_1 u(1) + a_1 a_1 b_1 u(2) + a_1 b_1 u(3) + b_1 u(4) \quad (3.44)$$

⋮

or

$$\begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ y(4) \\ y(5) \\ \vdots \end{bmatrix} = \begin{bmatrix} b_1 & 0 & 0 & 0 & 0 & \dots \\ a_1 b_1 & b_1 & 0 & 0 & 0 & \dots \\ a_1 a_1 b_1 & a_1 b_1 & b_1 & 0 & 0 & \dots \\ a_1 a_1 a_1 b_1 & a_1 a_1 b_1 & a_1 b_1 & b_1 & 0 & \dots \\ a_1 a_1 a_1 a_1 b_1 & a_1 a_1 a_1 b_1 & a_1 a_1 b_1 & a_1 b_1 & b_1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ u(3) \\ u(4) \\ \vdots \end{bmatrix} \quad (3.45)$$

Putting the values of  $a_1$  and  $b_1$  in (3.45) gives

$$\begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ y(4) \\ y(5) \\ \vdots \end{bmatrix} = \begin{bmatrix} 0.09063 & 0 & 0 & 0 & 0 & \dots \\ 0.07404 & 0.09063 & 0 & 0 & 0 & \dots \\ 0.06049 & 0.07404 & 0.09063 & 0 & 0 & \dots \\ 0.04942 & 0.06049 & 0.07404 & 0.09063 & 0 & \dots \\ 0.04037 & 0.04942 & 0.06049 & 0.07404 & 0.09063 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ u(3) \\ u(4) \\ \vdots \end{bmatrix} \quad (3.46)$$

Where

$$\mathbf{G} = \begin{bmatrix} 0.09063 & 0 & 0 & 0 & 0 & \dots \\ 0.07404 & 0.09063 & 0 & 0 & 0 & \dots \\ 0.06049 & 0.07404 & 0.09063 & 0 & 0 & \dots \\ 0.04942 & 0.06049 & 0.07404 & 0.09063 & 0 & \dots \\ 0.04037 & 0.04942 & 0.06049 & 0.07404 & 0.09063 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (3.47)$$

Using this proposed approach, the output of the system, as number of iterations increase, is given in figure 3.6. The bold dotted lines show the desired response and the thin continuous lines show the actual response of the system.

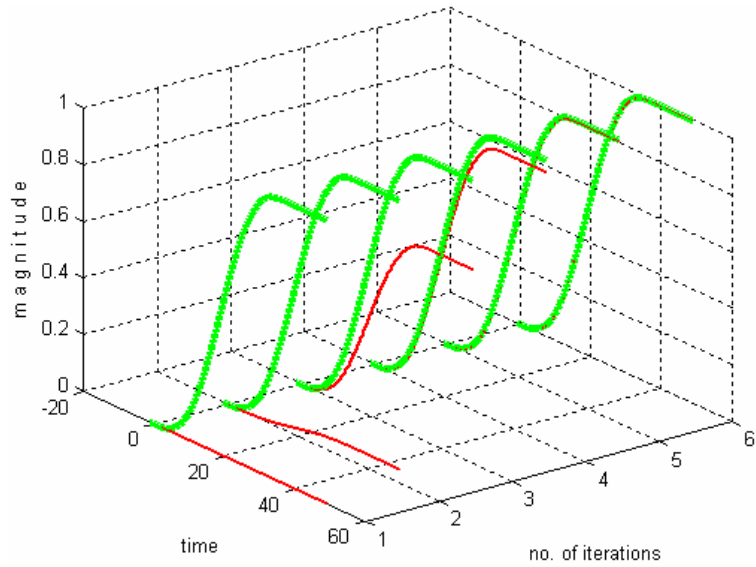


Figure 3.6: The proposed approach learning the desired output.

The norm of error as this learning was taking place is presented in figure 3.7.

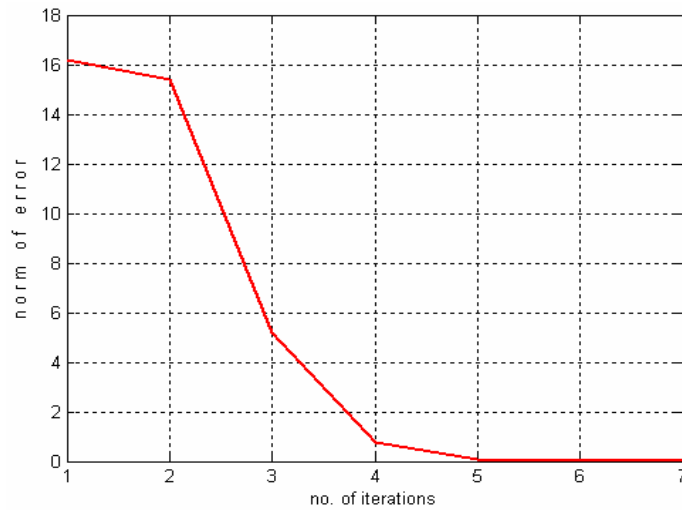


Figure 3.7: Norm of error as iterations increase.

The figure shows a rapid decrease in error as  $K$  is learnt iteratively using equation (3.37). The learning behaviour of  $K$  is shown in figure 3.8.

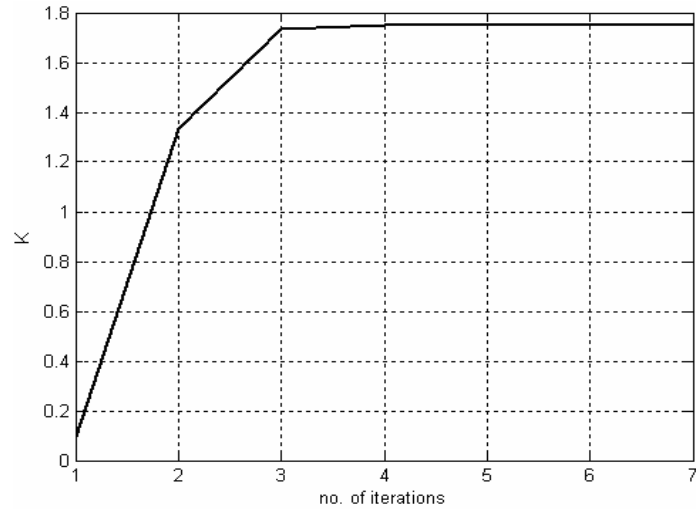


Figure 3.8: Learning behaviour of  $K$  .

This convergence behaviour was achieved even while the initial value of  $K$  was changed. For a starting value of  $K = 5$  the error response of the system as iterations increased is presented in figure 3.9.

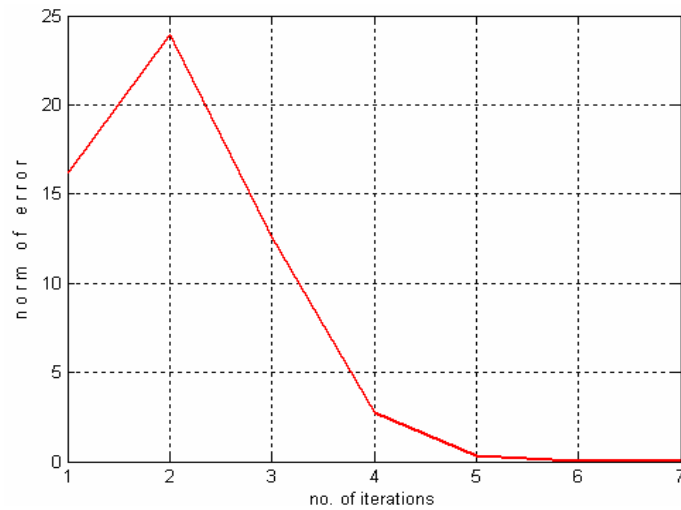


Figure 3.9: Error response with a starting value of,  $K = 5$  .

Figure 3.10 exhibits a plot of learnt values of  $K$  for different starting values of  $K$  .

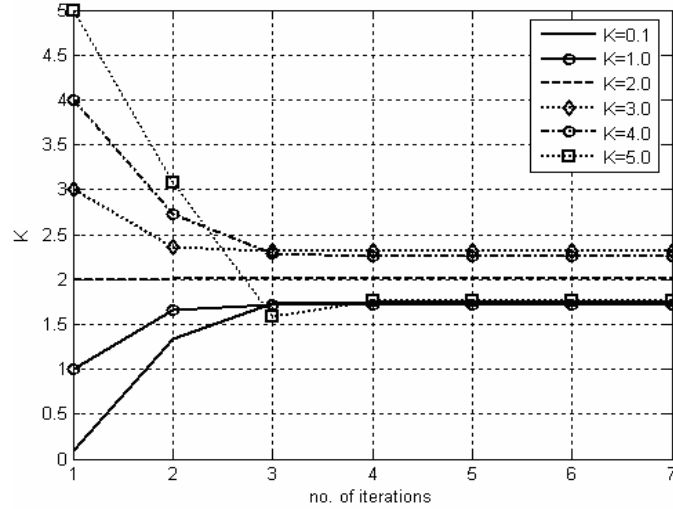


Figure 3.10: Learning behaviour of  $K$  for different initial values.

As can be seen in the figure,  $K$  always settles within the range 1.7 – 2.3, which is the optimal range for  $G_1(z)$ . For an initial value of  $K = 0.1$ ,  $K$  reaches 1.7528 at first run. This value is stored in memory and used in subsequent runs. After a few runs,  $K$  settles at 2.0193.

### 3.3 When System is Partially Known (Approach-2)

For this approach zero initial conditions are assumed i.e.  $u(k) = 0$  for  $k < 0$  and  $y(k) = 0$  for  $k < 1$ , with at least a single sample system delay.

Using equation (3.8), observations  $[y(1) \ y(2) \ \dots \ y(N)]$ , generated by applying an input sequence  $[u(0) \ u(1) \ \dots \ u(N-1)]$ , for the system  $G_1(z)$  can be written as

$$y(1) = -a_1 y(0) + b_1 u(0) \quad (3.48)$$

$$y(2) = -a_1 y(1) - a_2 y(0) + b_1 u(1) + b_2 u(0)$$



$$y(2) = -a_2y(0) - a_1y(1) + b_2u(0) + b_1u(1) \quad (3.49)$$

$$y(3) = -a_1y(2) - a_2y(1) - a_3y(0) + b_1u(2) + b_2u(1) + b_3u(0)$$

$$y(3) = -a_3y(0) - a_2y(1) - a_1y(2) + b_3u(0) + b_2u(1) + b_1u(2) \quad (3.50)$$

⋮

or

$$\begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ \vdots \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 & \dots \\ a_2 & a_1 & 0 & \dots \\ a_3 & a_2 & a_1 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} -y(0) \\ -y(1) \\ -y(2) \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1 & 0 & 0 & \dots \\ b_2 & b_1 & 0 & \dots \\ b_3 & b_2 & b_1 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \end{bmatrix} \quad (3.51)$$

Considering a single scalar gain in equation (2.4), equation (3.51) can be expanded to

$$\begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ \vdots \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 & \dots \\ a_2 & a_1 & 0 & \dots \\ a_3 & a_2 & a_1 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} -y(0) \\ -y(1) \\ -y(2) \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1 & 0 & 0 & \dots \\ b_2 & b_1 & 0 & \dots \\ b_3 & b_2 & b_1 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} u(-1) + Ke(-1) \\ u(0) + Ke(0) \\ u(1) + Ke(1) \\ \vdots \end{bmatrix} \quad (3.52)$$

$$\begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ \vdots \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 & \dots \\ a_2 & a_1 & 0 & \dots \\ a_3 & a_2 & a_1 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} -y(0) \\ -y(1) \\ -y(2) \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1 & 0 & 0 & \dots \\ b_2 & b_1 & 0 & \dots \\ b_3 & b_2 & b_1 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} u(-1) \\ u(0) \\ u(1) \\ \vdots \end{bmatrix} + K \begin{bmatrix} b_1 & 0 & 0 & \dots \\ b_2 & b_1 & 0 & \dots \\ b_3 & b_2 & b_1 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} e(-1) \\ e(0) \\ e(1) \\ \vdots \end{bmatrix} \quad (3.53)$$

This can be written in vector notation as

$$\mathbf{Y} = \mathbf{G}_1 \mathbf{X}_1 + \mathbf{G}_2 \mathbf{X}_2 + K \mathbf{G}_2 \mathbf{X}_3 \quad (3.54)$$

Where

$$\mathbf{G}_1 = \begin{bmatrix} a_1 & 0 & 0 & \dots \\ a_2 & a_1 & 0 & \dots \\ a_3 & a_2 & a_1 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (3.55)$$

$$\mathbf{G}_2 = \begin{bmatrix} b_1 & 0 & 0 & \dots \\ b_2 & b_1 & 0 & \dots \\ b_3 & b_2 & b_1 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (3.56)$$

$$\mathbf{X}_1^T = [-y(0) \quad -y(1) \quad \dots \quad -y(N-1)] \quad (3.57)$$

$$\mathbf{X}_2^T = [u(-1) \quad u(0) \quad \dots \quad u(N-2)] \quad (3.58)$$

$$\mathbf{X}_3^T = [e(-1) \quad e(0) \quad \dots \quad e(N-2)] \quad (3.59)$$

Putting (3.54) in (3.18) gives

$$\mathbf{E} = \mathbf{Y}_d - \mathbf{G}_1 \mathbf{X}_1 - \mathbf{G}_2 \mathbf{X}_2 - \mathbf{K} \mathbf{G}_2 \mathbf{X}_3 \quad (3.60)$$

### 3.3.1 Gradient descent for adaptive gain(s)

Taking cost function,  $J(\mathbf{K})$  as sum of error squared for current iteration, it can be written as

$$J(\mathbf{K}) = \sum_{k=1}^N e(k)^2 = \mathbf{E}^T \mathbf{E} \quad (3.61)$$

Putting the value of  $\mathbf{E}$  from equation (3.60) in (3.61)  $\Rightarrow$

$$J(\mathbf{K}) = (\mathbf{Y}_d - \mathbf{G}_1 \mathbf{X}_1 - \mathbf{G}_2 \mathbf{X}_2 - \mathbf{K} \mathbf{G}_2 \mathbf{X}_3)^T (\mathbf{Y}_d - \mathbf{G}_1 \mathbf{X}_1 - \mathbf{G}_2 \mathbf{X}_2 - \mathbf{K} \mathbf{G}_2 \mathbf{X}_3) \quad (3.62)$$

Expanding (3.62)  $\Rightarrow$

$$\begin{aligned} J(\mathbf{K}) = & \mathbf{Y}_d^T \mathbf{Y}_d - \mathbf{Y}_d^T \mathbf{G}_1 \mathbf{X}_1 - \mathbf{Y}_d^T \mathbf{G}_2 \mathbf{X}_2 - \mathbf{Y}_d^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} - \mathbf{X}_1^T \mathbf{G}_1^T \mathbf{Y}_d + \mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_1 \mathbf{X}_1 + \\ & \mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_2 \mathbf{X}_2 + \mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} - \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{Y}_d + \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{G}_1 \mathbf{X}_1 + \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_2 \\ & + \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} - \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{Y}_d + \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_1 \mathbf{X}_1 + \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_2 \\ & - \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} \end{aligned} \quad (3.63)$$

As  $\mathbf{Y}_d^T \mathbf{G}_1 \mathbf{X}_1 = \mathbf{X}_1^T \mathbf{G}_1^T \mathbf{Y}_d$ ,  $\mathbf{Y}_d^T \mathbf{G}_2 \mathbf{X}_2 = \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{Y}_d$ ,  $\mathbf{Y}_d^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} = \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{Y}_d$  and every term here is a scalar.

$$\begin{aligned} J(\mathbf{K}) = & \mathbf{Y}_d^T \mathbf{Y}_d - 2\mathbf{X}_1^T \mathbf{G}_1^T \mathbf{Y}_d - 2\mathbf{X}_2^T \mathbf{G}_2^T \mathbf{Y}_d - 2\mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{Y}_d + \mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_1 \mathbf{X}_1 \\ & + \mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_2 \mathbf{X}_2 + \mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} + \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{G}_1 \mathbf{X}_1 + \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_2 \\ & + \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} + \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_1 \mathbf{X}_1 + \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_2 + \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} \end{aligned} \quad (3.64)$$

As  $\mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} = \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_1 \mathbf{X}_1$  and every term here is a scalar.

$$\begin{aligned} J(\mathbf{K}) = & \mathbf{Y}_d^T \mathbf{Y}_d - 2\mathbf{X}_1^T \mathbf{G}_1^T \mathbf{Y}_d - 2\mathbf{X}_2^T \mathbf{G}_2^T \mathbf{Y}_d - 2\mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{Y}_d + \mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_1 \mathbf{X}_1 \\ & + \mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_2 \mathbf{X}_2 + 2\mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} + \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{G}_1 \mathbf{X}_1 + \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_2 \\ & + \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} + \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_2 + \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} \end{aligned} \quad (3.65)$$

As  $\mathbf{X}_2^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} = \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_2$  and every term here is a scalar.

$$\begin{aligned} J(\mathbf{K}) = & \mathbf{Y}_d^T \mathbf{Y}_d - 2\mathbf{X}_1^T \mathbf{G}_1^T \mathbf{Y}_d - 2\mathbf{X}_2^T \mathbf{G}_2^T \mathbf{Y}_d - 2\mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{Y}_d + \mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_1 \mathbf{X}_1 \\ & + \mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_2 \mathbf{X}_2 + 2\mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} + \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{G}_1 \mathbf{X}_1 + \mathbf{X}_2^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_2 \\ & + 2\mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_2 + \mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_3 \mathbf{K} \end{aligned} \quad (3.66)$$

Applying the  $\nabla$  operator to the cost function  $J(\mathbf{K})$

$$\begin{aligned} \mathbf{g} = \nabla J = & \frac{\partial J}{\partial \mathbf{K}} = -2\mathbf{X}_3^T \mathbf{G}_2^T \mathbf{Y}_d + 2\mathbf{X}_1^T \mathbf{G}_1^T \mathbf{G}_2 \mathbf{X}_3 + 2\mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_2 + \\ & 2\mathbf{K}^T \mathbf{X}_3^T \mathbf{G}_2^T \mathbf{G}_2 \mathbf{X}_3 \end{aligned} \quad (3.67)$$

$$= -2(\mathbf{Y}_d - \mathbf{X}_1^T \mathbf{G}_1^T - \mathbf{G}_2^T \mathbf{X}_2^T - \mathbf{K}^T \mathbf{G}_2^T \mathbf{X}_3^T) \mathbf{G}_2 \mathbf{X}_3$$

$$= -2\mathbf{E}^T \mathbf{G}_2 \mathbf{X}_3 \quad (3.68)$$

Putting this value of  $\mathbf{g}$  in (3.34) to find  $\mathbf{K}$  which results in minimum number of iterations, gives

$$\mathbf{K}_{j+1} = \mathbf{K}_j - \frac{1}{2} \mu (-2\mathbf{E}^T \mathbf{G}_2 \mathbf{X}_3)$$

$$K_{j+1} = K_j + \mu E^T G_2 X_3 \quad (3.69)$$

This adaptive mechanism requires only partial knowledge of the system. It requires only the knowledge about  $b$  coefficients. The approach is exhibited in block diagram in figure 3.11.

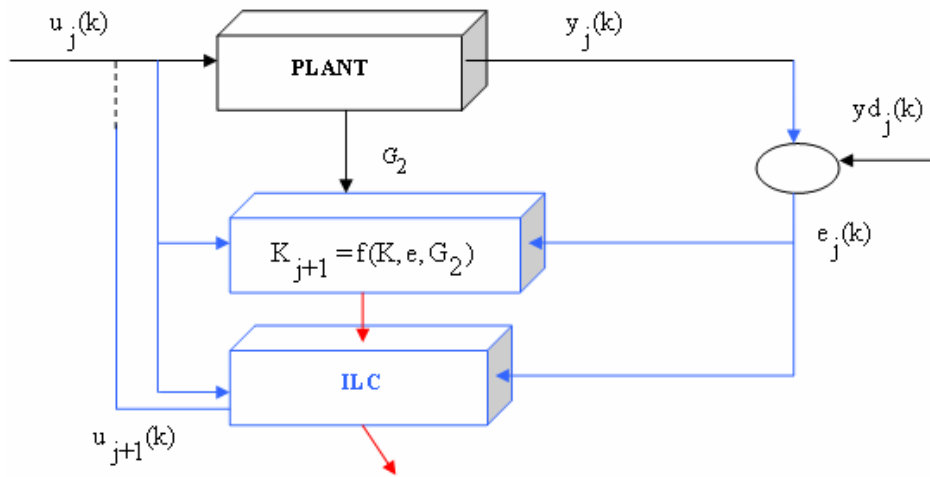


Figure 3.11: Block diagram for Approach-2.

Input  $u_j(k)$  is applied to the plant. This input results in an output sequence  $y_j(k)$ . Using the desired output sequence  $y_d_j(k)$ , error  $e_j(k)$  is calculated. This error, the knowledge about previous values of  $K$  and the partial knowledge about the plant  $G_2$  is used to calculate the next value of  $K$  i.e.  $K_{j+1}$ . This value will be used during the next iteration by the ILC. The ILC calculates the next input to the plant  $u_{j+1}(k)$  which is applied to reduce error. This process continues until the desired out is achieved.

### 3.3.2 Simulation results

Many systems were simulated to test this approach. For comparison with previous approach, simulation results from using system  $G_1(z)$  are presented in this section. Using

the difference equation representation given in (3.38), the output sequences generated for different values of  $k$  can be written as

$$y(1) = 0.8187y(0) + 0.09063u(0) \quad (3.70)$$

$$y(2) = 0.8187y(1) + 0.09063u(1) \quad (3.71)$$

$$y(3) = 0.8187y(2) + 0.09063u(2) \quad (3.72)$$

⋮

or

$$\begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ \vdots \end{bmatrix} = \begin{bmatrix} 0.8187 & 0 & 0 & \dots \\ 0 & 0.8187 & 0 & \dots \\ 0 & 0 & 0.8187 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \end{bmatrix} + \begin{bmatrix} 0.09063 & 0 & 0 & \dots \\ 0 & 0.09063 & 0 & \dots \\ 0 & 0 & 0.09063 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \end{bmatrix} \quad (3.73)$$

$$\begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ \vdots \end{bmatrix} = \begin{bmatrix} 0.8187 & 0 & 0 & \dots \\ 0 & 0.8187 & 0 & \dots \\ 0 & 0 & 0.8187 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ M \end{bmatrix} + \begin{bmatrix} 0.09063 & 0 & 0 & \dots \\ 0 & 0.09063 & 0 & \dots \\ 0 & 0 & 0.09063 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} u(-1) \\ u(0) \\ u(1) \\ \vdots \end{bmatrix} \quad (3.74)$$

$$+ K \begin{bmatrix} 0.09063 & 0 & 0 & \dots \\ 0 & 0.09063 & 0 & \dots \\ 0 & 0 & 0.09063 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} e(-1) \\ e(0) \\ e(1) \\ \vdots \end{bmatrix}$$

Here

$$\mathbf{G}_2 = \begin{bmatrix} 0.09063 & 0 & 0 & \dots \\ 0 & 0.09063 & 0 & \dots \\ 0 & 0 & 0.09063 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (3.75)$$

$$\mathbf{X}_3 = \begin{bmatrix} e(-1) \\ e(0) \\ e(1) \\ \vdots \end{bmatrix} \quad (3.76)$$

$$\mathbf{E} = \begin{bmatrix} e(1) \\ e(2) \\ e(3) \\ \vdots \end{bmatrix} \quad (3.77)$$

Figure 3.12 shows the output of the system using this approach against the desired output as number of iterations increase. The bold dotted lines show the desired response and the thin continuous lines show the actual response of the system.

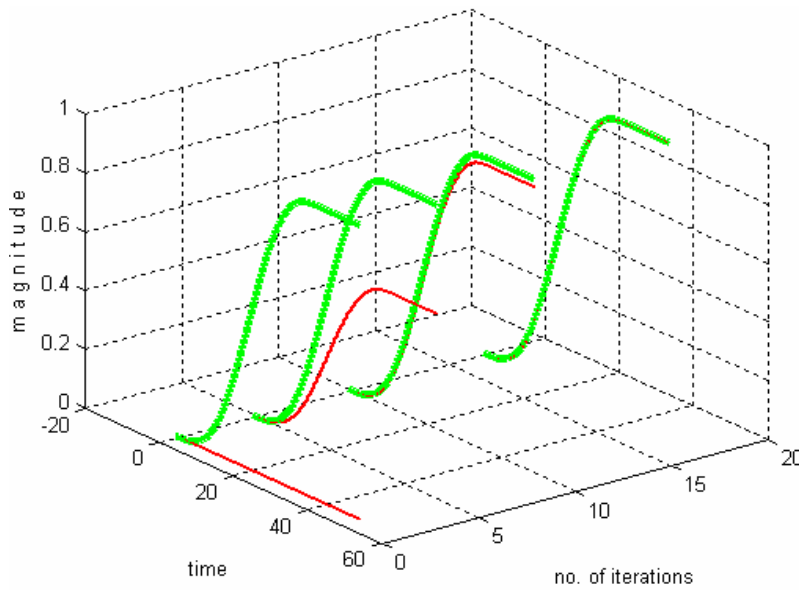


Figure 3.12: The proposed approach learning the desired output.

The output reaches the desired output in 17 iterations. During those iterations the error is reduced as exhibited in figure 3.13.

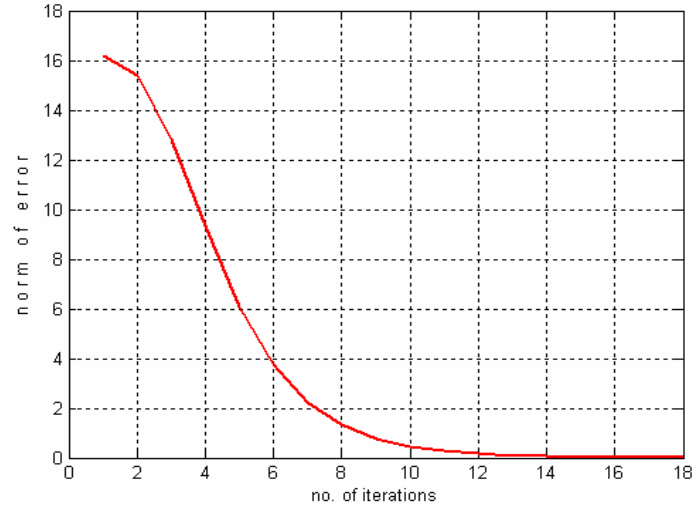


Figure 3.13: Norm of error as iterations increase.

During this run  $K$  is updated using (3.69). The learnt values of  $K$  for this run are presented in figure 3.14.

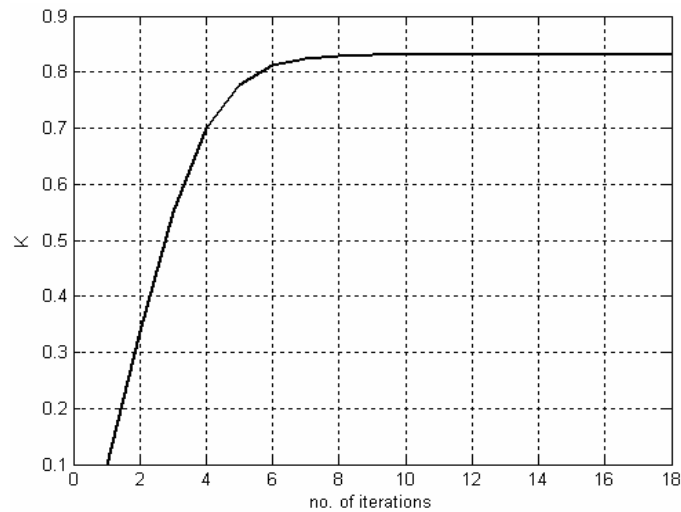


Figure 3.14: Learning behaviour of  $K$  at first run.

This value is further updated in subsequent runs until it reaches the optimal range. Though this approach takes more iterations than the previous approach, it requires only partial knowledge about the plant.

### 3.4 When System is Completely Unknown (Approach-3)

Learning controllers are well known to converge slowly. Simulation results from linear and non linear systems have shown that fixed values of gain matrices are one of the reasons for the slow convergence rate. They clearly need to be adaptive. Also, in most cases the complete knowledge of the system is not available. An approach that does not depend on the knowledge of the system was one of the aims of our research.

This section suggests a novel scheme shown in Figure 3.15. It is assumed that the plant (system) is completely unknown and hence its parameters are estimated to assist in the ILC scheme. For a detailed description on System Identification techniques see reference [96].

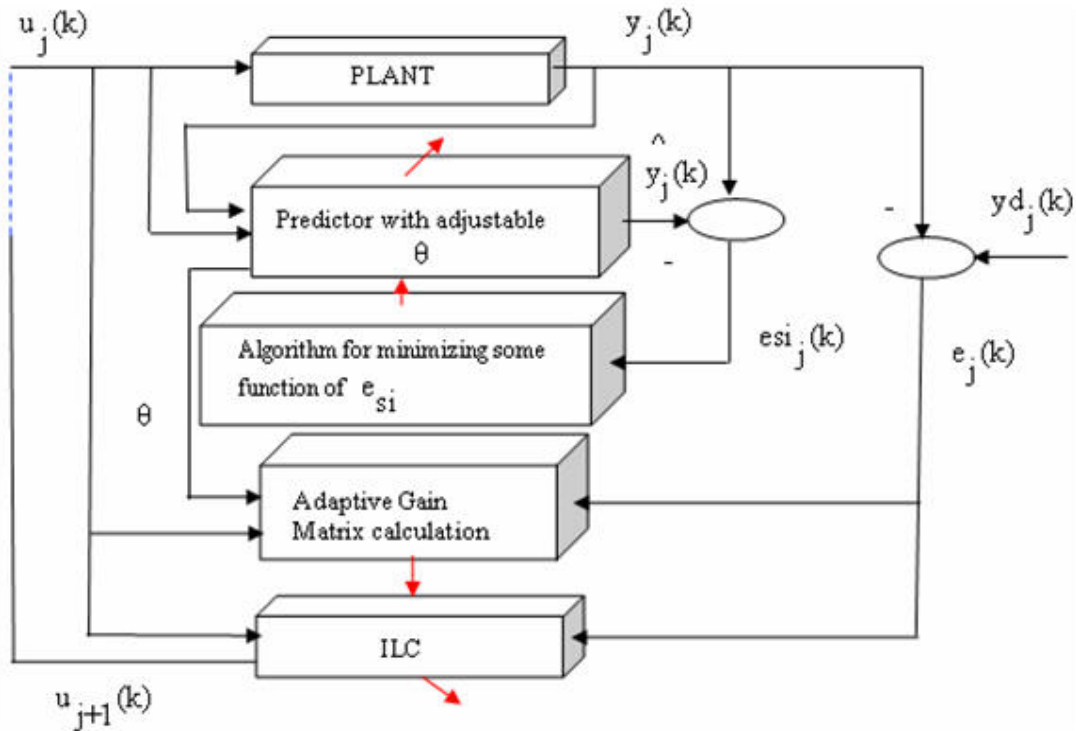


Figure 3.15: Block diagram representation of the proposed scheme.



Input  $u_j(k)$  is applied to the plant. This input produces an output  $y_j(k)$ . With the knowledge about the desired output  $y_{d_j}(k)$ , the error  $e_j(k)$  is calculated. This error is used by the adaptive gain matrix calculation block to adjust gain matrices and by the ILC block to calculate next input to the plant,  $u_{j+1}(k)$ . Predictor with adjustable  $\theta$  block produces the predicted output  $\hat{y}_j(k)$ , which is used to generate system identification error  $esi_j(k)$ . This error is then used to iteratively identify the unknown plant parameters  $\theta$ . Adaptive gain matrix calculation block uses this information, about the plant, to readjust  $K$ .

Equation (3.8), with disturbances, can be written in 2-D format as

$$y_j(k) = -a_1 y_j(k-1) - a_2 y_j(k-2) - \dots - a_{n_a} y_j(k-n_a) + b_1 u_j(k-1) + \dots + b_{n_b} u_j(k-n_b) + \varepsilon_j(k) \quad (3.78)$$

Here  $\varepsilon_j(k)$  is the white noise and  $a_{n_a}$  and  $b_{n_b}$  are the orders of the respective polynomials. Observations  $[y_j(1) \ y_j(2) \ \dots \ y_j(N)]$  generated by applying an input sequence  $[u_j(0) \ u_j(1) \ \dots \ u_j(N-1)]$ , using (3.78) are as follows

$$y_j(1) = b_1 u_j(0) \quad (3.79)$$

$$y_j(2) = (-a_1 b_1 + b_2) u_j(0) + b_1 u_j(1) \quad (3.80)$$

$$y_j(3) = (-a_1 a_1 b_1 - a_1 b_2 - a_2 b_1 + b_3) u_j(0) + (-a_1 b_1 + b_2) u_j(1) + b_1 u_j(2) \quad (3.81)$$

⋮

Two vectors called the input and output vectors and a matrix called the  $\mathbf{G}$  matrix are defined as

$$\mathbf{Y}^T = [y_j(1) \quad y_j(2) \quad \dots \quad y_j(N)] \quad (3.82)$$

$$\mathbf{U}^T = [u_j(0) \quad u_j(1) \quad \dots \quad u_j(N-1)] \quad (3.83)$$

$$\mathbf{G} = \begin{bmatrix} g(1) & 0 & \dots & 0 \\ g(2) & g(1) & 0 & 0 \\ \vdots & \vdots & \ddots & 0 \\ g(N) & g(N-1) & & g(1) \end{bmatrix} \quad (3.84)$$

The  $\mathbf{G}$  matrix, which is some times called the design matrix, is lower triangular and for linear systems it is also Toeplitz [106]. The values  $g(1)$ ,  $g(2)$ , ...,  $g(N)$  are the impulse response coefficients of  $\mathbf{G}$ .

Using this 2-D representation, the system can be written in vector notation form as

$$\mathbf{Y} = \mathbf{GU} + \boldsymbol{\varepsilon}_j \quad (3.85)$$

or for next iteration as

$$\mathbf{Y}_{j+1} = \mathbf{GU}_{j+1} + \boldsymbol{\varepsilon}_{j+1} \quad (3.86)$$

Here  $\boldsymbol{\varepsilon}_j$  is the white noise vector and is defined as

$$\boldsymbol{\varepsilon}_j^T = [\varepsilon_j(1), \varepsilon_j(2), \dots, \varepsilon_j(N)]$$

### 3.4.1 Identification

Identification of a system [134] is computationally heavy. It is suggested to identify the system once in the beginning and after words only if required. The system equation (3.78) in 2-D representation can also be expressed in the following form.

$$\mathbf{Y}_j = \mathbf{x}_j^T(k)\boldsymbol{\theta} + \boldsymbol{\varepsilon}_j(k) \quad (3.87)$$

Here  $\mathbf{x}_j(k)$  is a vector of past observations for  $k = [1, 2, \dots, N]$ . For the first learning cycle (3.87) is re written as

$$\mathbf{Y}_1 = \mathbf{x}_1^T(k) \boldsymbol{\theta} + \boldsymbol{\varepsilon}_1(k) \quad (3.88)$$

Here  $\mathbf{x}_1(k)$  is a vector of past observations at first learning cycle and is given by

$$\mathbf{x}_1^T(k) = \begin{bmatrix} -y_1(1) & \dots & -y_1(N) \\ -y_1(2) & \dots & -y_1(N+1) \\ \vdots & \vdots & \vdots \\ -y_1(na) & \dots & -y_1(N+na-1) \\ u_1(1) & \dots & u_1(N) \\ u_1(2) & \dots & u_1(N+1) \\ \vdots & \vdots & \vdots \\ u_1(nb) & \dots & u_1(N+nb-1) \end{bmatrix} \quad (3.89)$$

The dimensions of  $\mathbf{x}_1(k)$  depend on the number of parameters to be identified. For example if only  $a_1$  and  $b_1$  are identified, equation (3.89) reduces to

$$\mathbf{x}_1^T(k) = \begin{bmatrix} -y_1(1) & \dots & -y_1(N) \\ u_1(1) & \dots & u_1(N) \end{bmatrix} \quad (3.90)$$

Theta,  $\boldsymbol{\theta}$  in equation (3.88) is a vector of unknown system parameters and is defined as

$$\boldsymbol{\theta}^T = [a_1 \ a_2 \ \dots \ a_{na} \ b_1 \ b_2 \ \dots \ b_{nb}] \quad (3.91)$$

Using (3.88) for the first learning cycle and ignoring noise,  $\{\boldsymbol{\varepsilon}_j(k)\}$ , gives us the predicted output as

$$\hat{\mathbf{Y}}_1 = \mathbf{x}_1^T(k) \boldsymbol{\theta} \quad (3.92)$$

Here  $\hat{\mathbf{Y}}_1$  is the predicted output for the first iteration. According to figure 3.15, this gives rise to a residual, the system identification error, for first learning cycle, which is given by

$$esi_1(k) = y_1(k) - \hat{y}_1(k) \quad (3.93)$$

Here  $esi$  denotes residual error in the identification process. This in vector notation form can be written as

$$\mathbf{Esi}_1(k) = \mathbf{Y}_1 - \hat{\mathbf{Y}}_1 \quad (3.94)$$

Putting (3.92) in (3.94)  $\Rightarrow$

$$\mathbf{Esi}_1(k) = \mathbf{Y}_1 - \mathbf{x}_1^T(k)\boldsymbol{\theta} \quad (3.95)$$

Where

$$\mathbf{Esi}_1^T(k) = [esi_1(1) \quad esi_1(2) \quad \dots \quad esi_1(N)] \quad (3.96)$$

and

$$\mathbf{Y}_1^T = [y_1(1) \quad y_1(2) \quad \dots \quad y_1(N)] \quad (3.97)$$

So, observations  $\{y_1(1), y_1(2), \dots, y_1(N)\}$  have been generated by applying an input sequence  $\{u_1(1), u_1(2), \dots, u_1(N)\}$  which gives rise to a sequence of residuals  $\{esi_1(1), esi_1(2), \dots, esi_1(N)\}$ . For the first learning cycle, equation (3.95) for different values of  $k$ , gives.

$$esi_1(1) = Y_1(1) - \mathbf{x}_1^T(1)\boldsymbol{\theta} \quad (3.98)$$

$$esi_1(2) = Y_1(2) - \mathbf{x}_1^T(2)\boldsymbol{\theta} \quad (3.99)$$

...

$$esi_1(N) = Y_1(N) - \mathbf{x}_1^T(N)\boldsymbol{\theta} \quad (3.100)$$

To identify  $\boldsymbol{\theta}$  we define the cost function  $J(\boldsymbol{\theta})$  as the sum of error square.

$$J(\boldsymbol{\theta}) = \sum_{k=1}^N esi_1(k)^2 = \mathbf{Esi}_1^T(k)\mathbf{Esi}_1(k) \quad (3.101)$$

Using equation (3.97) in (3.101) and expanding  $\Rightarrow$

$$J(\boldsymbol{\theta}) = \mathbf{Y}_1^T(k)\mathbf{Y}_1(k) - \mathbf{Y}_1^T(k)\mathbf{x}_1(k)\boldsymbol{\theta} - \boldsymbol{\theta}^T\mathbf{x}_1^T(k)\mathbf{Y}_1(k) + \boldsymbol{\theta}^T\mathbf{x}_1(k)\mathbf{x}_1^T(k)\boldsymbol{\theta} \quad (3.102)$$

As  $\mathbf{Y}_1^T(k)\mathbf{x}_1(k)\boldsymbol{\theta} = \boldsymbol{\theta}^T\mathbf{x}_1^T(k)\mathbf{Y}_1(k)$  and every term here is a scalar

$$J(\boldsymbol{\theta}) = \mathbf{Y}_1^T(k)\mathbf{Y}_1(k) - 2\boldsymbol{\theta}^T\mathbf{x}_1^T(k)\mathbf{Y}_1(k) + \boldsymbol{\theta}^T\mathbf{x}_1(k)\mathbf{x}_1^T(k)\boldsymbol{\theta} \quad (3.103)$$

Differentiating with respect to  $\boldsymbol{\theta}$ , equation (3.103) gives

$$\frac{\partial J}{\partial \boldsymbol{\theta}} = 2\left[\mathbf{x}_1(k)\mathbf{x}_1^T(k)\right]\boldsymbol{\theta} - 2\left[\mathbf{x}_1^T(k)\mathbf{Y}_1(k)\right] \quad (3.104)$$

For minimum value  $\frac{\partial J}{\partial \boldsymbol{\theta}} = 0$ . Which gives

$$\hat{\boldsymbol{\theta}} = \left[\mathbf{x}_1(k)\mathbf{x}_1^T(k)\right]^{-1} \mathbf{x}_1^T(k)\mathbf{Y}_1(k) \quad (3.105)$$

Equation (3.105) gives us the power to estimate  $a_1, b_1$  using  $\{u_1(1), u_1(2), \dots, u_1(N)\}$  and  $\{y_1(1), y_1(2), \dots, y_1(N)\}$  sequences. If the input signal is not rich enough to excite all modes of the system for identification purposes, this process can be done separately with a sequence of pseudo random signal.

### 3.4.2 Gradient descent for adaptive gain(s)

Defining an error vector  $\mathbf{E}_j$  which is an  $N \times 1$  vector for current iteration as

$$\mathbf{E}_j^T = \left[ e_j(1) \quad e_j(2) \quad \dots \quad e_j(N) \right] \quad (3.106)$$

And an error vector for next iteration as

$$\mathbf{E}_{j+1}^T = \left[ e_{j+1}(1) \quad e_{j+1}(2) \quad \dots \quad e_{j+1}(N) \right] \quad (3.107)$$

The aim is to eliminate or reduce error in the next iteration or next learning cycle.

$$\lim_{j \rightarrow \infty} \left\| \mathbf{E}_{j+1}^T \right\| \rightarrow 0 \quad (3.108)$$

To achieve this reduction in error for next learning cycle, cost function  $J(K)$  as sum of error square for next learning cycle is defined as

$$J(K) = \sum_{k=1}^N e_{j+1}(k)^2 = \mathbf{E}_{j+1}^T \mathbf{E}_{j+1} \quad (3.109)$$

Defining output vector  $\mathbf{Y}_{j+1}$  for the next iteration and desired output vector  $\mathbf{Yd}_j$  as

$$\mathbf{Y}_{j+1}^T = \begin{bmatrix} y_{j+1}(1) & y_{j+1}(2) & \dots & y_{j+1}(N) \end{bmatrix} \quad (3.110)$$

and

$$\mathbf{Yd}_j^T = \begin{bmatrix} yd_j(1) & yd_j(2) & \dots & yd_j(N) \end{bmatrix} \quad (3.111)$$

For the following derivation,  $\mathbf{Yd}_j(k) = \mathbf{Yd}$ , for simplicity in mathematical representation.

Using the definition of error in equation (3.22), the error for next iteration can be written as:

$$\mathbf{E}_{j+1} = \mathbf{Yd} - \mathbf{Y}_{j+1} \quad (3.112)$$

Substituting this value in equation (3.109)  $\Rightarrow$

$$J(K) = (\mathbf{Yd} - \mathbf{Y}_{j+1})^T (\mathbf{Yd} - \mathbf{Y}_{j+1}) \quad (3.113)$$

Expanding (3.113)  $\Rightarrow$

$$J(K) = \mathbf{Yd}^T \mathbf{Yd} - \mathbf{Yd}^T \mathbf{G} \mathbf{U}_{j+1} - \mathbf{U}_{j+1}^T \mathbf{G}^T \mathbf{Yd} + \mathbf{U}_{j+1}^T \mathbf{G}^T \mathbf{G} \mathbf{U}_{j+1} \quad (3.114)$$

As  $\mathbf{Yd}^T \mathbf{G} \mathbf{U}_{j+1} = \mathbf{U}_{j+1}^T \mathbf{G}^T \mathbf{Yd}$ , where every term is a scalar. Equation (3.114) reduces to

$$J(K) = \mathbf{Yd}^T \mathbf{Yd} - 2\mathbf{U}_{j+1}^T \mathbf{G}^T \mathbf{Yd} + \mathbf{U}_{j+1}^T \mathbf{G}^T \mathbf{G} \mathbf{U}_{j+1} \quad (3.115)$$

For the following discussion,  $K$  will be taken as scalar quantity i.e. one value of gain for all the samples, as discussed in chapter 2. This  $K$  will also be written as  $K_j$ , in some equations, to show change from iteration to iteration. Despite the duplicate usage in nomenclature, for all practical purposes  $K = K_j$ .

Equation (2.4) can be written as:-

$$u_{j+1}(k) = u_j(k) + K_j(k)e_j(k) \quad (3.116)$$

This in vector notation form is

$$\mathbf{U}_{j+1} = \mathbf{U}_j + K_j \mathbf{E}_j \quad (3.117)$$

Where

$$\mathbf{U}_{j+1}^T = \begin{bmatrix} u_{j+1}(1) & u_{j+1}(2) & \dots & u_{j+1}(N) \end{bmatrix} \quad (3.118)$$

and  $K_j$  is a scalar value of gain at iteration  $j$ . Putting (3.117) in (3.115)  $\Rightarrow$

$$J(K) = \mathbf{Yd}^T \mathbf{Yd} - 2(\mathbf{U}_j + K_j \mathbf{E}_j)^T \mathbf{G}^T \mathbf{Yd} + (\mathbf{U}_j + K_j \mathbf{E}_j)^T \mathbf{G}^T \mathbf{G} (\mathbf{U}_j + K_j \mathbf{E}_j) \quad (3.119)$$

$$\begin{aligned} J(K) = \mathbf{Yd}^T \mathbf{Yd} - 2\mathbf{U}_j^T \mathbf{G}^T \mathbf{Yd} - 2K_j \mathbf{E}_j^T \mathbf{G}^T \mathbf{Yd} + \mathbf{U}_j^T \mathbf{G}^T \mathbf{G} \mathbf{U}_j + K_j \mathbf{E}_j^T \mathbf{G}^T \mathbf{G} \mathbf{U}_j \\ + \mathbf{U}_j^T \mathbf{G}^T \mathbf{G} \mathbf{E}_j K_j + K_j \mathbf{E}_j^T \mathbf{G}^T \mathbf{G} \mathbf{E}_j K_j \end{aligned} \quad (3.120)$$

As  $K_j \mathbf{E}_j^T \mathbf{G}^T \mathbf{G} \mathbf{U}_j = \mathbf{U}_j^T \mathbf{G}^T \mathbf{G} \mathbf{E}_j K_j$  and every term is a scalar

$$\begin{aligned} J(K) = \mathbf{Yd}^T \mathbf{Yd} - 2\mathbf{U}_j^T \mathbf{G}^T \mathbf{Yd} - 2K_j \mathbf{E}_j^T \mathbf{G}^T \mathbf{Yd} + \mathbf{U}_j^T \mathbf{G}^T \mathbf{G} \mathbf{U}_j + 2K_j \mathbf{E}_j^T \mathbf{G}^T \mathbf{G} \mathbf{U}_j + \\ K_j \mathbf{E}_j^T \mathbf{G}^T \mathbf{G} \mathbf{E}_j K_j \end{aligned} \quad (3.121)$$

Applying the  $\nabla$  operator to the cost function  $J(K)$

$$g = \nabla J(K) = -2\mathbf{E}_j^T \mathbf{G}^T \mathbf{Yd} + 2\mathbf{E}_j^T \mathbf{G}^T \mathbf{G} \mathbf{U}_j + 2K_j \mathbf{E}_j^T \mathbf{G}^T \mathbf{G} \mathbf{E}_j \quad (3.122)$$

Ignoring error square terms  $\Rightarrow$

$$\nabla J(K) = -2\mathbf{E}_j^T \mathbf{G}^T (\mathbf{Yd} - \mathbf{G} \mathbf{U}_{j+1}) \quad (3.123)$$

To find the value of  $K_j$  which will give us minimum number of iterations we use the steepest descent given in (3.34). Putting (3.124) in (3.34)  $\Rightarrow$

$$K_{j+1} = K_j + \mu \mathbf{E}_j^T \mathbf{G}^T (\mathbf{Yd} - \mathbf{G} \mathbf{U}_{j+1}) \quad (3.124)$$

Here  $\mu$  is the step size parameter. It should be noted here that equation (3.117) should be calculated before equation (3.124) can be executed.

### 3.4.3 Convergence analysis

By convergence we mean that the system output  $y_j(k)$  approaches desired output  $y_d(k)$  as the learning process continues i.e.

$$y_j(k) \rightarrow y_d(k) \quad \text{for } k \in [0, N] \text{ as } j \rightarrow \infty$$

or

$$e_j(k) \rightarrow 0 \quad \text{for } k \in [0, N] \text{ as } j \rightarrow \infty$$

First the convergence criterion of the learning control law is established followed by the convergence of the adaptive gain law.

#### 3.4.3.1 Convergence of iterative learning control law

Using the control scheme in equation (3.117) the system description equation (3.86) can be expanded to

$$\mathbf{Y}_{j+1} = \mathbf{G}(\mathbf{U}_j + \mathbf{K}_j \mathbf{E}_j) \quad (3.125)$$

$$\mathbf{Y}_{j+1} = \mathbf{G}\mathbf{U}_j + \mathbf{K}_j \mathbf{G}\mathbf{E}_j \quad (3.126)$$

Multiplying both sides with -1 and adding  $\mathbf{Yd}$  we get

$$\mathbf{Yd} - \mathbf{Y}_{j+1} = \mathbf{Yd} - \mathbf{G}\mathbf{U}_j - \mathbf{K}_j \mathbf{G}\mathbf{E}_j \quad (3.127)$$

Using 2-D system representation in (3.85) and ignoring noise

$$\mathbf{Yd} - \mathbf{Y}_{j+1} = \mathbf{Yd} - \mathbf{Y}_j - \mathbf{K}_j \mathbf{G}\mathbf{E}_j \quad (3.128)$$

Using error equation in (3.22)

$$\mathbf{E}_{j+1} = \mathbf{Yd} - \mathbf{Y}_j - \mathbf{K}_j \mathbf{G}\mathbf{E}_j \quad (3.129)$$

This can be written as



$$\mathbf{E}_{j+1} = \mathbf{E}_j - \mathbf{K}_j \mathbf{G} \mathbf{E}_j \quad (3.130)$$

$$\mathbf{E}_{j+1} = (\mathbf{I} - \mathbf{K}_j \mathbf{G}) \mathbf{E}_j \quad (3.131)$$

Based on equation (3.131) it can be shown that if the value  $\mathbf{K}_j$  is selected such that

$\|\mathbf{I} - \mathbf{K}_j \mathbf{G}\| < 1$  then the error will decrease continuously and the system will converge at some iteration.

### 3.4.3.2 Convergence for adaptive gain

Adaptive gain  $\mathbf{K}_j$  is calculated using the technique of steepest descent. Our adaptive iterative learning controller and the steepest descent method combine well as both are iterative techniques. In our case  $\mathbf{K}_j = \mathbf{K}_j(k)$ , as the gain is same for all values of  $k$  for a particular  $j$ .

We consider a cost function  $J(\mathbf{K}_j(k))$  which is continuously differentiable function of some unknown gain  $\mathbf{K}_j(k)$ . We want to find an optimal solution  $\mathbf{K}_{op_j}(k)$  that satisfies the condition

$$J(\mathbf{K}_{op_j}(k)) \leq J(\mathbf{K}_j(k)) \quad \text{for all values of } \mathbf{K}_j \quad (3.132)$$

This is a mathematical statement of unconstrained optimization. A class of unconstrained optimization algorithms that is found to be well suited in ILC is based on the idea of local iterative descent.

Starting with an initial guess  $\mathbf{K}_{i_j}(k)$  we have to generate subsequent gains such that

$$J(\mathbf{K}_{j+1}(k)) < J(\mathbf{K}_j(k)) \quad (3.133)$$

where  $J(\mathbf{K}_j(k))$  is the old value of the gain and  $J(\mathbf{K}_{j+1}(k))$  is the updated value.

It is anticipated that the algorithm will eventually converge on to the optimal value  $K_{op_j}(k)$ . In the simple form of iterative descent known as the method of steepest descent, the successive adjustments applied to the gain  $K_j(k)$  are in the direction of steepest descent that is in a direction opposite to the gradient vector of the cost function  $J(K_j(k))$  which is denoted by  $\Delta J(K_j(k))$ . For convenience of presentation we write

$$g = \Delta J(K_j(k)) \quad (3.134)$$

According to the steepest descent, the suggested algorithm is formally described by equation (3.34) and is re-written as

$$K_{j+1}(k) = K_j(k) - \frac{1}{2}\mu(\Delta J(K_j(k))) \quad (3.135)$$

Where  $j$  denotes the iterations,  $\mu$  is a factor represents the step size parameter and  $\frac{1}{2}$  term is introduced for mathematical convenience.

In going from iteration  $j$  to  $j+1$  the algorithm applies the gain adjustment

$$\delta K_j(k) = K_{j+1}(k) - K_j(k) \quad (3.136)$$

Using (3.135) in (3.136)

$$\delta K_j(k) = -\frac{1}{2}\mu(\Delta J(K_j(k))) \quad (3.137)$$

To show that formulation of steepest descent algorithm satisfies the condition in (3.132), Taylor series expansion around  $K_j(k)$  can be used to obtain approximation

$$J(K_{j+1}(k)) \cong J(K_j(k)) + \Delta J(K_j(k))\delta K_j(k) \quad (3.138)$$

Substituting (3.137) in (3.138) we get

$$J(K_{j+1}(k)) \cong J(K_j(k)) - \frac{1}{2}\mu \|\Delta J(K_j(k))\|^2 \quad (3.139)$$

Which shows that  $J(K_{j+1}(k))$  is smaller than  $J(K_j(k))$  provided that the step size parameter  $\mu$  is positive. Hence, it follows that with increasing  $j$  the cost function  $J(K_j(k))$  progressively decreases, approaching the minimum value  $J_{\min}$  in terms of

error at  $j \rightarrow \infty$ . This shows that we are always moving towards  $Kop_j(k)$ , after each iteration.

### 3.4.4 Simulation results

The scheme presented in this section was also tested through simulations. Results from three systems, described in Appendix A, are presented in this section. For this section the system is identified using  $a_1$  and  $b_1$  only, as it was observed that these two parameters had the maximum impact on the results. More parameters can be identified, if required.

#### 3.4.4.1 A Simple System

The system  $G_1(z)$  is written in difference equation form as

$$y(k+1) = 0.8187y(k) + 0.09063u(k) \quad (3.140)$$

With  $a_1 = 0.8187$ ,  $b_1 = 0.09063$  and

$$g(n) = \begin{cases} b_1 & n = 1 \\ a_1 g(n-1) & N \leq n \leq 2 \end{cases} \quad (3.141)$$

As shown before, in figure (3.10), there is a range of values of optimal values of  $K$  that give optimal results. For  $G_1(z)$ , this range is 1.84 to 2.19. By optimal value we mean that value of  $K$  which takes the minimum number of ILC iterations to converge. Using the conventional ILC scheme and starting with a value of  $K = 0.1$  the system converges at iteration 146.

Using the proposed approach, the values of  $a_1$  and  $b_1$  is identified after first iteration with  $a_1 = 0.8187$  and  $b_1 = 0.0906$ . Starting with the same chosen starting value of  $K = 0.1$  and  $\mu = 0.1$ , the system converges at iteration 6 on first run. Here first run means, system used for the first time. Other starting values of  $K$  could also have been

chosen with similar results, as  $K$  is learnt adaptively. The value of  $K$  is stored in memory after each run. The number of iterations decrease at each run until  $K$  reaches its optimal value.

Even if  $K$  starts from a much higher value, say 5,  $K$  re-adjusts and settles at some optimal value between 1.84 to 2.19. A plot of  $K$  vs. number of runs is shown in Figure 3.16.

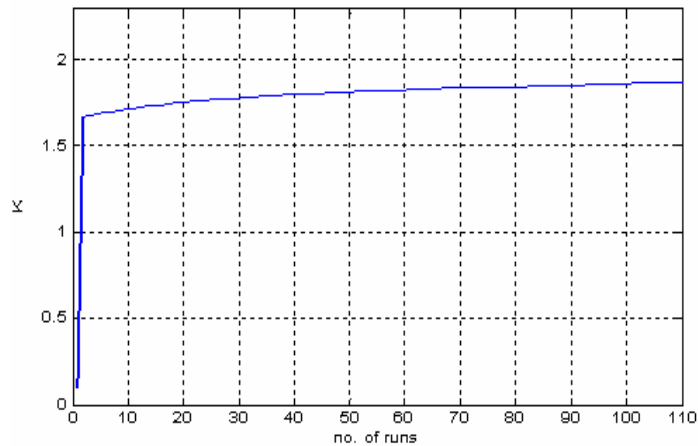


Figure 3.16: Learnt values of  $K$  with an initial value of 0.1.

The plot clearly shows  $K$  being learnt towards the optimal value as number of runs increase. A plot of norm of error vs. number of iterations is exhibited in figure 3.17.

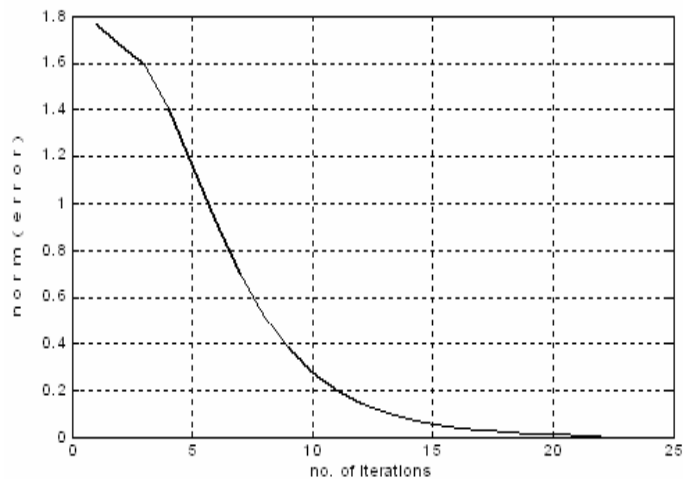


Figure 3.17: Behaviour of norm of error as iterations increase.

If the system undergoes a change in parameter due to wear or any other reason the approach has the capability to readjust its learnt parameters. As an example, suppose another first order system

$$y(k+1) = 0.9046y(k) + 0.09516u(k) \quad (3.142)$$

With  $a_1 = 0.8187$  and  $b_1 = 0.09063$ .

A plot of  $K$  vs. number of iterations for this system, using conventional ILC, is given in figure 3.18.

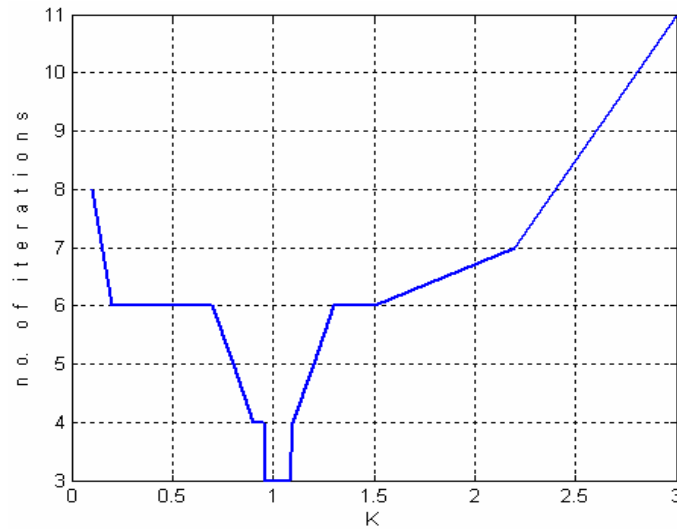


Figure 3.18: Number of iterations taken to converge against different values of  $K$ .

The optimal range for  $K$  is approximately 0.96-1.1. We now suppose that the system in equation (3.140) changes to (3.142) at sixth iteration, just when the system is about to converge. A 3D plot of  $y_d(j)(k)$  (thick dotted line) and  $y_j(k)$  (thin solid line) vis-a-vis number of iterations, is shown in figure 3.19.

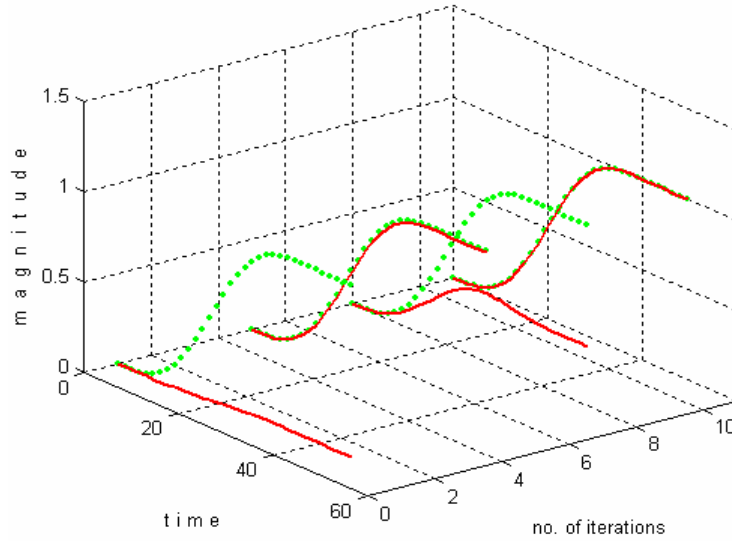


Figure 3.19: The proposed approach tracking the desired output as the system is changed in 6<sup>th</sup> iteration.

The behaviour of  $K$  during this shift in system is recorded in figure 3.20.

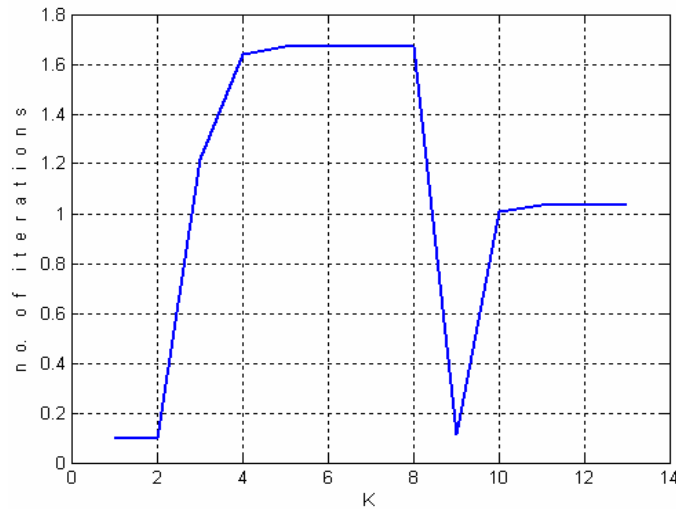


Figure 3.20: Readjustment done by the approach in the value of  $K$  as the system is changed in 6<sup>th</sup> iteration.

The scheme was able to readjust the value of  $K$ , within the new optimal range, for the modified system.

### 3.4.4.2 Car Suspension System

Simulation results from another system,  $G_3(z)$ , given in Appendix A, are discussed below. The system can be written in difference equation form as  $y(k+2) = 0.2779y(k+1) - 0.006738y(k) + 0.03052u(k+1) + 0.005925u(k)$

With  $b_1 = 0.03052$ ,  $b_2 = 0.005925$ ,  $a_1 = 0.2779$ ,  $a_2 = -0.006738$  and

$$g(n) = \begin{cases} b_1 & , \quad n = 1 \\ a_1 b_1 + b_2 & , \quad n = 2 \\ a_1 g(n-1) + a_2 g(n-2) & , \quad N \leq n \leq 3 \end{cases} \quad (3.143)$$

A plot of  $K$  vs. number of iterations for this system using conventional ILC is given in figure 3.21 below.

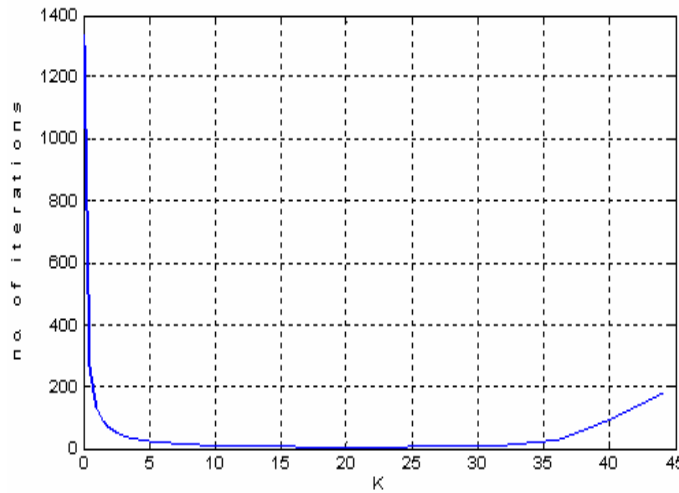


Figure 3.21: Number of iterations taken to converge against different values of  $K$ .

The gain,  $K$  here has optimal values from approximately 18 to 22. Using conventional ILC scheme and starting with a value of  $K = 0.1$  the system converges at iteration 1335. Other starting values of  $K$  gave similar results.

Using the proposed approach the values of  $a_1$  and  $b_1$  were identified after first iteration. These are  $a_1 = 0.2779$  and  $b_1 = -0.0067$ . The parameters  $a_2$  and  $b_2$  were not

identified as it was observed that parameters  $a_1$  and  $b_1$  always had maximum effect on the final results and in most cases there was no need to identify any more parameters. Starting with  $K = 0.1$  and  $\mu = 0.1$  the system converged at iteration 83 on first run. A plot of norm of error vs. number of iterations is shown in figure 3.22 below.

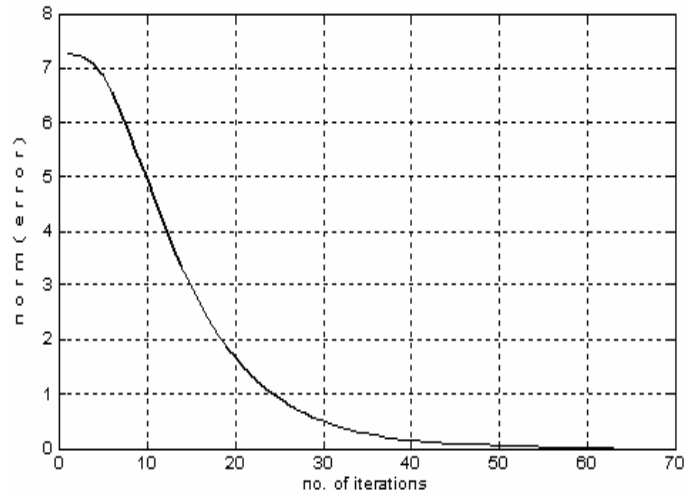


Figure 3.22: Behaviour of Euclidean norm of error for  $G_3(z)$ .

Figure 3.23 shows the plot of  $yd_j(k)$  (thick dotted line) and  $y_j(k)$  (thin solid line) against number of iterations.

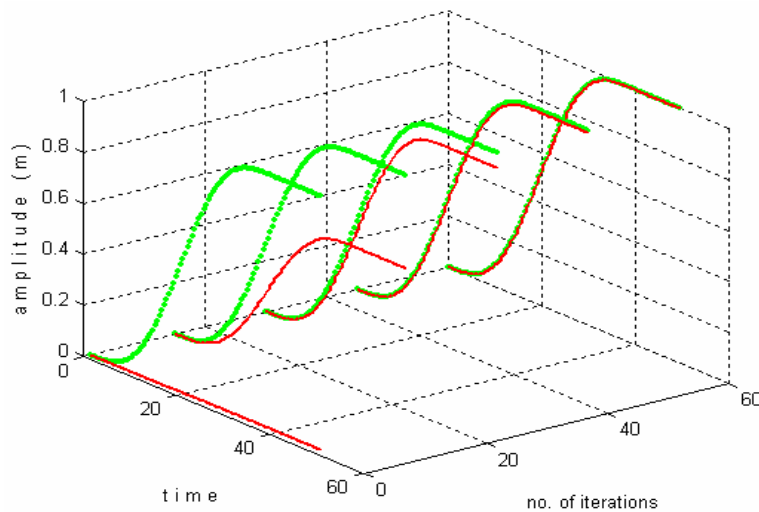


Figure 3.23: The proposed approach tracking the desired output.



The approach has the capability to adapt for change in performance requirements at run time. As an example, if the desired output is changed during run time the behaviour of the approach is presented in figure 3.24.

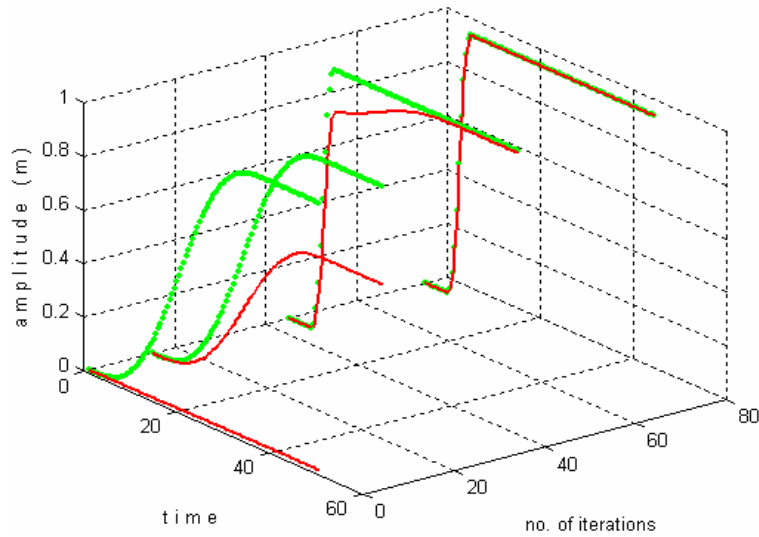


Figure 3.24: The proposed approach tracking the changing desired output.

The error recorded during this learning process as desired output was changed during run time is presented in figure 3.25.

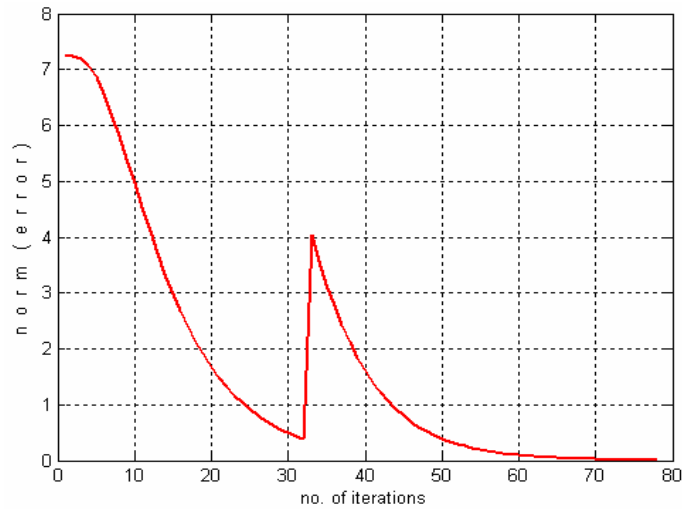


Figure 3.25: Behavior of Euclidean norm of error for a changing desired response.

The learning performance of  $K$  during this run is shown in figure 3.26.

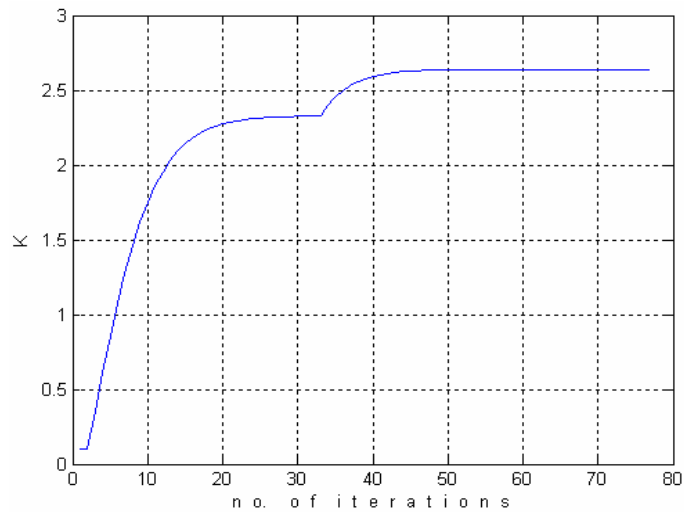


Figure 3.26: Learnt values of  $K$  as the desired response changes in real time.

The plot shows a readjustment in  $K$  as the desired response changes.

### 3.4.4.3 A Non-Linear System

The results from a second order non-linear system (NLS) are also presented to show the effectiveness of the approach. A plot of  $K$  vs. no. of iterations using the conventional ILC scheme is shown in figure 3.27.

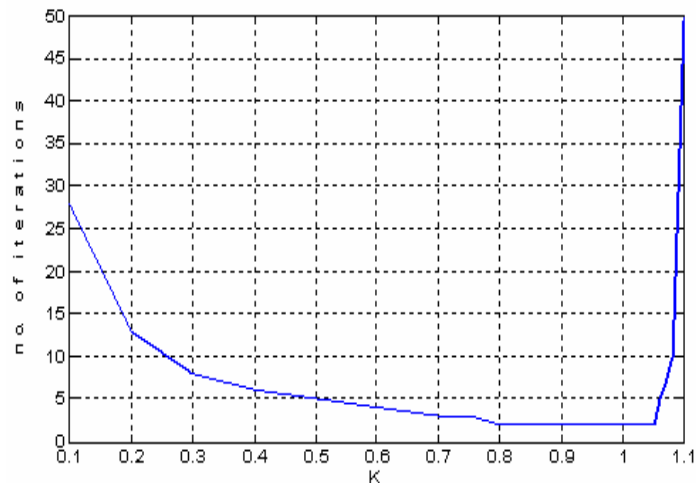


Figure 3.27: Number of iterations taken to converge against different values of  $K$ .

This plot again emphasises the fact that there is a range of values of  $K$  that produce minimum iterations. For the non-linear system this range is approximately between 0.8 and 1.05.

The values of  $a_1$  and  $b_1$  identified after first iteration were  $a_1 = 0.4148$  and  $b_1 = 0.5902$ . Figure 3.28 describes the behaviour of the system as it converges at 6<sup>th</sup> iteration.

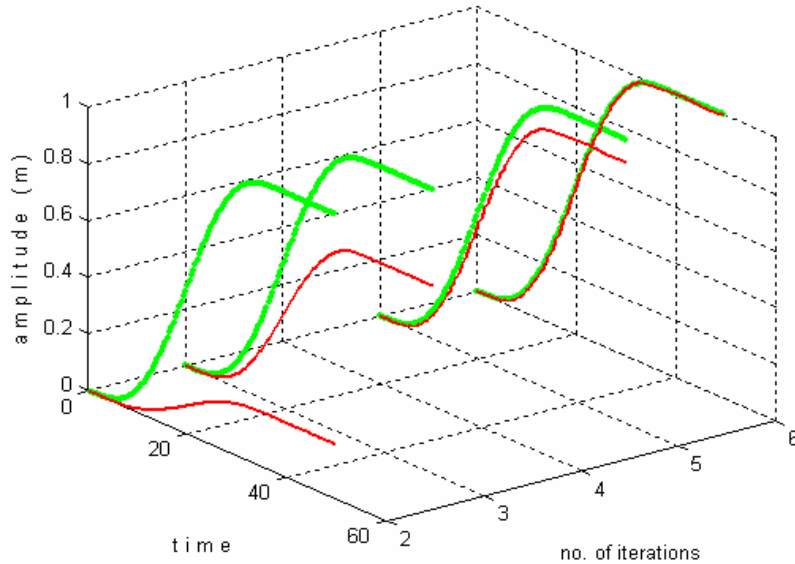


Figure 3.28: The proposed approach learning the desired output.

During this process the response of norm of error is plotted in figure 3.29.

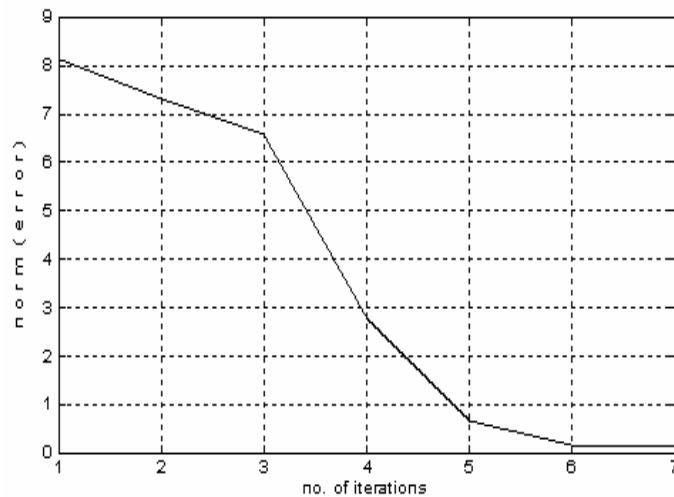


Figure 3.29: Behaviour of Euclidean norm of error for a non-linear system.

To get a measure of the effectiveness of this Identification based Adaptive Iterative Learning Scheme, a comparison between this scheme and the conventional ILC is given in the next section.

### 3.4.5 Discussion and comparison

Simulation results presented in previous section are tabulated in Table 3.1 and Table 3.2 for comparison. All results were taken with  $K = 0.1$  as the initial value. Other values of  $K$ , gave similar results as  $K$  was learnt adaptively. For example, starting with an initial value of  $K = 5$  for  $G_1(z)$ , the approach converges at iteration 8 at first run and iteration 5 on second run. Table 3.1, column 2, under the heading “Conventional ILC” gives the number of iterations it took to converge with a conventional ILC. The number of iterations will not decrease under the conventional scheme; no matter how many times (runs) the scheme is used. Column 3 presents the number of iterations it takes to converge using the proposed schemes. The numbers of iterations are significantly lower compared to the conventional scheme even at first run. This is due to the adaptive nature of the scheme. After every run the values of gain matrices are stored in memory and are used in the next run as initial values. The last column in Table 3.1, show the number of iterations it took to converge at second run.

<b>System</b>	<b>Conventional ILC</b> (Iterations)	<b>Proposed Scheme</b> After 1st run (Iterations)	<b>Proposed Scheme</b> After 2nd run (Iterations)
<b>SS</b>	146	6	2
<b>CSS</b>	1335	83	42
<b>NLS</b>	35	6	3

Table 3.1: Comparison between the conventional ILC and the proposed scheme.

In this table, for SS and CSS the convergence criteria is  $\|e_j(k)\| < 0.01$ , while for NLS with conventional ILC, it is  $\|e_j(k)\| < 0.4$ .

Table 3.2 shows the values of K learnt as the approach is repeatedly used. For a conventional system, 2nd column, under the heading ‘‘Conventional ILC’’, the value of K used is noted for the three systems presented in previous section. This value of K does not change, no matter how many times the system is used. This is because conventional scheme is not adaptive. For the proposed scheme as K is adaptive, its value changes from iteration to iteration. This can be seen from the last two columns of Table 3.2. As the number of runs increase, the proposed approach learns the value of K that will give the minimum iterations.

<b>System</b>	<b>Conventional ILC</b> (value of K)	<b>Proposed Scheme</b> After 1st run (value of K)	<b>Proposed Scheme</b> After 2nd run (value of K)
<b>SS</b>	0.1	1.98	1.99
<b>CSS</b>	0.1	2.32	3.01
<b>NLS</b>	0.1	0.781	0.788

Table 3.2: Values of K learnt for different systems, using conventional ILC and the proposed algorithm with a starting initial value of 0.1.

As opposed to conventional ILC which uses fixed gain, the suggested approach adjusts the gain adaptively and consequently the numbers of iterations are significantly reduced. For the Simple System, number of iterations came down from 146 to 6, for Car Suspension System, from 1335 to 83 and for non-linear system, from 35 to 5 at first run, as shown in Table 3.1. It is pointed out that ‘Memory’ helps to reduce the number of iterations further in subsequent runs. Though this comparison is not exhaustive, it clearly

shows that the number of iterations decreases continuously as optimal value of  $K$  is being learnt.

### 3.5 Cost Function

Usually the cost function applied is to minimize square of error or sum of square of error. As explained in section 3.2.1 and 3.4.2, we proposed a novel cost function, sum of square of error for next iteration. This cost function, helped in the derivation of proposed adaptive schemes. The proposed schemes are not only focusing on reducing error but also are looking at finding the next input to the plant. This opens the opportunity to develop other cost functions. The research on these innovative, unconventional cost functions, gave interesting results. One of the more useful results are presented in this section.

#### 3.5.1 Difference of Input (Approach-4)

Let the cost function be the difference in current input and next input, squared.

$$J = \sum \left( u_j(k) - u_{j+1}(k) \right)^2 \quad (3.144)$$

Using (3.116)

$$J = \sum \left( u_j(k) - \left( u_j(k) + Ke_j(k) \right) \right)^2 \quad (3.145)$$

$$J = \sum \left( u_j(k) - u_j(k) - Ke_j(k) \right)^2 \quad (3.146)$$

$$J = \sum \left( -Ke_j(k) \right)^2 \quad (3.147)$$

Equation (3.147) can be written in vector notation form as

$$J = \mathbf{K}^T (-\mathbf{E}^T) (-\mathbf{E}) \mathbf{K} \quad (3.148)$$

Applying the  $\nabla$  operator to the cost function

$$\mathbf{g} = \nabla J = 2\mathbf{E}^T \mathbf{E} \mathbf{K} \quad (3.149)$$

To find the value of  $\mathbf{K}_j$  which will give us minimum number of iterations we use the steepest descent formula given in (3.34). As  $\mathbf{E}$  is negative in the gradient function the direction of the steepest descent will be opposite to what it would have been with positive  $\mathbf{E}$ , i.e. (3.34)  $\Rightarrow$

$$\mathbf{K}_{j+1} = \mathbf{K}_j + \frac{1}{2} \mu \mathbf{g} \quad (3.150)$$

$$\mathbf{K}_{j+1} = \mathbf{K}_j + \frac{1}{2} \mu (2\mathbf{E}^T \mathbf{E} \mathbf{K}) \quad (3.151)$$

$$\mathbf{K}_{j+1} = \mathbf{K}_j + \mu \mathbf{E}^T (\mathbf{E} \mathbf{K}) \quad (3.152)$$

$$\mathbf{K}_{j+1} = \mathbf{K}_j + \mu \mathbf{E}^T (\mathbf{K} \mathbf{E}) \quad (3.153)$$

Rearranging equation (3.117)  $\Rightarrow$

$$\mathbf{K} \mathbf{E} = \mathbf{U}_{j+1} - \mathbf{U} \quad (3.154)$$

Putting (3.154) in (3.153)  $\Rightarrow$

$$\mathbf{K}_{j+1} = \mathbf{K}_j + \mu \mathbf{E}^T (\mathbf{U}_{j+1} - \mathbf{U}) \quad (3.155)$$

The difference in inputs and the error are used for the calculation of next gain value. These values are already available. We now present some simulation results to show the effectiveness of this adaptive gain law.

### 3.5.1.1 Simulation results

Results from three systems  $G_1(z)$ ,  $G_3(z)$  and NLS are presented in this section. For  $G_1(z)$ , a plot of error norm as number of iterations increase, for a starting value of  $K = 0.1$ , is given in figure 3.30 below.

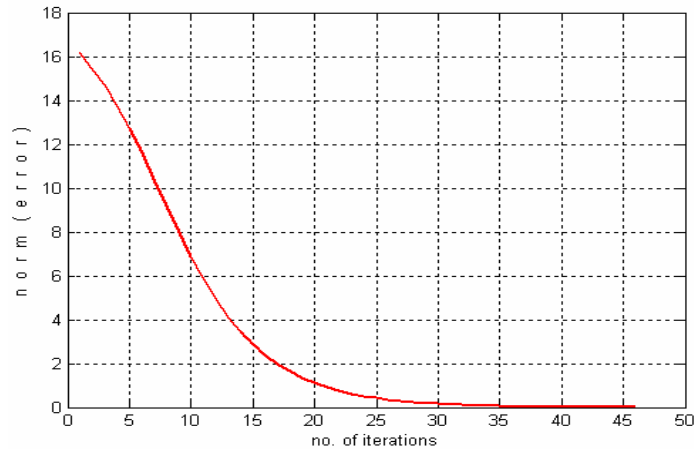


Figure 3.30: Behaviour of error as iterations increase.

It took 44 iterations to converge at first run. The learnt value of  $K$  is stored in memory and used again in the next run. The number of iterations decreased after every run, until  $K$  reached the optimal range, 1.84 to 2.19. The learnt values of  $K$  for a starting value of 0.1 are shown in figure 3.31.

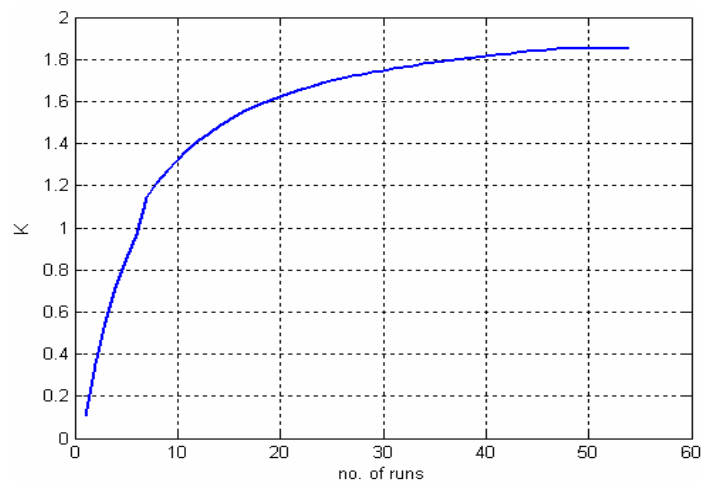


Figure 3.31: Learnt values of  $K$  as task is repeated.



Though a bit slow, the scheme was able to learn the optimal value of  $K$ . The number of iterations it took to converge as the system is used again and again is plotted in figure 3.32.

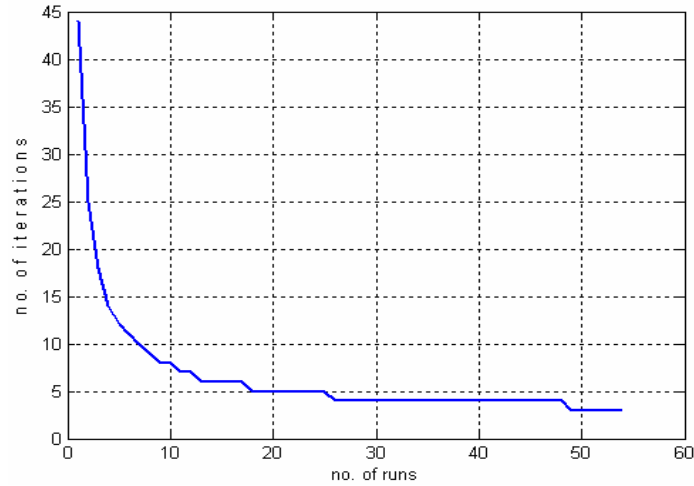


Figure 3.32: Number of iterations for convergence as the task is repeated.

The iterations are decreasing continuously. Similar behaviour was observed for different systems and with different starting values of  $K$ .

For  $G_3(z)$  it took 238 iterations to converge at first run. The error produced during that run is exhibited in figure 3.33.

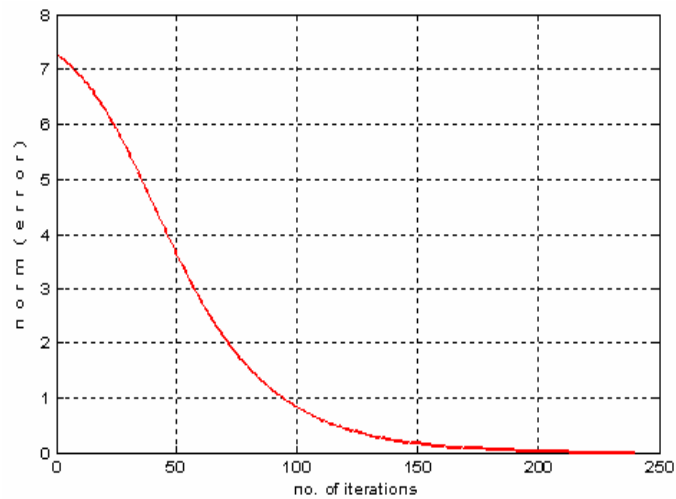


Figure 3.33: Behaviour of error as iterations increase.

The output of the system with the desired output, using this scheme is exhibited in a 3-D plot in figure 3.34.

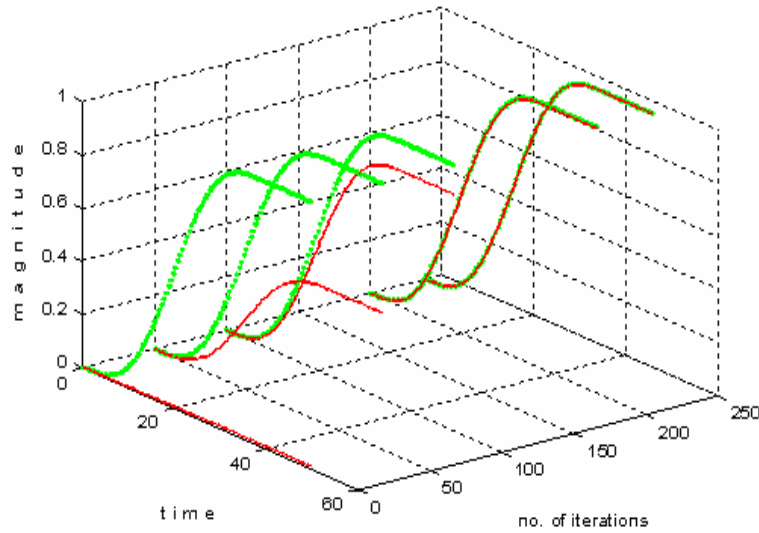


Figure 3.34: The proposed approach trying to follow the desired output.

Using NLS, it took 30 iterations on first run which steadily came down to 16 iterations at the 5<sup>th</sup> run. At 5<sup>th</sup> run the behaviour of error is plotted in figure 3.35.

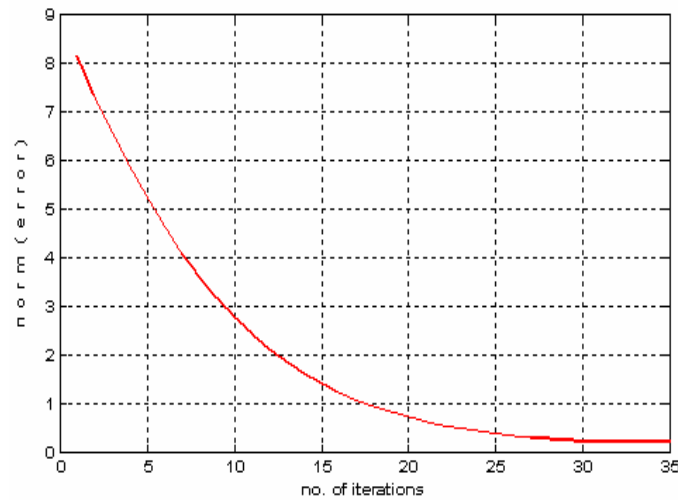


Figure 3.35: The performance of error as iterations increase.

The plot shows an exponential decrease in error.

The next section uses the ideas that if Iterative Learning Control (ILC) can benefit from previous trails, why not use a similar law to update gain values.

### 3.6 Iterative Learning Control with an Iterative Learning Gain (Approach-5)

This section describes the design of an adaptive iterative learning controller with an iterative learning gain (ILCILG). This proposed scheme extends the idea of ILC further and suggests that the information obtained from one trial should also be used to improve control algorithm parameter, the gain matrix.

The proposed approach is explained in block diagram form in figure 3.36.

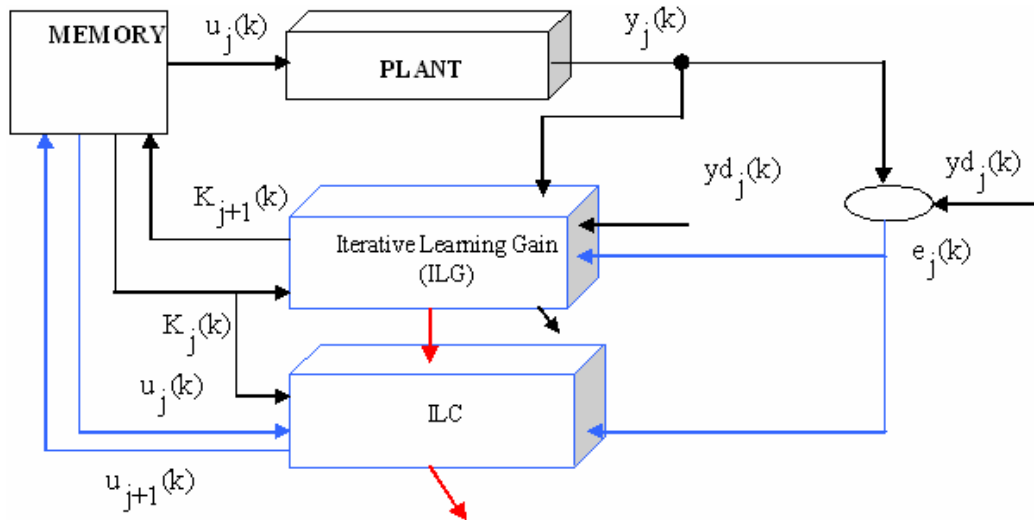


Figure 3.36: Block diagram of the proposed scheme.

Input  $u_j(k)$  is applied to the plant which produces an output  $y_j(k)$ . Error  $e_j(k)$  is the difference between the desired output  $y_d(j)(k)$  and the actual output of the system  $y_j(k)$ .

This error and the value of the previous gain are used in the block “Iterative Learning Gain (ILG)” to adjust gain. The “ILC” block also uses the same error to calculate next input to the plant,  $u_{j+1}(k)$ . After every iteration the value of  $K$  is stored in memory and retrieved before each iteration.

It is suggested to calculate  $K$  using the equation below.

$$K_{j+1}(k) = K_j(k) + \mu(\|y_d_j(k)\| - \|y_j(k)\|) \quad (3.156)$$

Here  $K_{j+1}$  is the value of  $K$  to be calculated for next iteration,  $K_j$  is the current value of  $K$ ,  $\|y_d_j(k)\| - \|y_j(k)\|$  is the difference between the norm of desired output and norm of current output and  $\mu$  is the step size parameter.

### 3.6.1 Convergence analysis

For equation (3.156), we define convergence as finding a value of  $K_j$  that will produce  $\|y_d\| - \|y_j(k)\| \leq \varepsilon$ , in minimum number of iterations. Here  $\varepsilon$  is the tolerance in error.

There can be three possible cases.

**Case 1:**  $\|y_d\| = \|y_j(k)\|$

In this case desired output and system output are same. Therefore, there is no change in  $K_j$ . According to equation (3.166)  $K_{j+1} = K_j$  and convergence is governed by (3.116) alone.

**Case 2:**  $\|y_d\| > \|y_j(k)\|$

In this case the desired output is larger than the actual output so the term  $\|y_d\| - \|y_j(k)\|$  will give a positive number. Under this condition,  $K_{j+1} > K_j$ , which in turn will raise the input to the system according to equation (3.116), resulting in an increase in output. This in turn will bring  $\|y_d\|$  and  $\|y_j(k)\|$  closer to each other.

**Case 3:**  $\|y_d\| < \|y_j(k)\|$

In this case the desired output is smaller than the actual output, so the term  $\|y_d\| - \|y_j(k)\|$  will give a negative number. Under this condition,  $K_{j+1} < K_j$ , which in

turn will reduce the input to the system according to equation (3.116), resulting in a decrease in output. This will in turn bring  $\|y_d\|$  and  $\|y_j(k)\|$  closer to each other.

For both cases 2 and case 3, the system will behave as to reduce  $\|y_d\| - \|y_j(k)\|$ . For smooth convergence whenever there is a change in sign in  $\|y_d\| - \|y_j(k)\|$  or when ever there is a shift from case 2 to case 3 or case 3 to case 2 we can reduce  $\mu$ . For this,  $\mu$  can be tied up with rate of change of  $\|y_d\| - \|y_j(k)\|$ . However, in this section  $\mu$  is kept constant.

The number of iterations has always been an issue with ILC. This scheme because of its simple mathematical structure can easily be implemented with lower memory requirements and simpler hardware as opposed to other such adaptive schemes which are computationally expensive.

We now present some simulation results obtained from three selected systems from Appendix A.

### 3.6.2 Simulation results

Simulation results from three systems  $G_1(z)$ , NLS and  $G_4(z)$  are discussed in this section. First, results from  $G_1(z)$  are presented. As shown previously in figure 3.10, this system has a range of values of  $K$ , which results in minimum number of iterations. The range for  $G_1(z)$  is 1.84 to 2.19.

Using the ILCILG technique, with starting values of  $K = 0.1$  and  $\mu = 0.01$ , the system converged at iteration 18, with a final value of  $K_{18} = 0.867$ , on first run. As shown in figure 3.36, the value of  $K_j$  is stored in memory after each run. The value stored in memory is used as the initial value for the next run. After a few runs this value settles with in the optimal range. Even if, initially a much higher value of  $K_j$  say 5 is

taken,  $K_j$  readjusts and settles at some value between 1.84 to 2.19. The rate of learning is governed by  $\mu$ . A plot of norm of error against number of iterations for first run is shown in figure 3.37.

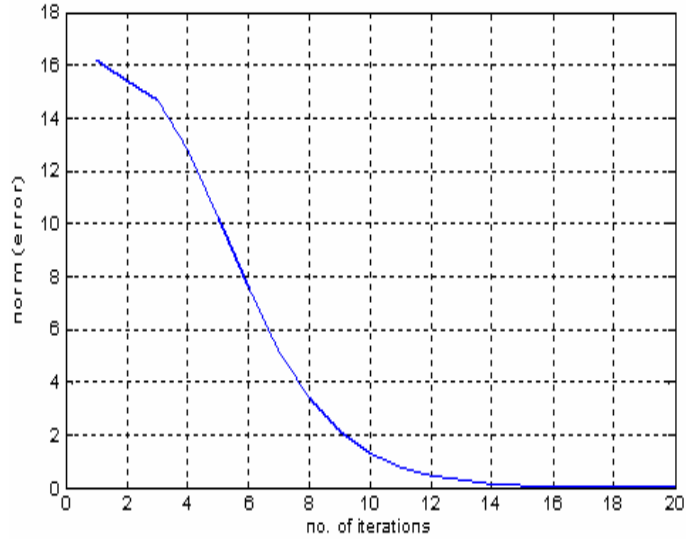


Figure 3.37: Behavior of Euclidean norm of error for the first order system.

The conventional ILC took 146 iterations to converge for this system, with similar settings.

The motor speed control system (MSCS) was also tested using the same initial gain,  $K = 0.1$ . Using the conventional ILC it took 661 iterations to converge. As the conventional ILC scheme is not adaptive it will always take 661 iterations to converge with  $K = 0.1$ . Using the proposed approach and starting with  $K = 0.1$  and  $\mu = 0.01$  the system converged at iteration 63 with a final value of gain  $K_j = 1.23$ , at first run.

A plot of norm of error against number of iterations for first run is shown in figure 3.38 below.

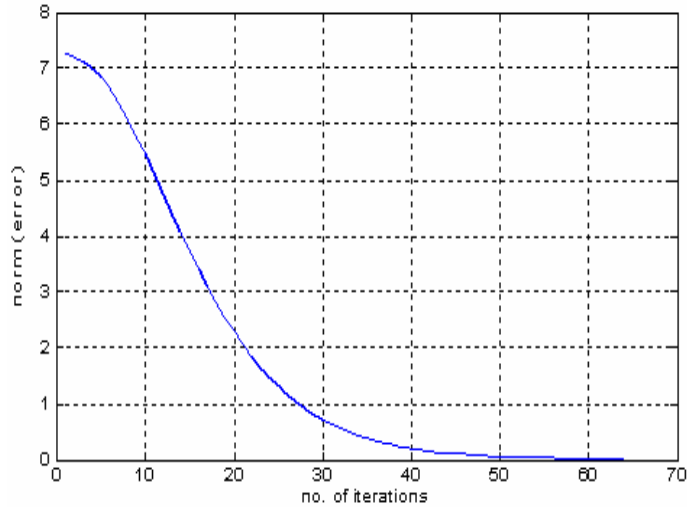


Figure 3.38: Behaviour of Euclidean norm of error for a motor speed control system.

Subsequent runs decreased the number of iterations further. A 3-dimensional plot of the desired output (shown in bold dotted lines) vs. actual output (shown in thin solid lines) against number of iterations is presented in figure 3.39 below. The plot clearly shows the output reaching the desired output as iterations increase.

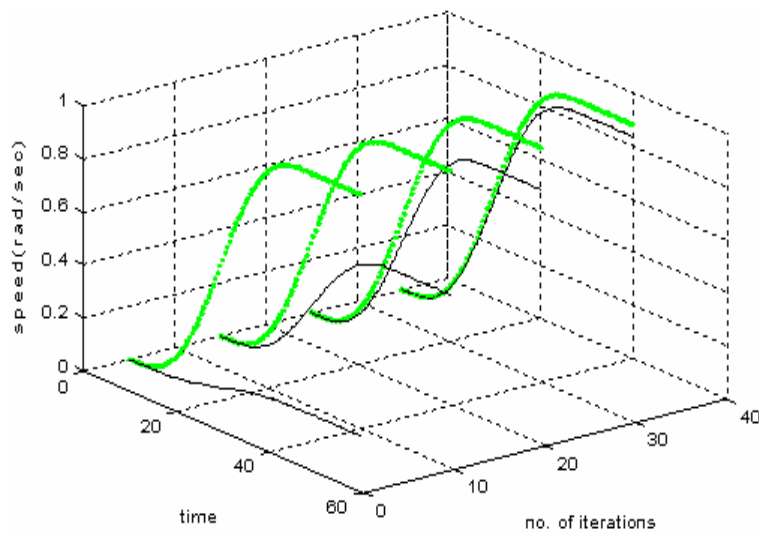


Figure 3.39: Output converging towards desired output.

Subsequent runs showed a decrease in number of iterations as  $K_j$  was updated.

If there is a change in system dynamics e.g. the damping ratio is doubled during operation, the approach can readjust at run time. Figure 3.40 shows the behaviour of error in case the system changes during run time. For this simulation the system was changed at iteration 20.

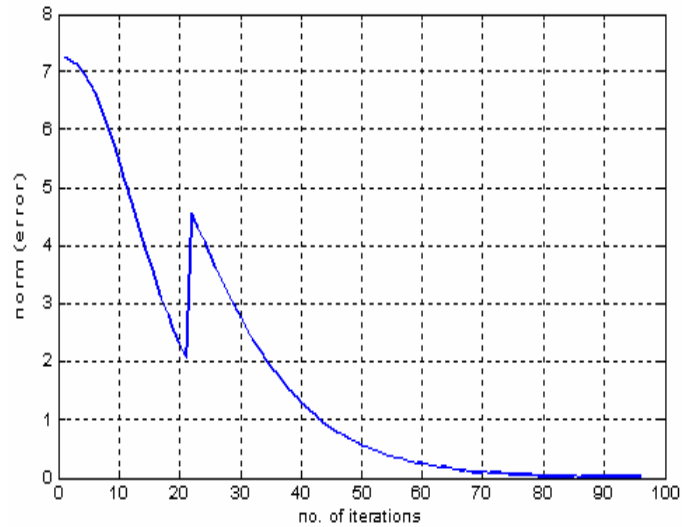


Figure 3.40: Behaviour of Euclidean norm of error as the system changes.

With this change in system dynamics, it took 95 iterations to converge. The behaviour of  $K_j$  is shown in figure 3.41. It can be clearly seen that gain makes a small readjustment at iteration 21.

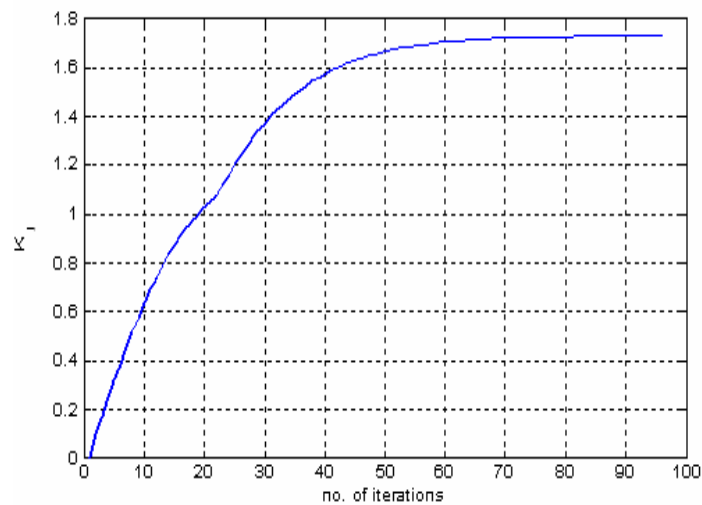


Figure 3.41: Behaviour of  $K_j$  against number of runs.



This readjustment is further clarified by a 3-D plot of the output in figure 3.42. The desired output (in dotted bold lines), is also presented for better comparison.

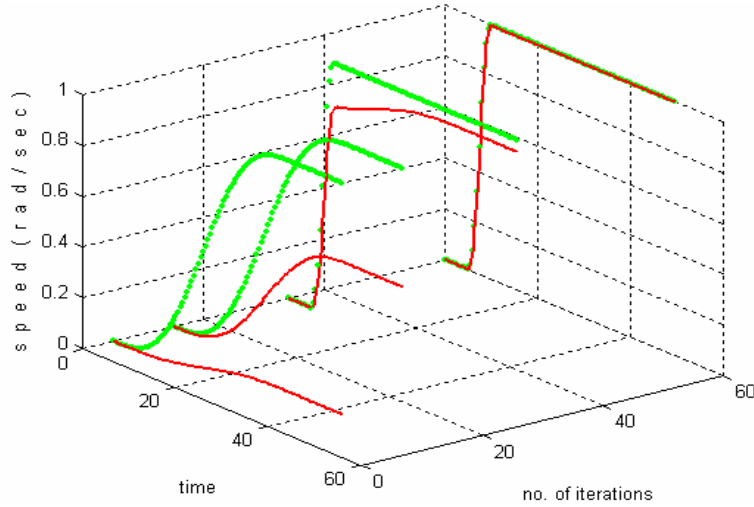


Figure 3.42: The proposed approach tracking the changing desired output.

The desired output was changed during iteration 20. It took 65 iterations to converge this time at first run.

For the non-linear system, the behaviour of  $K_j$  for different initial values of  $K$  is shown in figure 3.43. This figure shows that even starting with different initial values of  $K_j$  the scheme eventually settles to an optimal range, which for this non-linear system is between 0.8 and 1.05.

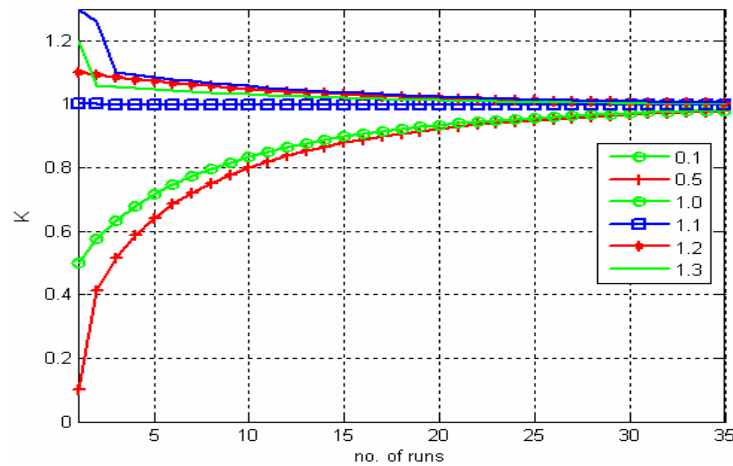


Figure 3.43: Behaviour of  $K$  against number of runs with different initial values of  $K$ .

The system converged at 11<sup>th</sup> iteration on first run. The behaviour of the norm of error for first run is shown in figure 3.44.

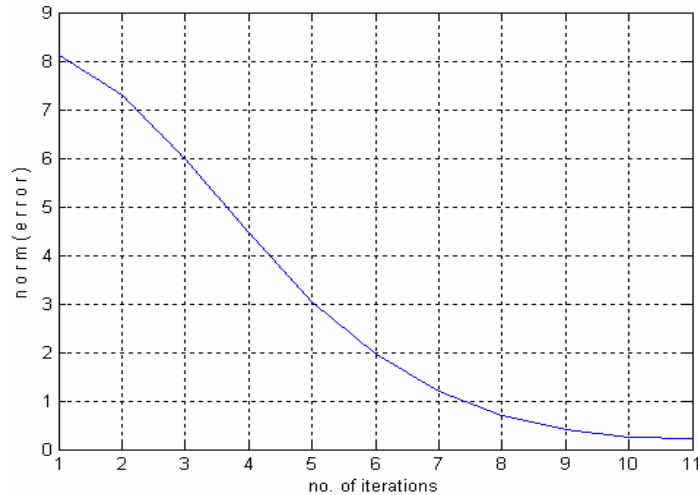


Figure 3.44: Behaviour of Euclidean norm of error for a non-linear system.

Subsequent runs reduced the number of iterations further. The results show that ILCILG performed better than the two cost function based schemes. A comparison with the conventional scheme will give us a measure of improvement made by ILCILG.

### 3.6.3 Discussions and comparison

Simulation results are tabulated in Table 3.3 for comparison. All results were obtained with  $K = 0.1$  and  $\mu = 0.01$  as initial starting values. A desired output, given in Appendix A, was defined for all these systems. Column 2 under the heading “Conventional ILC” shows the number of iterations it took to converge for a first order ( $G_1(z)$ ), motor speed control ( $G_4(z)$ ) and a non linear system. They are 146, 661 and 35 respectively. Column 3 shows the number of iterations taken to converge at first run using ILCILG. The numbers of iterations are significantly lower as compared to the conventional ILC scheme even at first run. At fifth run, as shown in the last column, the

numbers of iterations are reduced further as  $K_j$  is updated. Though this comparison is not extensive but the trend is obvious.

<b>System</b>	<b>Conventional ILC</b> (Iterations)	<b>ILCILG</b> At 1st run (Iterations)	<b>ILCILG</b> At 5th run (Iterations)
<b>First order</b>	146	18	4
<b>Motor speed control</b>	661	63	24
<b>Non linear</b>	35	11	4

Table 3.3: Some comparative results showing reduction in number of iterations for ILCILG controller.

Excellent results shown by the ILCILG controller made it a good choice to test it on a practical setup. The setup with results is presented in the next section.

### 3.6.4 Experimental setup and results

Experiments were made using QET DC motor kit by Quanser Consulting Inc [57]. The complete set up is shown in figure 3.45 below.



Figure 3.45: Quanser's DC motor kit used to test the approach.

The aim was to follow the desired signal which is to make the motor run at a speed of 100 rad/sec. The control software was developed in MATLAB. A 3-dimensional plot of the desired output (shown in dotted lines) vs. actual output (shown in solid lines) against number of iterations for the first five iterations are shown below. The plot clearly shows the system output reaching the desired output as iterations increase.

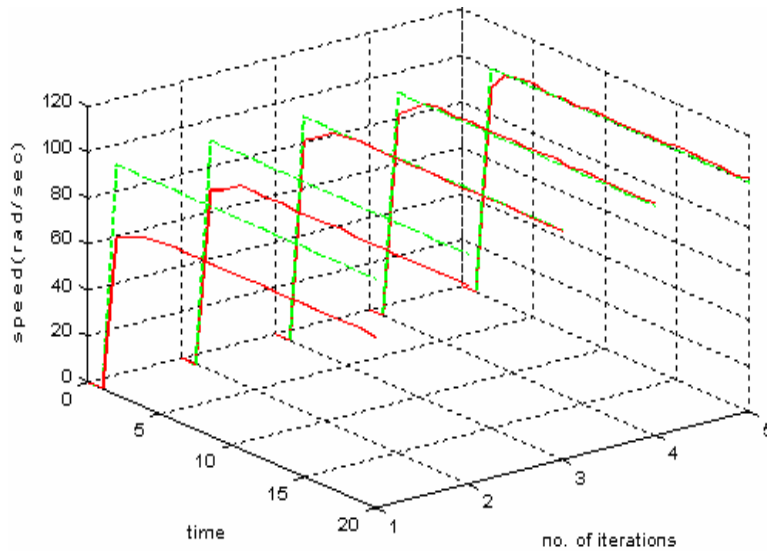


Figure 3.46: Output converging towards desired output.

A plot of norm of error for different iterations is shown in figure 3.47. The plot shows continuous decrease in error.

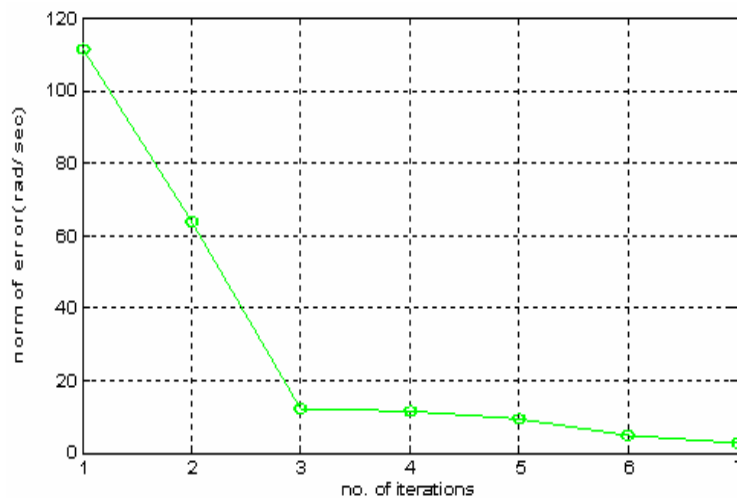


Figure 3.47: Behaviour of Euclidean norm of error.

As per equation (3.156), the ILCILG controller learns the values of gain  $K_j$ , increasing speed of convergence. The value of  $K_j$  learnt at first iteration was 0.278. This value as per figure 3.36 is stored in memory, to be used in later iterations. The behaviour of  $K_j$  is presented in figure 3.48.

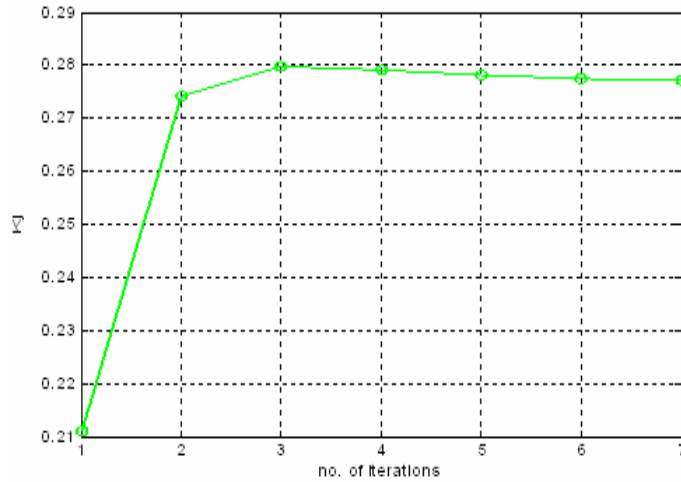


Figure 3.48: Learnt values of gain as iterations increase.

This adaptive behaviour of  $K_j$  helped to reduce the number of iterations. The ILC used in this approach was OSATILC. Using this setting, the ILCILG was made to track real time trajectories also.

### 3.6.5 Real time tracking using iterative learning control with an iterative learning gain

The scheme has the capability to track objects and paths in real time, unlike ILC schemes given in the literature. For  $\mu = 0.001$ , equation (3.156) can be written as

$$K_{j+1}(k) = K_j(k) + 0.001(\|y_d(k) - y_j(k)\|) \quad (3.157)$$

For real time implementation this equation is readjusted to

$$K(k) = K(k-1) + 0.001e(k) \quad (3.158)$$

Where  $K(k)$  is the value of  $K$  for current sample.  $K(k-1)$  is the value of  $K$  for previous sample and  $e(k)$  is the difference between the desired output and actual output.

### 3.6.5.1 Simulation results

For simulation purposes  $G_4(z)$  system was used. A desired motor speed trajectory was fed into the scheme. The scheme tried to follow the trajectory in real time. The performance is shown in figure 3.49.

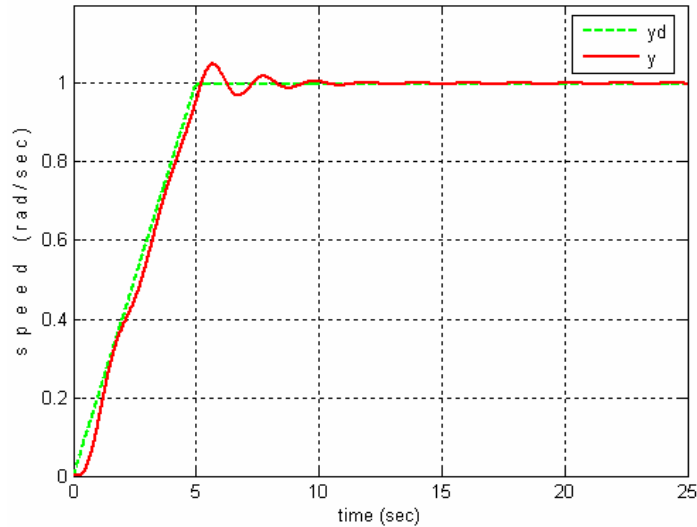


Figure 3.49: Tracking behaviour.

The behaviour of error as the trajectory is being followed is shown in figure 3.50.

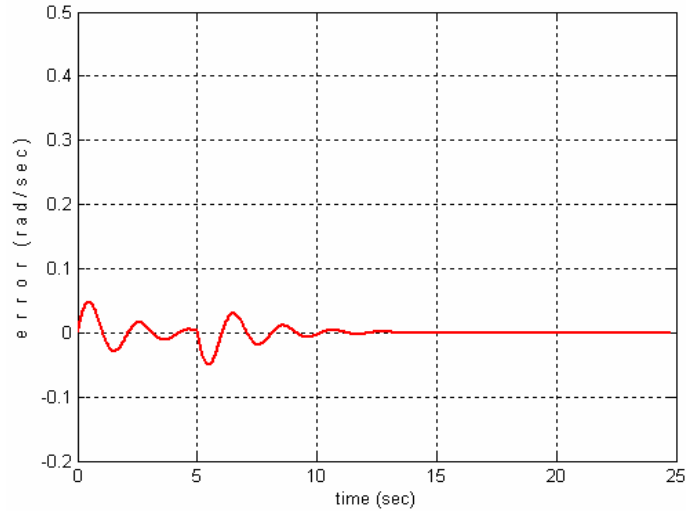


Figure 3.50: Error as desired trajectory is being tracked.

During this tracking performance the values of  $K$  learnt are indicated in figure 3.51.

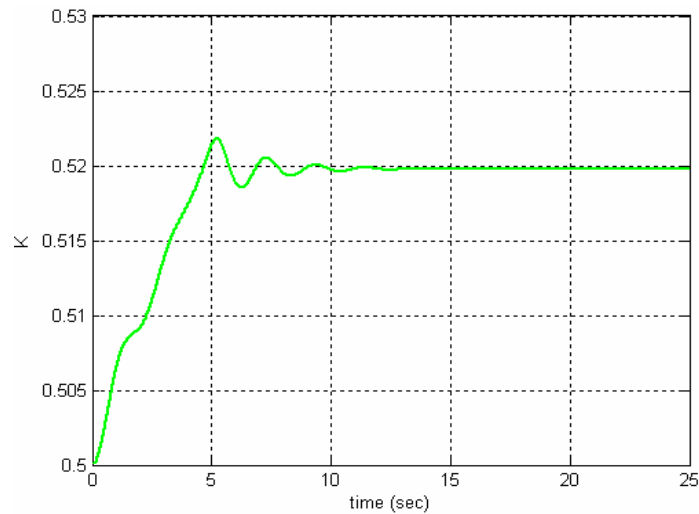


Figure 3.51: Learnt values of  $K$  as desired trajectory is tracked.

The same system was also made to track other waveforms, like sigmoid, different sinusoidals, parabolic etc. The tracking performance while following a sinusoidal track is exhibited in figure 3.52.

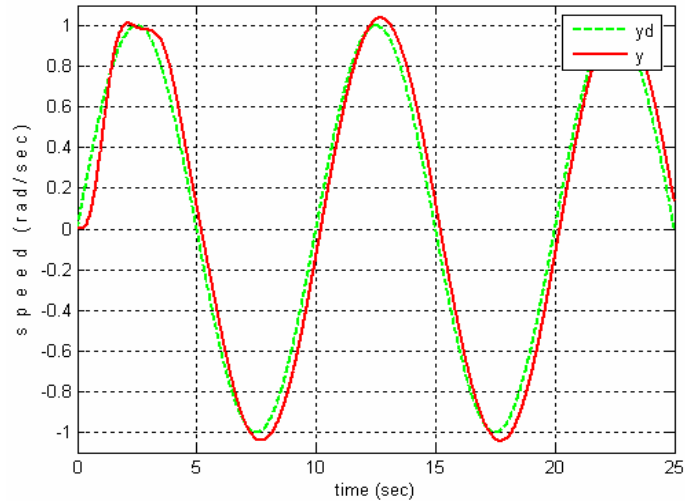


Figure 3.52: Tracking a sine wave.

The plot shows very good tracking performance by the ILCILG controller. The error during this tracking performance is plotted in figure 3.53.

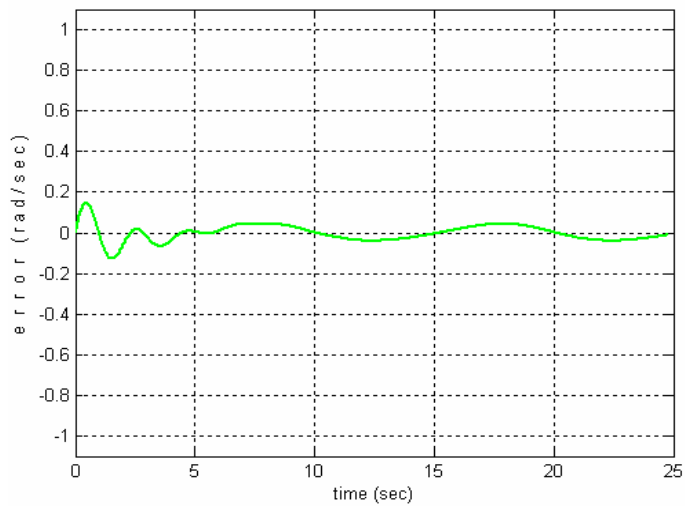


Figure 3.53: Behaviour of error as sine wave is tracked.

During this sinusoidal speed chase the values of  $K$ 's used are shown in figure 3.54.



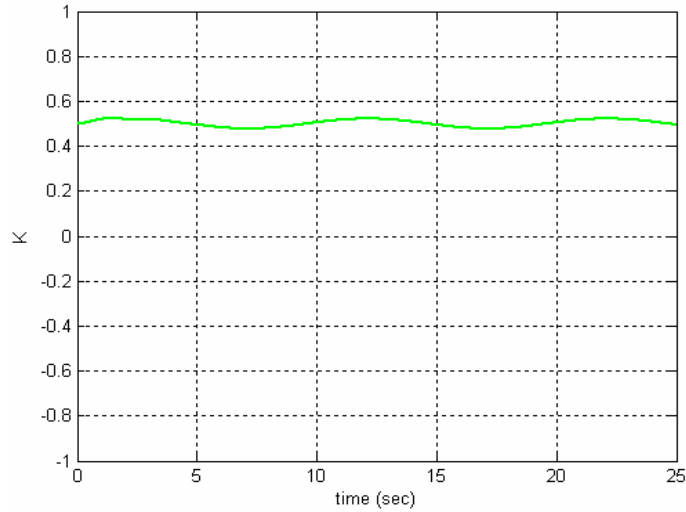


Figure 3.54: Learnt values of  $K$  as desired trajectory is tracked.

The tracking performance was tested using the same QET DC motor kit by Quanser Consulting Inc. The results from one such experiment are discussed in the next section.

### 3.6.5.2 Real time tracking using an experimental set up

The QET DC motor kit with the ILCILG controller was also made to track a speed of 100 rad /sec in real time. The control software was developed in MATLAB. A 3-dimensional plot of the desired output (shown in dotted lines) vs. actual output (shown in solid lines) against number of iterations for the first five iterations are shown in figure 3.55 below. The plot clearly shows the output reaching the desired output as iterations increase.

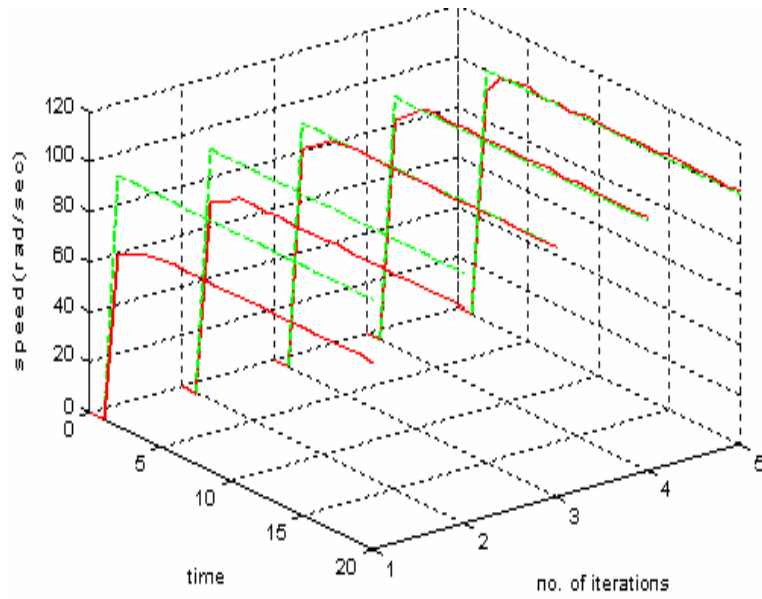


Figure 3.55: Output converging towards desired output.

For the first 13 samples this tracking is shown in figure 3.56.

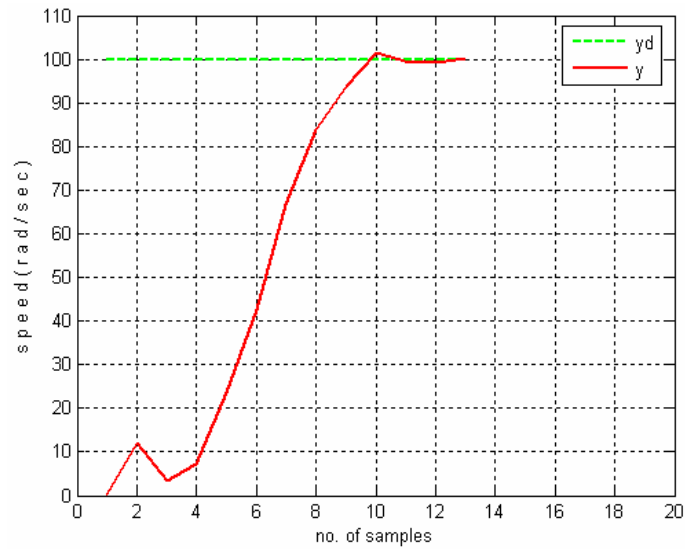


Figure 3.56: Quanser's DC motor tracking a desired step response with adaptive gain.

For a sampling rate of 10 samples per second the motor achieved the required speed in 1 sec. The performance of error as the desired speed is met is shown in figure 3.57.

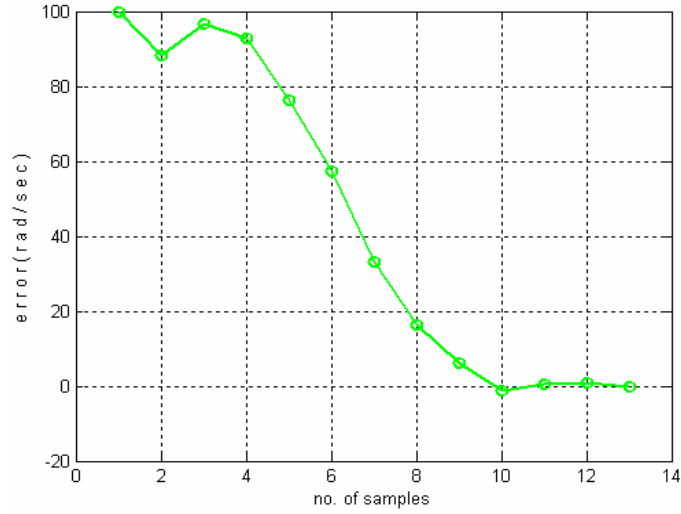


Figure 3.57: Error as trajectory is tracked.

To achieve this performance the input calculated by the proposed scheme is presented in figure 3.58.

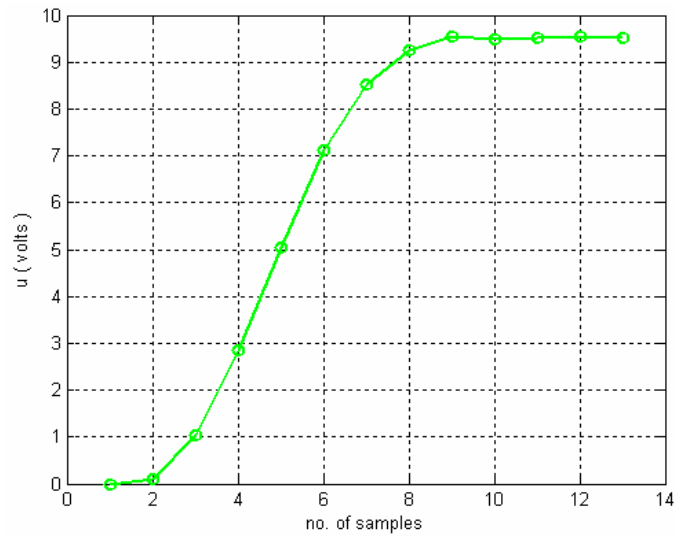


Figure 3.58: Input voltages learnt and supplied to the motor with adaptive gain.

The values of  $K$  learnt during this tracking is shown in figure 3.59.

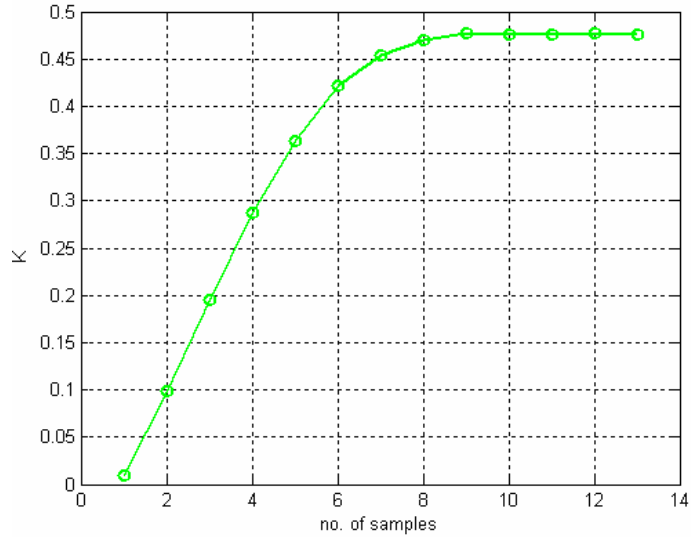


Figure 3.59: Learnt values of gain.

After initial learning, the value of  $K$  settles to approximately 0.47.

In all adaptive schemes presented so far, the step size parameter has been kept constant. This step size parameter can be varied also to increase or decrease the rate of convergence of  $K$ . The following section looks at one such possibility.

### ***3.7 Iterative Learning Control with an Iterative Learning Gain and Adaptive Step Size***

The proposed scheme is presented in figure 3.60 below.

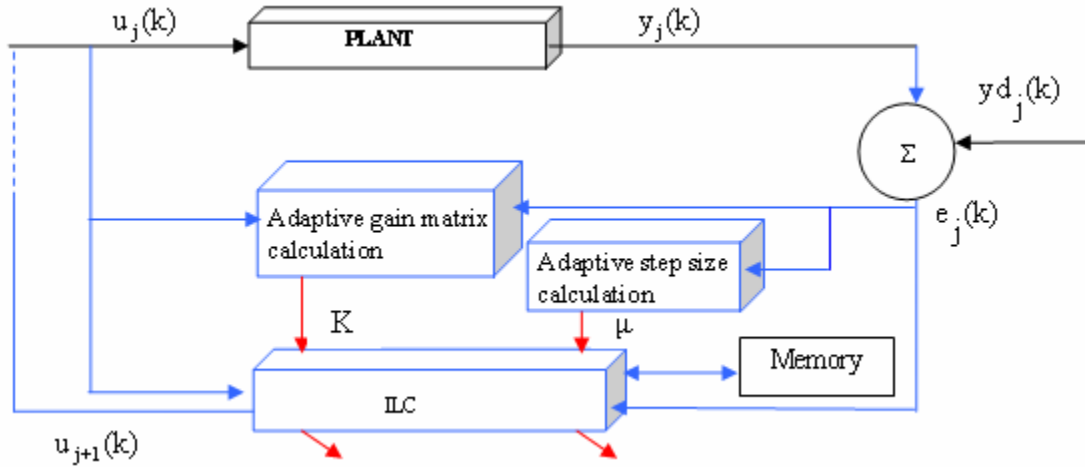


Figure 3.60: Block diagram representation of the proposed scheme.

Input  $u_j(k)$  is applied to the plant. This input produces an output  $y_j(k)$ . Error  $e_j(k)$  is the difference between the desired output  $y_{d_j}(k)$  and the actual output of the system  $y_j(k)$ . This error is used by the adaptive gain matrix block to adjust gain matrices. The iteration horizon is made use of by calculating the difference in magnitude of this error during different iterations to adjust the step size. The ILC block also uses the same error to calculate next input to the plant. The adjusted values of  $K$  and  $\mu$  are stored in memory.

Using steepest decent to find  $K$  which results in minimum number of iterations.

$$K_{j+1} = K_j - \frac{1}{2} \mu_j(k) g \quad (3.159)$$

Equation (3.124) can be rewritten for adaptive  $\mu$ .

$$K_{j+1} = K_j + \mu_j(k) E_j^T(k) G^T (Y_d - G U_{j+1}(k)) \quad (3.160)$$

Here  $\mu_j(k)$  is the step size parameter and  $G$  matrix is an  $N \times M$  matrix. The  $G$  matrix is lower triangular and for linear systems it is also Toeplitz.

### 3.7.1 Adaptive step size

The second dimension of a 2-D learning process called the learning horizon is used to make the step size adaptive.

$$\mu_j(k) = \begin{cases} 2.0 \times \mu_j(k) & , & \|e_{j-1}(k)\| - \|e_j(k)\| > 0 \\ 0.5 \times \mu_j(k) & , & \|e_{j-1}(k)\| - \|e_j(k)\| \leq 0 \end{cases} \quad (3.161)$$

Here  $\|e_{j-1}(k)\|$  is the Euclidean norm of error vector in the previous iteration and  $\|e_j(k)\|$  is the Euclidean norm of the error vector in the current iteration.

### 3.7.2 Simulation results

The scheme presented in this paper was tested through simulations using  $G_1(z)$ ,  $G_3(z)$  and NLS. For  $G_1(z)$ , using the proposed approach and starting with a  $K$  of 0.1 and  $\mu$  of 0.1, the system converges when the gain reaches 1.715 and  $\mu$  reaches 0.32 at iteration 6, on first run. The behaviour of  $K$  for the first run is shown in figure 3.61 below.

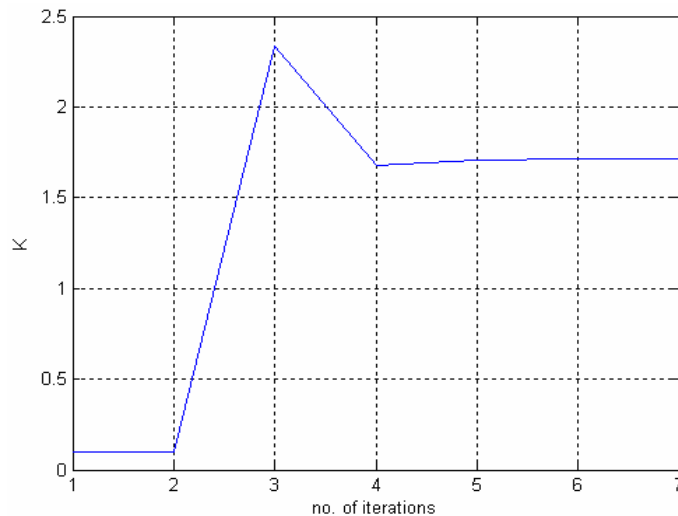


Figure 3.61: Variation of  $K$  for first run.

The value of  $K$  was stored in memory after each run. The number of iterations decreased after each run, until  $K$  settled in the optimal range. Figure 3.62 shows how the norm of error behaved during the first run.

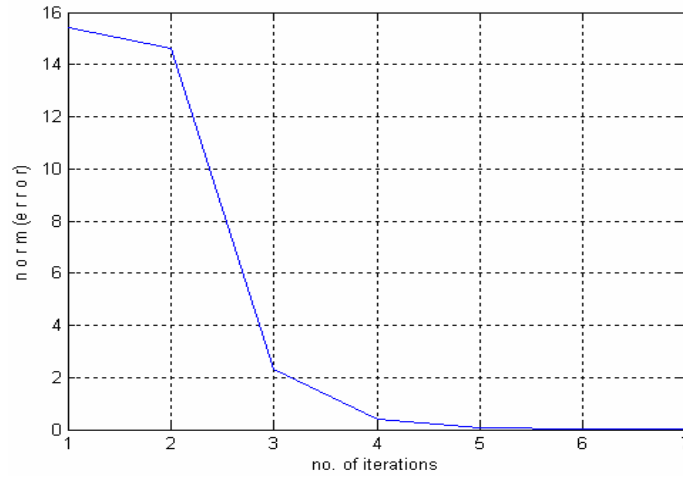


Figure 3.62: Behaviour of Euclidean norm of error.

For  $G_3(z)$ , again starting with  $K = 0.1$  and  $\mu = 0.1$ ,  $K$  reaches 12.61 and  $\mu$  reaches 163.8 after first run. The system converges at iteration 17 on first run as opposed to iteration 1335, if only the conventional ILC scheme is used. A plot of norm of error vs. number of iterations is shown in figure 3.63.

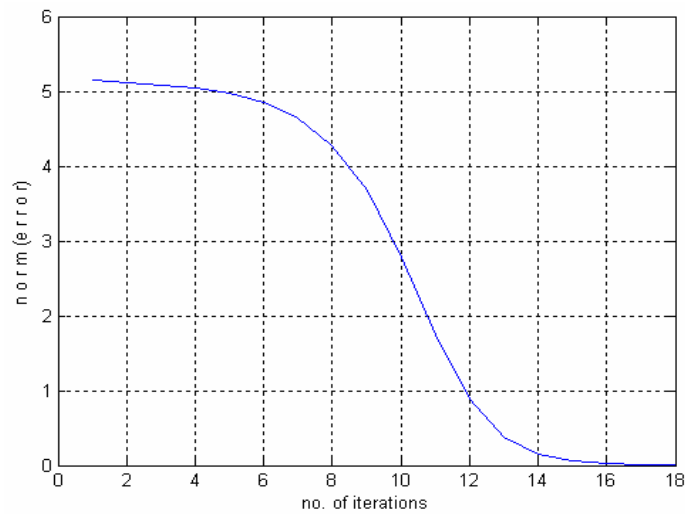


Figure 3.63: Behaviour of Euclidean norm of error for a mass, spring and damper system.

A 3-dimensional plot of the desired output (shown in dotted lines) vs. actual output (shown in solid lines) against number of iterations is shown in figure 3.64 below. The plot clearly shows the output reaching the desired value at the 17<sup>th</sup> iteration.

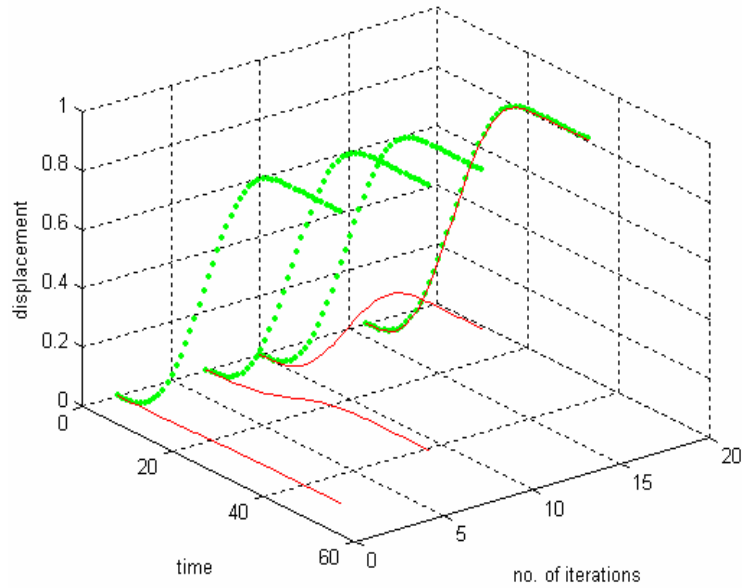


Figure 3.64: Output converging towards desired output.

Subsequent runs resulted in even fewer numbers of iterations, as value of  $K$  reached optimal range.

Similar data was recorded for the non-linear system. The figure below shows the system converging at 4<sup>th</sup> iteration. The adaptiveness of  $\mu$  is shown in figure 3.65 below.



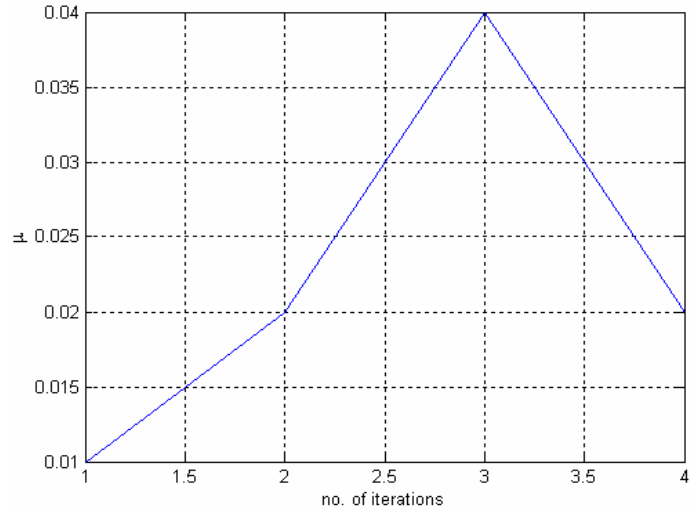


Figure 3.65: Variation of  $\mu$  for first run.

The system converged at 4<sup>th</sup> iteration on first run. The behaviour of the norm of error is shown in figure 3.66.

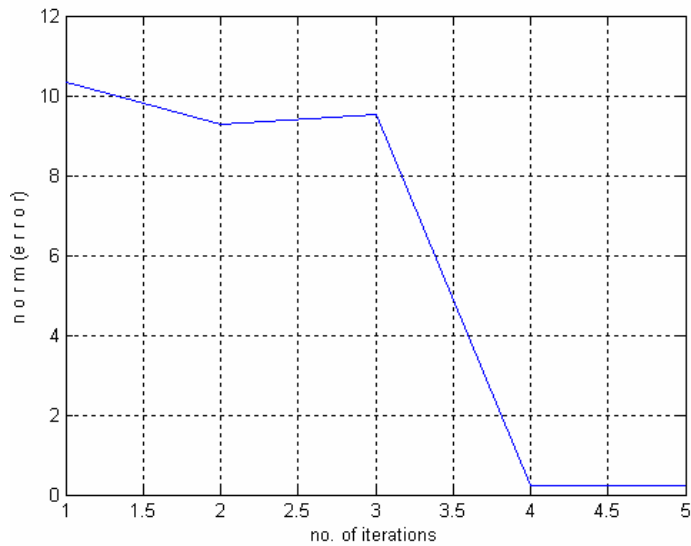


Figure 3.66: Behaviour of Euclidean norm of error for a non-linear system.

The error quickly reduced to within acceptable limit. The performance can be best compared through a table given in the next section.

### 3.7.3 Discussion and comparison

Simulation results are tabulated in Table 3.4 for comparison. All results were taken with  $K$  and  $\mu$  having an initial starting value of 0.1. A desired output was defined for all these systems. Number of iterations taken to learn an input which produced the desired output is given. The table shows a significant decrease in number of iterations, even at first run, using the proposed scheme as compared with conventional ILC scheme.

	<b>Normal ILC</b> (Iterations)	<b>ILCILG</b> With adaptive gain only (Iterations)	<b>ILCILG</b> With adaptive gain and adaptive step size (Iterations)
<b>First order system</b> (SS)	146	7	6
<b>Second order system</b> (CSS)	1335	61	17
<b>Non linear system</b> (NLS)	35	5	4

Table 3.4: Some comparative results.

The learning is further enhanced with adaptive step size parameter.

### 3.8 Summary

Following the research work presented in chapter 2, this chapter describes a number of adaptive iterative learning controllers. For these controllers a specific mathematical frame work was developed. The different schemes made use of combination of novel cost functions, gradient descent approach and innovative control laws. Stability and convergence criteria are also established. The performances are confirmed using a number of simulations on different models of practical systems.

A practical setup, using DC motor kit by Quanser Consulting was also used to test the effectiveness of the controllers in real world applications. All the controllers were able to readjust for changes in plant as well as desired response. The chapter ends with a proposal to control rate of adaptation using the step size parameter of the control laws.

The main focus of this research aimed at capturing the two main aspects of human behaviour, namely

(a) Learning from experience.

(b) Perception based approach.

We propose to tackle “Learning from experience” aspect through adaptive iterative learning. To tackle “perceptions”, we need to incorporate fuzzy logic.

The next chapter develops this philosophy.

## **4 SELF LEARNING FUZZY CONTROLLERS USING ITERATIVE LEARNING TUNER**

This chapter describes the design of an adaptive fuzzy controller using iterative learning to tune input membership functions and scaling factor(s). The control scheme consists of a fuzzy controller and learning control laws. People's perception about the meaning of a linguistic variable differs from person to person or even from expert to expert. This difference in perception usually leads to different fuzzy control designs. Somewhere within these designs lies the required design which meets specific performance criteria. The result of this research proposes an approach to tackle this uncertainty in perception, i.e., to find the required design by adjusting membership functions. This uncertainty was rarely tackled as a concept before Type-2 Fuzzy (T2-F) was invented. The membership functions are adaptively adjusted using iterative learning technique. The results show that the scheme is robust, cost effective and very simple to implement. It makes use of the non-linearity inherent in the fuzzy systems. Designing fuzzy controllers with desired performance specifications is not a trivial task. Even the specification of linguistic variables, a key concept in fuzzy system design, can be different from different experts, creating uncertainty in the design. This scheme tries to fill this gap by using a unique adaptive procedure for designing fuzzy controllers through iterative learning process.

### ***4.1 Problems in Fuzzy Logic Based Design***

It all started with the seminal concept of using fuzzy sets [94] to tackle imprecision in the definition of classes of objects. This concept gave a continuum of grades of membership to classes of objects like “the class of tall men” or “the class of real numbers greater than ten”. Zadeh [93] went on to present the concept of linguistic variables and laid down the basic framework that underlie most of the practical applications of the fuzzy set theory. He also introduced the concept of “if-then rules” to

characterize the working of a fuzzy system [92]. The overall concept is now referred to as the “linguistic approach to describe and solve problems”. One of the recent offshoots of this approach is the “Computational Theory of Perception” or CTP in short [90,91]. This methodology aims at “Computing with Words”.

Although Procyk and Mamdani [20, 137] were the first ones to demonstrate the construction of fuzzy controller, it became the main focus of research in Japan where hundreds of applications were developed, from washing machines to fuzzy subway control system, in the 1990’s.

Humans have capability to perform a very wide variety of physical and mental tasks without any computations. Familiar examples of such tasks are parking a car, driving in heavy traffic, playing cricket and summarizing a story. Underlying this remarkable capability is the brain’s crucial ability to manipulate perceptions. Perceptions can be of distance, size, weight, colour, speed, time, direction, force, number, truth, likelihood and other characteristics of physical objects. Manipulation of perception plays a key role in human decision making. Zadeh argues that we need ways to deal with perception, in addition to the tools that we have for dealing with measurement, to advance in the frontiers of technology beyond where we are today. Especially, in the fields of machine intelligence and automation of decision making processes.

Differences in perceptions give rise to uncertainties [89]. Uncertainty is defined as partial truth by some [89] and lack of complete information by others [25]. When dealing with real world problems one can rarely avoid uncertainty. It is an inseparable part of any measurement. It can even be due to reading errors or imprecision in measuring instruments. With regard to fuzzy systems, it is due to vagueness and ambiguity inherent in natural languages. Even at social level, people create and maintain uncertainty to exercise secrecy or privacy [26]. In engineering terms uncertainty comes from lack of complete information and reflects incompleteness, imprecision, missing information or randomness in data and process. Two important kinds of uncertainties are linguistic and random. The former is associated with words, and the fact that word mean different things to different people, and the later is associated with unpredictability. Probability theory is used to handle random uncertainty and Fuzzy Systems (FSs) are used to handle

linguistic uncertainties. Some times FSs are used to handle both kinds of uncertainties [71].

To handle linguistic uncertainties, Type-2 Fuzzy sets (T2 FS) and their related logic was developed [73]. In such uncertainties it is difficult to determine the exact Membership functions (MF) for a fuzzy system (FS). As an example, suppose the variable of interest is motor speed, denoted by  $x$  where  $x \in [0,100]$  and this gives a speed of 0 to 100 rev/sec. One of the terms that might characterize the amount of perceived speed is ‘slow’. Now if one asks 10 experts to locate the ends of an interval for slow speed on the scale of 0-100, different experts will give different ranges for a particular application in mind. To demonstrate whether uncertainty is associated with words or not, different surveys were made. The results of one such survey conducted by the author, quantizing the range of slowness of motor car speed is tabulated below.

<b>Serial No.</b>	<b>Range for slowness (km/h)</b>
1	10 – 40
2	0 – 40
3	10 – 50
4	10 – 30
5	15 – 40
6	20 – 40
7	10 – 40
8	10 – 30
9	15 – 40
10	20– 40

Table 4.1: Survey results.

The survey clearly reflects the difference in perception of the concept / linguistic variable “slowness of speed” and the uncertainty associated with it. This uncertainty will not only make the membership function creation difficult but will also hamper the controller performance.

According to the data above, the left and right end points of the MF slow (S) are blurred as shown in figure 4.1.

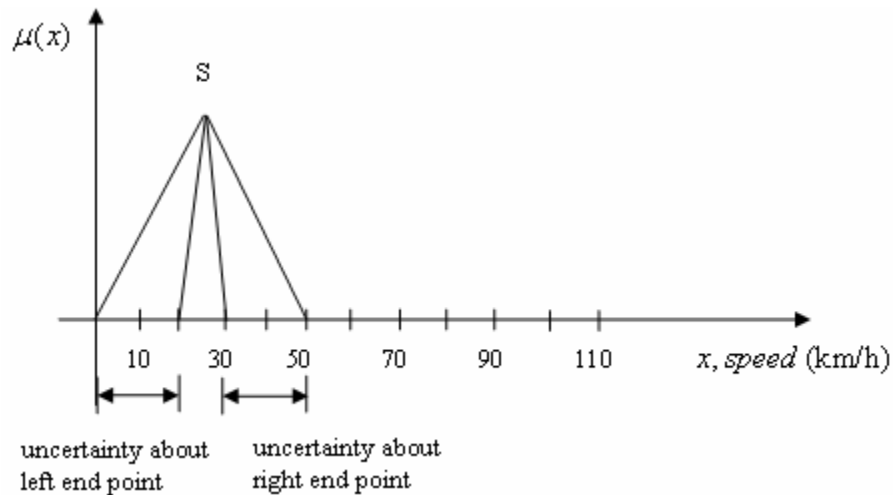


Figure 4.1: Triangular MFs when base end points have uncertainty associated with them.

This region is described as the footprint of uncertainty (FOU) in T-2 FS theory. Somewhere in this FOU are located the lower and upper extremities of our desired MF. There can be  $N$  membership functions in this region of uncertainty. The goal is to find  $MF_{\text{desired}} = MF_1(x)$  OR  $MF_2(x)$ ... OR  $MF_N(x)$

In T-2 FS, each potential MF is assigned a weight, extending the concept into third dimension. This makes representation and computation extremely difficult.

The challenging tasks associated with fuzzy control design has always been to choose appropriate membership functions, minimum rule base and the most suitable fuzzifier and defuzzifier. Having made these choices, the fuzzy controller has to be tuned to deliver the desired response. Multiple simultaneous adjustments (rules, membership functions and gains) make the optimum tuning even more difficult. Many techniques have been used to overcome this difficulty including a phase plane technique for rule base design [62], neural network techniques [29, 74] and gain phase margin analysis technique [76].

Before any rules can be made we need to find the membership functions. Now membership functions, as discussed above, have uncertainties associated with them

which have a trickle down effect on other processes of a fuzzy control system. The research in this chapter proposes to tackle this root cause of uncertainty using a learning approach. The learning approach adjusts these MFs and is also linked with steady state error and overshoot, which are used to specify design requirements.

Almost all fuzzy controllers to date have been made using type-1 fuzzy systems (T-1 FS). However, such fuzzy systems (FSs) have limited capabilities to directly handle data uncertainties [71]. Our approach remains in type-1 but still achieves the purpose for which type-2 was created.

Controllers, be they fuzzy or conventional, are robust when they have some adaptability in them. Most adaptive fuzzy systems use neural networks to incorporate adaptability and are called Adaptive Neuro Fuzzy Inference System (ANFIS) [75]. Such adaptive techniques generally make use of some model of the system or signal that one is trying to predict or control.

We will first discuss basics of fuzzy, very briefly and then present some results gathered from our research on membership function design. These results will form the basics of the adaptive fuzzy controller, named iterative learning fuzzy tuner (ILFT), in this chapter.

## 4.2 Basics of Fuzzy Control

Although the founding father of fuzzy logic [94] initially expected its main applications in economics, medicines, psychology, biology and linguistics, most of the real applications have been developed in engineering system control. A typical block diagram of a fuzzy control system is explained in figure 4.2.

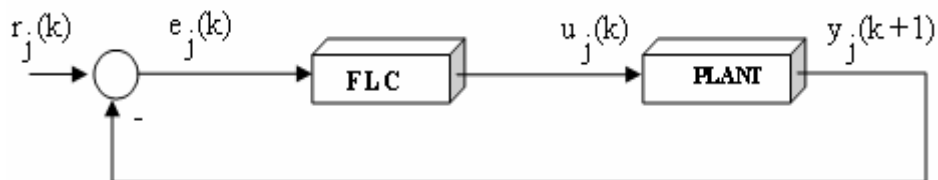


Figure 4.2: Block diagram of a fuzzy control system.



Here  $r_j(k)$  represents the reference signal for  $k=1\dots N$  and  $j=1\dots\infty$ . Variable  $j$  represents the iteration number and  $k$  represents the samples. The error is represented by  $e_j(k)$ , the input to the plant is  $u_j(k)$  and the next plant output is  $y_j(k+1)$ . Using the error the FLC produces the desired input to the plant.

It is recommended to normalize the universe of discourse of input and output variables. Universe of discourse basically determines the applicable range for a characteristic variable in the context of the system designed. Because of this normalization, it is also recommend using the input and output scaling factors so as to adjust input from sensors and output to actuators. The modified block diagram of the fuzzy logic controller is described in figure 4.3 below.

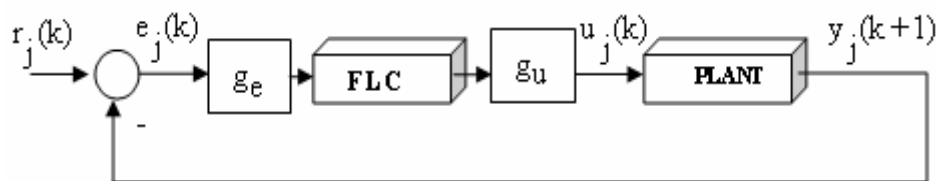


Figure 4.3: Modified block diagram of a fuzzy control system.

Factors  $g_e$  and  $g_u$  are the input and output scaling factors. The Fuzzy Logic Controller (FLC) adjusts the input to the plant. This input is multiplied by the output scaling factor before being applied to the plant. The output of the FLC is dependent on the choice of all four blocks of a fuzzy controller. These four blocks are shown in figure 4.4 below

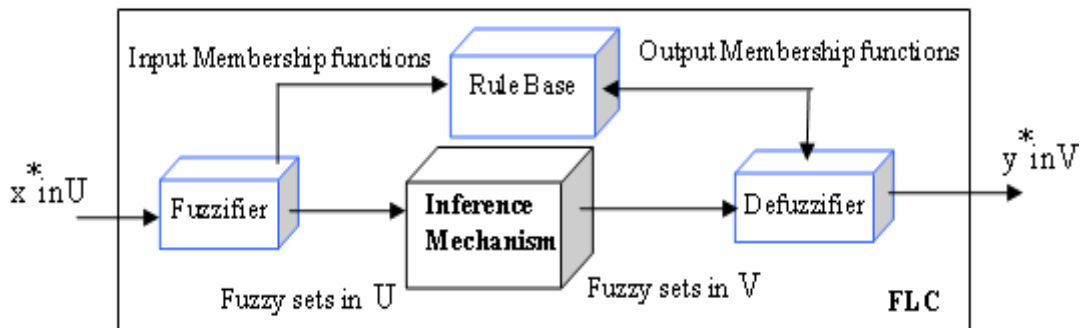


Figure 4.4: A typical structure of a fuzzy controller.

Here  $U$  and  $V$  are the universes of discourse for input and output membership functions. Universe of discourse is the  $n$ -dimensional Euclidean space  $R^n$ . Fuzzifier is defined as a mapping from a real valued point  $x^* \in U$  to fuzzy set  $A'$  in  $U$  [103]. The input to the fuzzifier is crisp. Typically a fuzzifier could be a Singleton fuzzifier, a Gaussian fuzzifier or a Triangular fuzzifier. Singleton fuzzifier is represented as

$$\mu_{A'}(x) = \begin{cases} 1 & \text{if } x = x^* \\ 0 & \text{otherwise} \end{cases}$$

In a fuzzy inference engine, fuzzy logic principles are used to combine the fuzzy IF-THEN rules in the fuzzy rule base into a mapping from a fuzzy set  $A'$  in  $U$  to a fuzzy set  $B'$  in  $V$ . There are many choices of inference engines. Some of the most popular ones are product, minimum, Lukasiewicz, Zadeh and Dienes-Rescher inference engines.

Fuzzy rule base is the heart and soul of the fuzzy system. It contains rules of the form

IF  $x_1$  is  $A_1$  and...and  $x_n$  is  $A_n$  THEN  $y$  is  $B$

Where  $x_1, \dots, x_n$  are linguistic variables,  $A_1 \dots A_n$  and  $B$  are fuzzy sets.

Defuzzifier is defined as mapping from fuzzy set  $B'$  in  $V \subset R$  (which is the output of the fuzzy inference engine) to crisp point  $y^* \in V$ . The defuzzifier output is a crisp value. Three of the most popular defuzzifiers are centre of gravity, centre average and maximum defuzzifier.

### **4.3 Supporting Work**

In order to find out the effect of linguistic uncertainties on Fuzzy controller and because of it on membership functions, scaling factors, rule base, fuzzifier, defuzzifier and on inference engine performances, a lot of research was done. This research comprised simulations. These simulations were run for different systems and with different combinations of the above mentioned factors. The data was analyzed and links between required performance criteria like steady state error and percentage overshoot

were established. Some of the results achieved using a second order cruise control system with input as force in Newtons (N) and output as velocity in m/sec are discussed below. The system transfer function was given by

$$\frac{Y(s)}{U(s)} = \frac{75}{s^2 + 20s + 75} \quad (4.1)$$

Simulations were performed with different number of input and output membership functions using both Mamdani and Sugeno type rule processing. In Mamdani type rule processing both input and output of the fuzzy system are fuzzy sets (i.e., words in natural languages). This creates problem while designing engineering systems. In Sugeno type rule processing the consequent part of the rule is a mathematical function of the input variables. This helps designers represent real-valued variables. The results of some of the simulations, performed on system of equation (4.1) are discussed below. One such simulation involved the use of 3 input MFs for error and 3 output MFs for velocity as shown in figure 4.5.

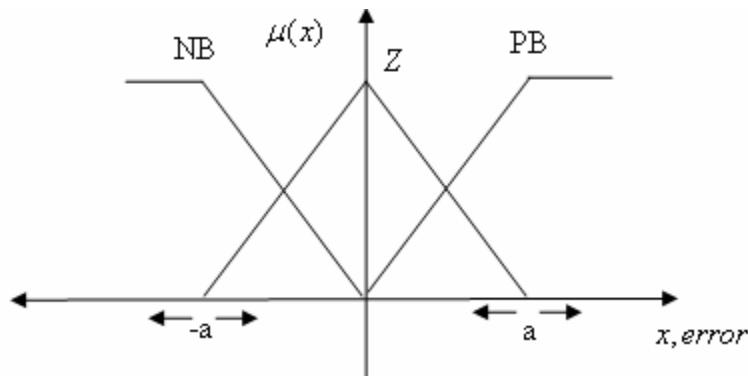


Figure 4.5: Input Membership functions.

Here NB stands for Negative Big, Z stands for Zero and PB for Positive Big. Degree of the input (force) membership function, is represented by  $\mu(x)$ . The left and right end points of the membership function are represented by ‘-a’ and ‘a’. The output membership function is presented in figure 4.6.

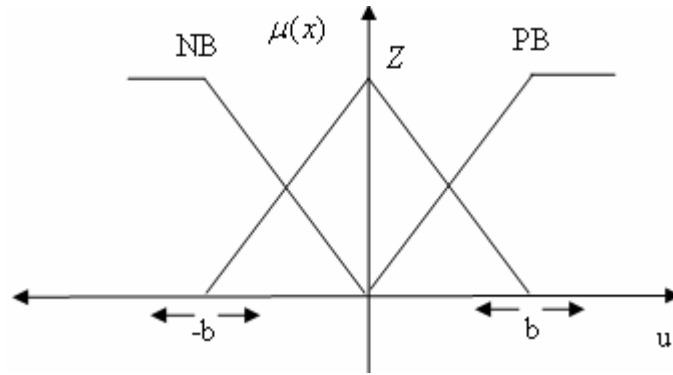


Figure 4.6: Output Membership functions for Mamdani rule processing.

Here  $-b$  and  $b$  are the left and right end points of the output (velocity) membership function. This output serves as an input to the plant. Research has shown that Sugeno type rule processing works better for the proposed scheme. The output membership functions for the Sugeno type rule processing is exhibited in figure 4.7.

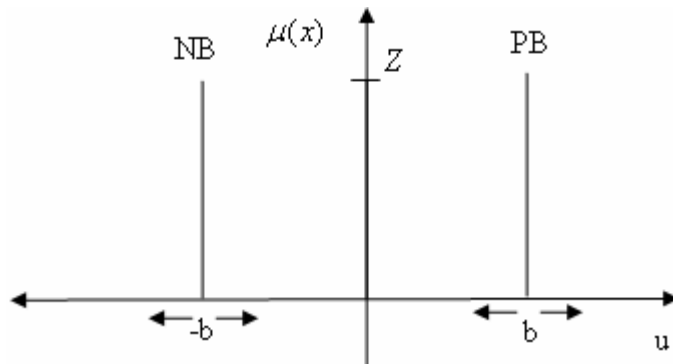


Figure 4.7: Output Membership functions for Sugeno rule processing.

For all these membership functions, the end points are to be adaptively adjusted.

Response of the second order cruise control system ( $G_2(z)$ ) against a step input with different values of 'a' are given in figure 4.8.

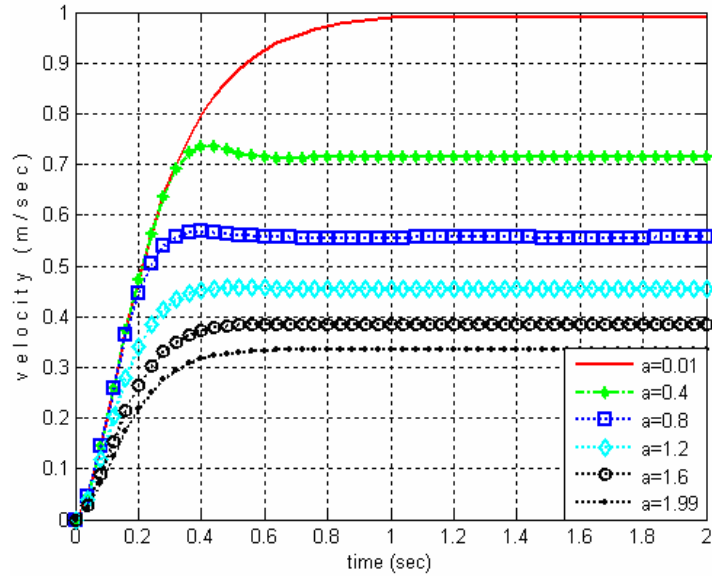


Figure 4.8: Response with different values of ‘ a ’.

Clearly the response is different for different membership function widths and it increases with decreasing ‘ a ’. The control surfaces (input-output plot) with these membership functions are shown in figure 4.9.

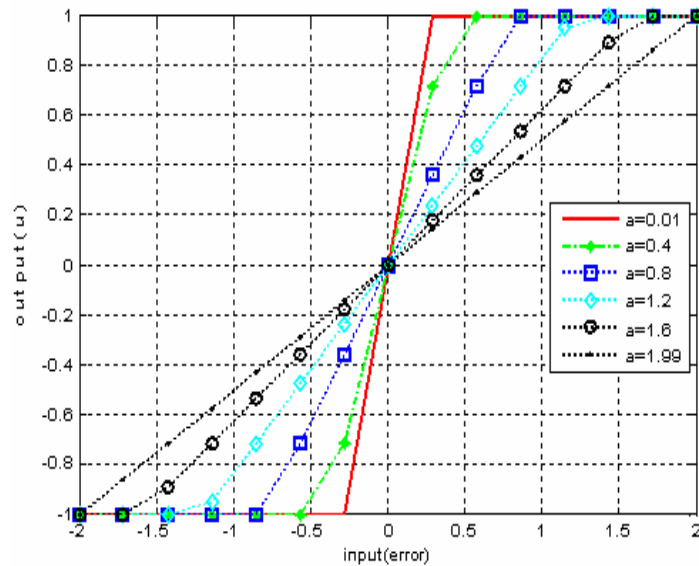


Figure 4.9: Control surfaces with different values of ‘ a ’.

The slope of the control surface is increasing with decrease in value of ‘ $a$ ’. These control surfaces highlight the fact that Fuzzy Controller (FC) response can be varied with varying membership function widths.

Response of the second order cruise control system for different output membership function widths and the corresponding control surface plots are shown in figure 4.10 and 4.11.

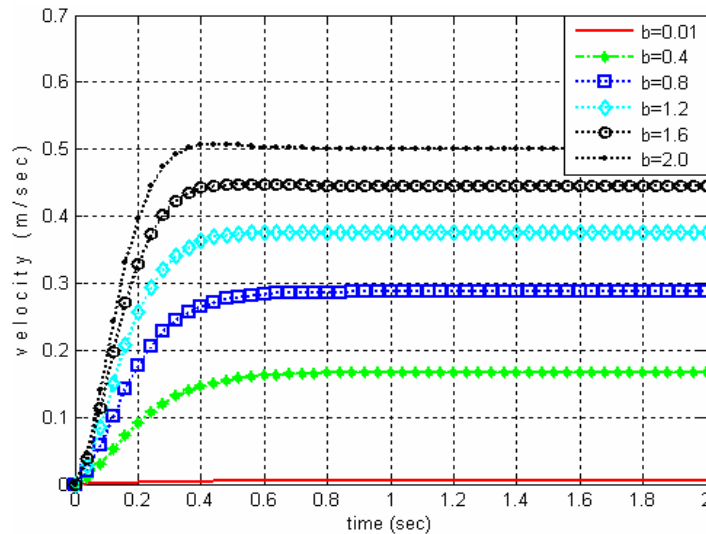


Figure 4.10: Response with different values of ‘ $b$ ’.

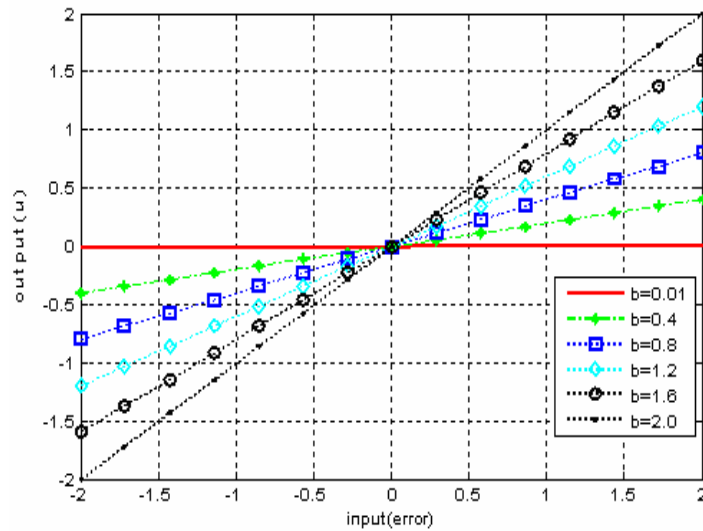


Figure 4.11: Control surfaces with different values of ‘ $b$ ’.

Response shows that a decrease in output membership function width or reducing the value of end point ‘b’, decreases the output of the system. Also, reducing the value of ‘b’ reduces the slope of the control surface.

Simulations were also run to find the effect of  $g_e$  and  $g_u$  on the response. This effect was recorded keeping the input and output membership functions fixed. The effect of  $g_u$  on system response is shown in figure 4.12. The input and output membership function end point values for this result were,  $a = 1.0$  and  $b = 1.0$ .

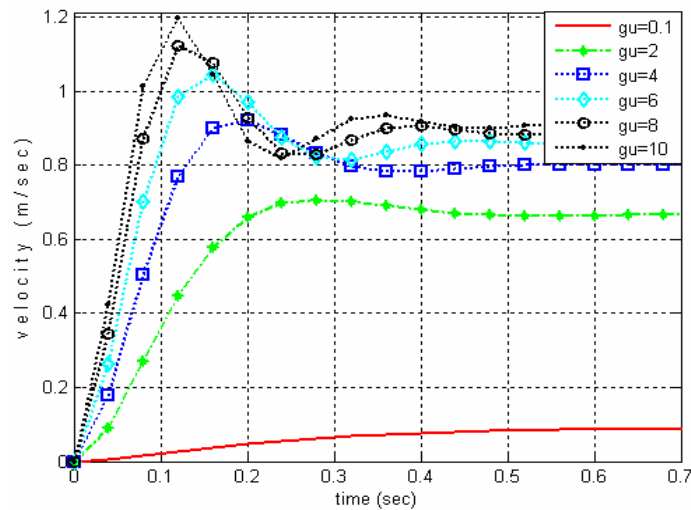


Figure 4.12: Systems response with different values of  $g_u$ .

The response shows  $g_u$ 's effect on damping and hence over-shoot in the system's response.

In a fuzzy controller there are different parameters to play with, like  $g_e$ ,  $g_u$ ,  $a$ ,  $b$ , number of input and output membership functions and the number of rules. The research aimed at finding a link or links between these parameters and the steady state error, percentage overshoot and the peak time of the response. Lots of simulations and tabulated results were analyzed and some conclusions were derived. Some of the results are discussed below.

Table 4.2 presents the data showing values of steady state error, percentage overshoot and peak time against a changing value of ‘a’, while keeping all other parameters constant.

<b>Input Membership function end point ‘a’</b>	<b>Steady state Error (sse)</b>	<b>% sse</b>	<b>% over shoot</b>	<b>Peak time (sec)</b>
0.000000001	0.000000266	0.0000266	-0.0000000998	4.22
0.00001	0.0000136	0.00136	-0.000995	3.9
0.0001	0.000107	0.0107	-0.0099	1.92
0.001	0.0010	0.1	-0.0980	1.469
0.01	0.0099	0.99	-0.94	1.02
0.02	0.0196	1.96	-1.83	0.89
0.03	0.0291	2.91	-2.69	0.81
0.04	0.0385	3.8	-3.54	0.76
0.05	0.0476	4.7	-4.35	0.72
0.06	0.0566	5.6	-5.16	0.69
0.07	0.0654	6.5	-5.94	0.66
0.08	0.0741	7.4	-6.71	0.64
0.125	0.1111	11.11	-10.0	0.57
0.25	0.2000	20	-18.1	0.47
0.375	0.2727	27	-25.1	0.42
0.5	0.3333	33	-31.2	0.39
0.625	0.3846	38	-36.7	0.38
0.75	0.4286	42	-41.5	0.39
0.875	0.4667	46	-45.7	0.41
1	0.5	50	-49.3	0.44

Table 4.2: Establishing a link between steady state error, over shoot and peak time with ‘a’.



This table hints at the movement of steady state response upwards and the movement of peak overshoot down wards as the ‘ a ’ parameters is decreased.

The data showing the effect of changing ‘ b ’ while keeping other parameters constant is tabulated in table 4.3.

<b>Output Membership function end point ‘b’</b>	<b>Steady state Error</b>	<b>% steady state error</b>	<b>% over shoot</b>	<b>Peak time</b>
0.1	0.5000	50	-50.00	5.00
0.2	0.2000	20	-18.14	0.47
0.3	0.1429	14.29	-8.50	0.30
0.4	0.1111	11.11	-1.83	0.239
0.5	0.0909	9.09	3.1868	0.204
0.6	0.0769	7.69	7.1604	0.180
0.7	0.0667	6.67	10.4233	0.163
0.8	0.0588	5.88	13.1750	0.150
0.9	0.0526	5.26	15.5355	0.140
1.0	0.0476	4.76	17.5989	0.131

Table 4.3: Establishing a link between steady state error, over shoot and peak time with ‘ b ’.

The table shows that the steady state response moves downwards as the parameter b is decreased.

Simulations were also run on different systems to figure out which portion of normalized error has more effect on steady state error, percentage over shoot and peak time. Based on the results obtained from these simulations a mathematical frame work for the scheme was developed which is discussed next.

#### 4.4 Iterative Learning Fuzzy Tuner (ILFT)

Fuzzy controller designers are required to make a number of choices; choice about the structure of the controller i.e. how many inputs and outputs the controller will have, choice about the shape of membership functions i.e. triangular, gaussian etc., choice about the fuzzifier, rule processing, inference mechanism and defuzzification method. But the most important choice is the choice of membership function end points which can handle all uncertainties associated with fuzzy design.

Research results have shown that singleton fuzzifier, triangular input membership functions, centre average defuzzifier and product inference engine works best for the proposed controller. The block diagram of the complete controller is indicated in figure 4.13.

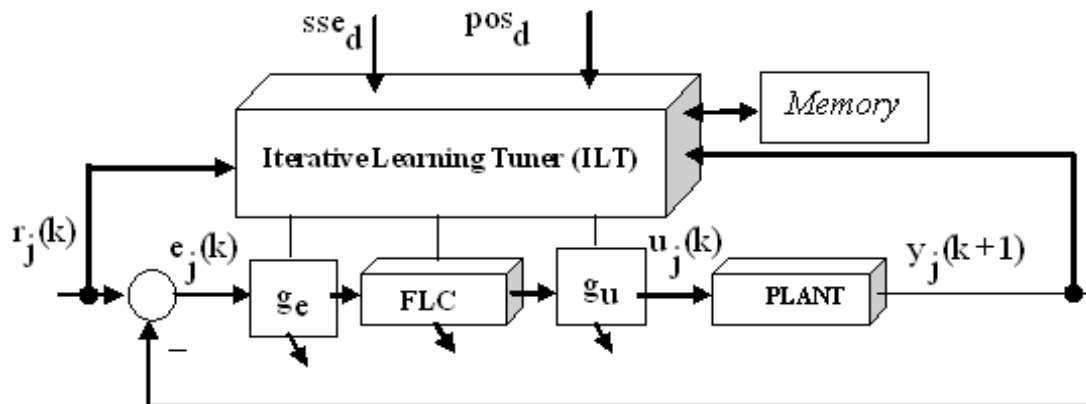


Figure 4.13: Block diagram of the proposed controller.

Here  $sse_d$  is the desired steady state error,  $pos_d$  is the desired percentage over shoot,  $r_j(k)$ ,  $e_j(k)$ ,  $u_j(k)$  and  $y_j(k+1)$  are the reference input, error, input to the plant and next plant output at iteration  $j$ . Factors  $g_e$  and  $g_u$  are the input and output scaling factors. Desired steady state error and percentage over shoot are supplied to Iterative Learning Tuner (ILT). The ILT adjusts  $g_e$ ,  $g_u$  and tunes the fuzzy logic controller (FLC) by adaptively adjusting the membership function end points. The aim is to remove

uncertainty associated with linguistic variables and to converge with respect to given steady state error and percentage overshoot. The learnt values of  $g_e$ ,  $g_u$  and membership function end points are stored in memory to be used in future iterations.

Taking a Sugeno type rule processing the input member ship functions proposed are of the form given in figure 4.14 below.

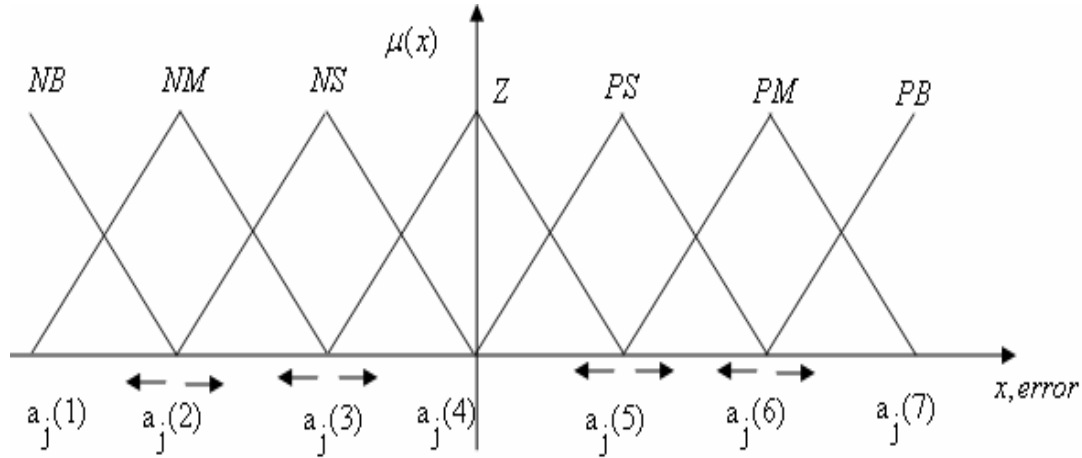


Figure 4.14: Proposed Input Membership functions.

In figure 4.14 seven MFs are defined for error. End points of membership functions  $a_j(k)$ , where  $k=1\dots n$ , are to be adjusted by the ILT block. Here  $n$  is the number of member ship functions.

A Takagi-Sugeno-Kang (TSK) [54, 144] fuzzy system is constructed from the following rules

$$\text{IF } x_1 \text{ is } A_1^l \text{ and } \dots \text{ and } x_n \text{ is } A_n^l, \text{ THEN } y^l = c_0^l + c_1^l x_1 + \dots + c_n^l x_n \quad (4.2)$$

Where  $A_i^l$  are fuzzy sets in  $U_i \subset \mathbb{R}$ ,  $c_i^l$  are constants,  $l=1,2,\dots,M$  and  $i=1,2,\dots,n$ .

Number of rules in the fuzzy rule base is denoted by  $M$  and  $n$  is the number of membership functions. The IF parts of the rules are the same as in the ordinary fuzzy IF-THEN rules, but the THEN parts are linear combinations of the input variables. Here  $\mathbf{x} = (x_1, \dots, x_n)^T \in U \subset \mathbb{R}^n$  and  $y \in V$  are input and output (linguistic) variables of the fuzzy system, respectively. Given an input  $\mathbf{x}$ , the output  $f(\mathbf{x}) \in V \subset \mathbb{R}$  of the TSK fuzzy

system, with product inference engine, singleton fuzzifier and centre average defuzzifier, is computed as the weighted average of the  $y^l$ 's in (4.2), that is

$$f(x) = \frac{\sum_{l=1}^M y^l w^l}{\sum_{l=1}^M w^l} \quad (4.3)$$

where the weights  $w^l$  are computed as

$$w^l = \prod_{i=1}^n \mu_{A_i^l}(x_i) \quad (4.4)$$

Equation (4.3) and (4.4) gives

$$f(x) = \frac{\sum_{l=1}^M y^l \left( \prod_{i=1}^n \mu_{A_i^l}(x_i) \right)}{\sum_{l=1}^M \left( \prod_{i=1}^n \mu_{A_i^l}(x_i) \right)} \quad (4.5)$$

The output of the fuzzy controller is dependent on the input and output membership functions. To see the behaviour of this TSK fuzzy system the following derivation is made.

Let us take 2 membership functions PS(positive small) and PM(positive medium) for the linguistic variable error (e) as shown in figure 4.15 below.

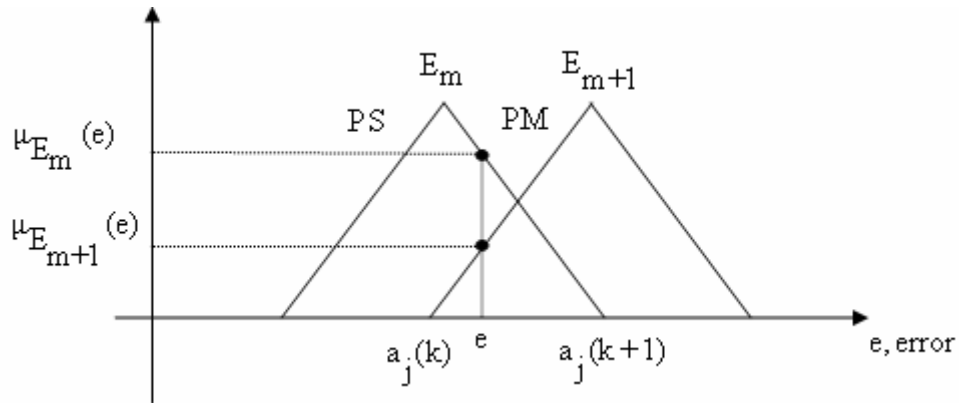


Figure 4.15: PS and PM membership function from figure 4.14.

The membership functions are triangular and their centres are marked as  $E_m$  and  $E_{m+1}$ . The end points are adaptive (fuzzy) and two of the end points are labelled  $a_j(k)$  and

$a_j(k+1)$ . Here  $j$  refers to current iteration. An error value at current iteration is given as  $e$ .

$$\text{Slope of line } \overline{E_m a_j(k+1)} = \frac{-1}{a_j(k+1) - a_j(k)}$$

$$\text{Slope of line } \overline{\mu_{E_m}(e) a_j(k+1)} = \frac{-\mu_{E_m}(e)}{a_j(k+1) - e}$$

$$\text{Slope of line } \overline{a_j(k) E_{m+1}} = \frac{1}{a_j(k+1) - a_j(k)} \text{ and}$$

$$\text{Slope of line } \overline{e E_{(m+1,j)}} = \frac{1 - \mu_{E_{m+1}}}{a_j(k+1) - e}$$

Using the fact that

$$\text{Slope of line } \overline{E_m a_j(k+1)} = \text{Slope of line } \overline{\mu_{E_m}(e) a_j(k+1)} \text{ and}$$

$$\text{Slope of line } \overline{a_j(k) E_{m+1}} = \text{Slope of line } \overline{e E_{(m+1,j)}}, \text{ it can be shown that}$$

$$\mu_{E_m}(e) = \frac{a_j(k+1) - e}{a_j(k+1) - a_j(k)} \quad (4.6)$$

$$\mu_{E_{m+1}}(e) = \frac{e - a_j(k)}{a_j(k+1) - a_j(k)} \quad (4.7)$$

Taking the centre values of the output membership functions as  $b_j(k)$ . As the approach in this paper does not modify output membership functions,  $b_j(k)$  can be written as  $b(k)$ . Using this definition of output membership function and the weighted average defuzzifier, equation (4.5) can be written as

$$f(x_j) = \frac{\mu_{E_m}(e)b(k) + \mu_{E_{m+1}}(e)b(k+1)}{\mu_{E_m}(e) + \mu_{E_{m+1}}(e)} \quad (4.8)$$

Here  $f(x_j)$  is the output of the fuzzy controller at iteration  $j$  with error value  $e$ . Putting equation (4.6) and (4.7) in (4.8) gives

$$f(x_j) = \frac{\left( \frac{a_j(k+1) - e}{a_j(k+1) - a_j(k)} \right) b(k) + \left( \frac{e - a_j(k)}{a_j(k+1) - a_j(k)} \right) b(k+1)}{\left( \frac{a_j(k+1) - e}{a_j(k+1) - a_j(k)} \right) + \left( \frac{e - a_j(k)}{a_j(k+1) - a_j(k)} \right)}$$

$$f(x_j) = \frac{a_j(k+1)b(k) - eb(k) + eb(k+1) - a_j(k)b(k+1)}{a_j(k+1) - e + e - a_j(k)}$$

$$f(x_j) = \frac{b(k+1) - b(k)}{a_j(k+1) - a_j(k)} e + \frac{a_j(k+1)b(k) - a_j(k)b(k+1)}{a_j(k+1) - a_j(k)} \quad (4.9)$$

The end points  $a_j(k)$  will take on different values as different pair of MFs are considered.

Equation (4.9) shows that the fuzzy controller output is dependent both on the end points of the input and output member ship functions. A decrease in width of the input member ship function (i.e. change in  $a_j(k+1) - a_j(k)$ ) will result in an increase in the fuzzy controller output and vice versa. This gives us the capability of increasing and decreasing the effective gain at different intervals of the control surface. Similar results could be achieved by changing the difference,  $b(k+1) - b(k)$ , as can be achieved by changing the difference,  $a_j(k+1) - a_j(k)$ .

For this scheme, proposal is made to keep  $a_j(1)$ ,  $a_j(4)$  and  $a_j(7)$  fixed at -1,0 and 1 respectively. Only modifications in  $a_j(5)$  and  $a_j(6)$  are done. The values or ranges of the input MF end points are summarized in table 4.4.

$a_j(1) = -1$	$a_j(2) = -a_j(6)$	$a_j(3) = -a_j(5)$	$a_j(4) = 0$
$a_j(5) = [a_j(4), a_j(6)]$	$a_j(6) = [a_j(5), a_j(7)]$	$a_j(7) = 1$	

Table 4.4: Input membership end point values.

This scheme proposes to keep the output membership function end points fixed. The reason is that similar responses can be achieved with either input or output membership function end point manipulations.

Research has also shown that it is better to divide the region evenly so that there is equal space to move the end points in either direction. One group of suggested output membership function end points are shown in table 4.5.

$b(1) = -1$	$b(2) = -0.6$	$b(3) = -0.3$	$b(4) = 0$
$b(5) = 0.3$	$b(6) = 0.6$	$b(7) = 1$	

Table 4.5: Output membership end point values.

An example is given to illustrate the effect of input MF end point values on control surface.

**Example 4.1:**

Taking  $a_j(5) = 0.4$  and  $a_j(6) = 0.5$  for the MFs Z and PS. The MFs are shown in figure 4.16.

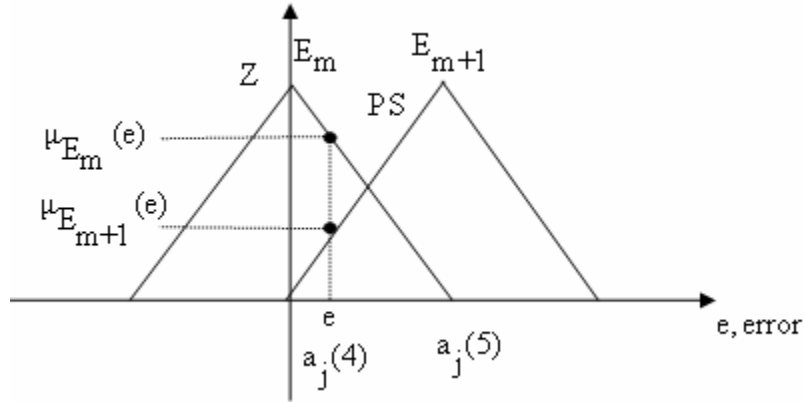


Figure 4.16: Z and PS membership functions from figure 4.14.

For  $k = 4$ , equation (4.9)  $\Rightarrow$

$$f(x_j) = \frac{b(5) - b(4)}{a_j(5) - a_j(4)} e + \frac{a_j(5)b(4) - a_j(4)b(5)}{a_j(5) - a_j(4)}$$

$$f(x_j) = \frac{0.3 - 0}{0.4 - 0} e + \frac{0.4(0) - 0(0.3)}{0.4 - 0}$$

$$f(x_j) = 0.75e \tag{4.10}$$

This equation is effective for a range of error,  $e = [0, 0.4]$ . Similarly for different values of  $k$  different fuzzy controller output equations were obtained. Here  $k$  represents, number of membership function end points. This gives different equations for different ranges of errors. The results for some of the values of  $k$  are tabulated in table 4.6 below.

	<b>k = 4</b> <b>e = [0, 0.4]</b>	<b>k = 5</b> <b>e = [0.4, 0.5]</b>	<b>k = 6</b> <b>e = [0.5, 1]</b>
$f(x_j) =$	$0.75e$	$3e - 0.9$	$0.8e + 0.2$

Table 4.6: Effective control surface equations for different ranges of error.

A plot of error vs.  $f(x_j)$  is described in figure 4.17.



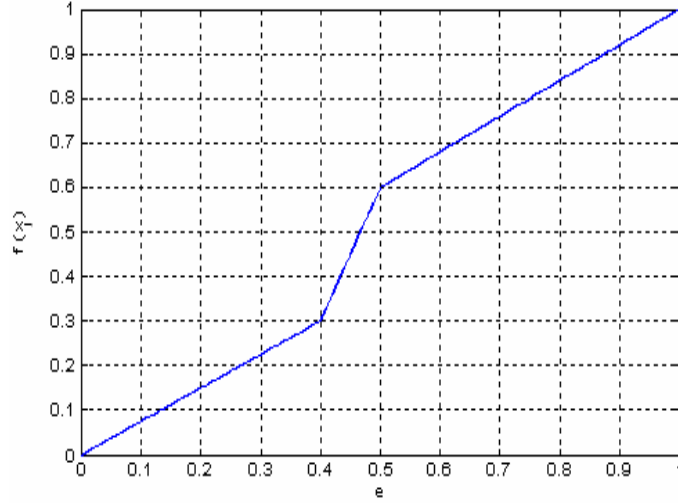


Figure 4.17: Control surface plot for the data in table 4.6.

Similar plots can be constructed for negative values of error. The above discussion shows that one can make three linear input-output regions for different ranges of error for positive error values. Similarly three input-output regions can be made for negative values of error. Hence the proposed control surface can be taken as a piece-wise-linear control surface.

Following the results from the research we propose the following procedure for the up gradation of  $g_u$ ,  $a_j(5)$  and  $a_j(6)$ . This procedure mixes iterative learning with fuzzy to achieve our design objectives.

1- Find  $g_u$  such that the system is critically damped using the learning law

$$g_{u_{j+1}}(k) = g_{u_j}(k) \pm \mu_{gu} (\|r_j(k)\| - \|y_j(k)\|) \quad (4.11)$$

Here  $\mu_{gu}$  is the step size parameter for finding  $g_u$  that will make the system critically damped.

2- Change the value of  $a_j(5)$  such that  $sse \leq sse_d$  using the learning law

$$a_{j+1}(5) = a_j(5) \pm \mu_{sse} (\|r_j(k)\| - \|y_j(k)\|) \quad (4.12)$$

Here  $\mu_{sse}$  is the step size parameter for correcting steady state error.

3- Change the value of  $a_j(6)$  such that  $pos \leq pos_d$  using the learning law

$$a_{j+1}(6) = a_j(6) \pm \mu_{pos} (\|r_j(k)\| - \|y_j(k)\|) \quad (4.13)$$

Here  $\mu_{pos}$  is the step size parameter for correcting percentage over shoot.

The procedure is summarized using a flow diagram in figure 4.18.

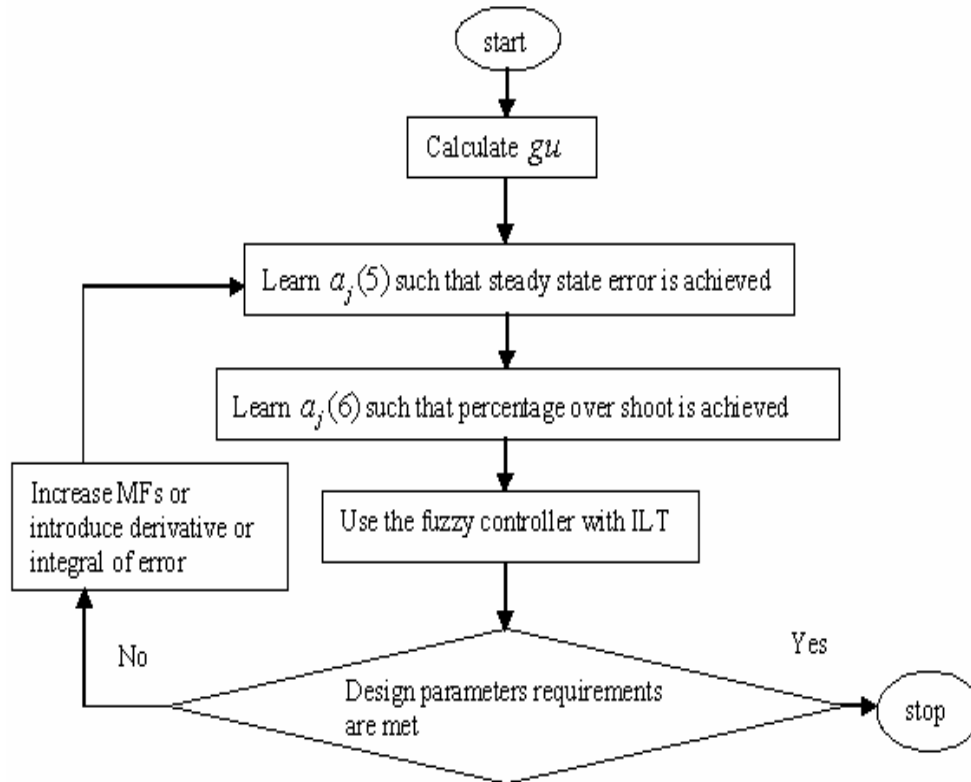


Figure 4.18: Flow chart for fast design using ILT.

The flow chart suggests to calculate  $g_u$ ,  $a_j(5)$  and  $a_j(6)$  in respective order. In some rare cases when the design requirements are not met, we need to increase MFs or introduce derivative or integral of error. This situation is discussed in more detail in later sections.

#### 4.4.1 Simulations and results

The scheme presented in this chapter was tested through simulations. One of the simulations involved a model of DC motor assuming rigid rotor and shaft. The motor was of the form given in Appendix A, but with the following parameters. The parameters were especially chosen to increase the difficulty in controller design.

- $J=0.01 \text{ kg.m}^2/\text{s}^2$  ;     % moment of inertia of the motor  
 $b=0.1 \text{ Nms}$  ;             % damping ratio of the mechanical system  
 $K=0.05 \text{ Nm/Amp}$  ;     % electromotive force constant  
 $R=1 \text{ ohm}$  ;             % electric resistance  
 $L=0.5 \text{ H}$  ;             % electric inductance

The transfer function of the motor is given as

$$G(s) = \frac{0.05}{0.005s^2 + 0.06s + 0.1025} \quad (4.14)$$

With a sampling time of  $T_s=0.01$  the system of equation (4.14), in difference equation form is given by

$$y(k+1) - 1.885y(k) + 0.8869y(k-1) = 0.0004805u(k) + 0.0004617u(k-1) \quad (4.15)$$

The aim was to design a speed controller which should endure less than 5% steady state error (sse) and have less than 5% over shoot (pos). The required speed is 1 radian/second.

The open loop unit step response of the system is described in figure 4.19.

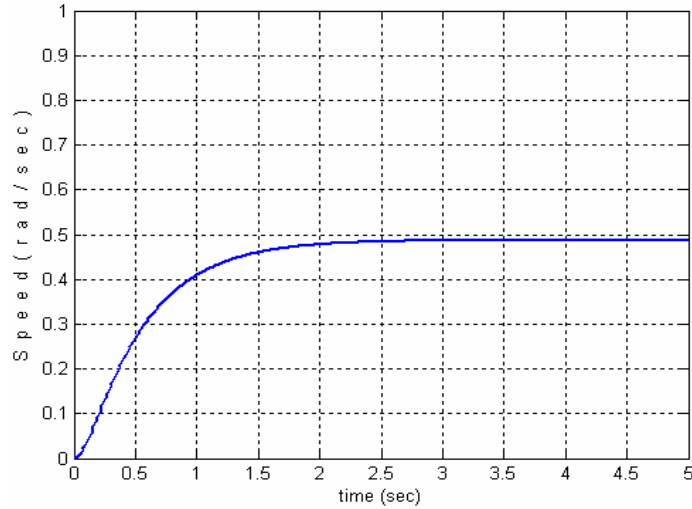


Figure 4.19: Open loop step response of the DC motor system.

The figure shows damped response. The system gain needs to be improved in order to achieve the desired speed.

#### 4.4.1.1 Conventional Proportional Controller

In order to gauge the performance of the proposed controller, conventional proportional controller was also designed for the system. A feed back control system is represented by a block diagram given in figure 4.20.

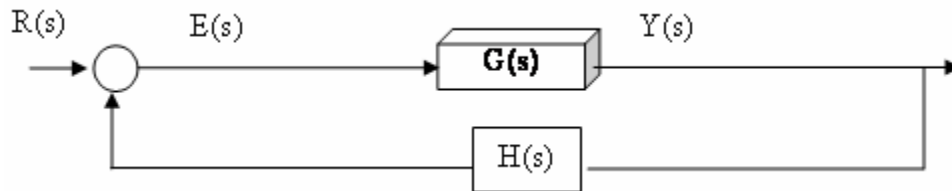


Figure 4.20: Block diagram of a feed back control system.

Here  $G(s)$  represents the plant and  $H(s)$  the feedback system. The over all transfer function of the feed back control system in figure (4.20) with unity feed back  $H(s) = 1$ , is given by

$$\frac{Y(s)}{R(s)} = \frac{G(s)}{1+G(s)} \quad (4.16)$$

A block diagram of a conventional proportional controller with unity feedback is given in figure 4.21.

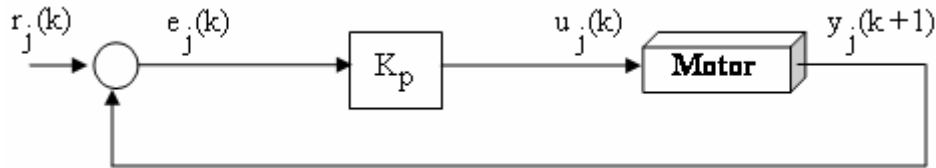


Figure 4.21: A conventional proportional controller.

Here  $K_p$  is the proportional gain. The error,  $e_j(k)$  which is the difference between the actual output of the system,  $y_j(k+1)$  and the reference signal  $r_j(k)$ . This error is multiplied with gain to produce the input,  $u_j(k)$  to the Motor.

The aim is to design a conventional proportional controller with a steady state error ( $e_{ss}$ ) of less than 5% and percentage overshoot (pos) of less than 5%. The steady state error to a unit step input is given by the following relationship.

$$e_{ss} = \frac{1}{1+K_p G(0)} \quad (4.17)$$

Where  $G(0)$  is the DC gain of the plant. The motor transfer function in equation (4.14) can also be written as

$$G(s) = \frac{10}{s^2 + 12s + 20.5} \quad \text{or} \quad (4.18)$$

$$G(s) = \frac{10}{(s+9.9)(s+2.0)}$$

Applying final value theorem on (4.18) and multiplying with  $K_p \Rightarrow$

$$K_p G(0) = K_p \lim_{s \rightarrow 0} G(s) = \lim_{s \rightarrow 0} \frac{10K_p e}{(s+9.9)(s+2.0)} \cong 0.5K_p$$

$$K_p G(0) \cong 0.5K_p \quad (4.19)$$

Putting this value in (4.17) for a required steady state error of 5%  $\Rightarrow$

$$e_{ss} = \frac{1}{1 + K_p G(0)} < 0.05 \quad (4.20)$$

$\Rightarrow$

$$1 + 0.5K_p \geq 20 \quad (4.21)$$

To meet the steady state requirement, proportional gain should be,  $K_p \geq 38$ .

Taking  $K_p = 39$  and using it in the standard feedback system equation with unity feed back, equation (4.14), becomes

$$\begin{aligned} \frac{Y(s)}{R(s)} &= \frac{K_p G(s)}{1 + K_p G(s)} = \frac{\frac{39(10)}{s^2 + 12s + 20.5}}{1 + \frac{39(10)}{s^2 + 12s + 20.5}} = \frac{390}{s^2 + 12s + 410.5} \\ \frac{Y(s)}{R(s)} &= 0.95 \frac{(20.26)^2}{s^2 + 2(0.296 * 20.26)s + (20.26)^2} \end{aligned} \quad (4.22)$$

Giving natural frequency and damping ratio values of

$$w_n \cong 20.26 \text{ and } \zeta \cong 0.296$$

The percentage overshoot at these values is

$$\text{Percentage overshoot is } = 100e^{-\pi\left(\frac{\zeta}{\sqrt{1-\zeta^2}}\right)} \cong 37.7\% \quad (4.23)$$

This is much higher than the required value. For higher values of  $K_p$  the overshoot is even higher. For a percentage overshoot of less than 5%,  $K_p$  has to be  $\leq 5.5$ .

For this system, it is impossible to design a conventional Proportional controller which can meet both design requirements. The best result w.r.t. steady state error, is achieved with  $K_p = 39$ . With that value the response to a step input is shown in figure 4.22.

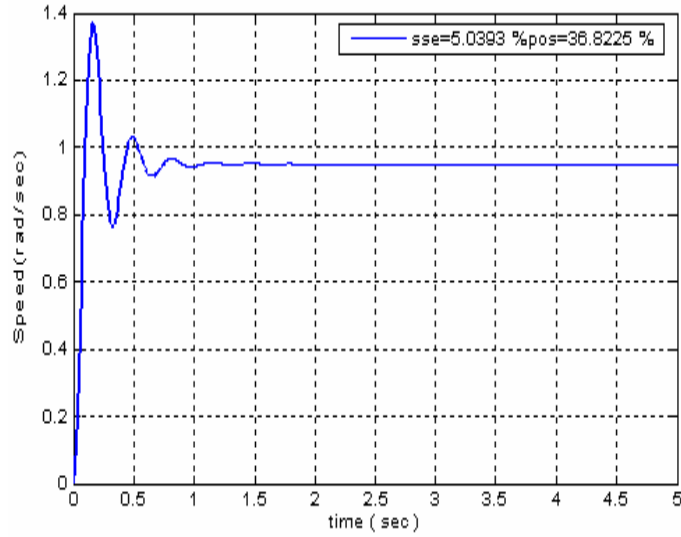


Figure 4.22: Step response using a conventional proportional controller with  $K_p = 39$ .

The response shows a high overshoot while the steady state error is near the required limit. The input-output mapping of the proportional controller with  $K_p = 39$  is shown in figure 4.23 below.

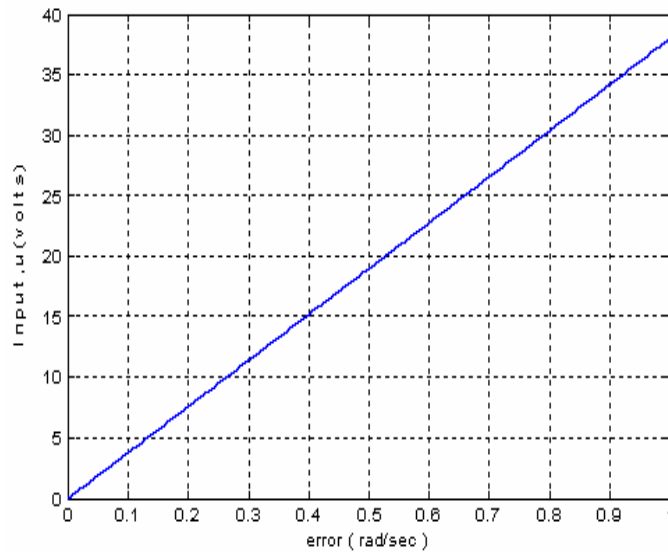


Figure 4.23: Input-output plot with  $K_p = 39$ .

As expected the proportional controller produced a linear response.

### 4.4.1.2 Iterative Learning Fuzzy Tuner

Some simulation results are discussed in this section.

#### 4.4.1.2.1 DC Motor

Using the same system as given in equation (4.14), a controller was designed to achieve same design requirements as in section 4.4.1.1. The universe of discourse for all linguistic variables was normalized between  $[-1,1]$  i.e.  $U = [-1,1]$ . The membership function end points and their permissible ranges were taken from table 4.4 and 4.5.

The proposed scheme adjusts  $a_j(5)$  and  $a_j(6)$  to tackle linguistic uncertainty and also helps in meeting the steady state and percentage overshoot requirements. The scaling factors  $g_u$  and  $g_e$  are also adjusted, if required. Though  $a_j(5)$  and  $a_j(6)$  can take up any initial starting value within their permissible range, it is recommended, without any loss of generality to start with values that divide the Universe of discourse evenly. For this simulation, values of  $a_j(5) = 0.3$  and  $a_j(6) = 0.6$  were chosen which divides the range reasonably, though any other values could have also been chosen. The values of  $g_e$  and  $g_u$  are given a starting value of 1, suggesting that there is no initial input and output scaling. In brief, the initial values chosen were

$$a_j(5) = 0.3, a_j(6) = 0.6, g_e = 1.0 \text{ and } g_u = 1.0$$

A set of 7 Membership functions for input variable (error) for FLC, of the form in figure 4.14, was used. The rule base of the fuzzy controller is shown in table 4.7.



	<b>NB</b>	<b>NM</b>	<b>NS</b>	<b>Z</b>	<b>PS</b>	<b>PM</b>	<b>PB</b>
$e_j(k)$	-1	$a_j(2)$	$a_j(3)$	0	$a_j(5)$	$a_j(6)$	1
$u_j(k)$	-1	-0.6	-0.3	0	0.3	0.6	1

Table 4.7: Rule base for proposed scheme.

After learning  $g_u$  using equation (4.11) with a step size value of  $\mu_{gu} = 0.1$ , the behaviour of percentage over shoot (pos) is presented in figure 4.24. It took 19 iterations to learn  $g_u$ .

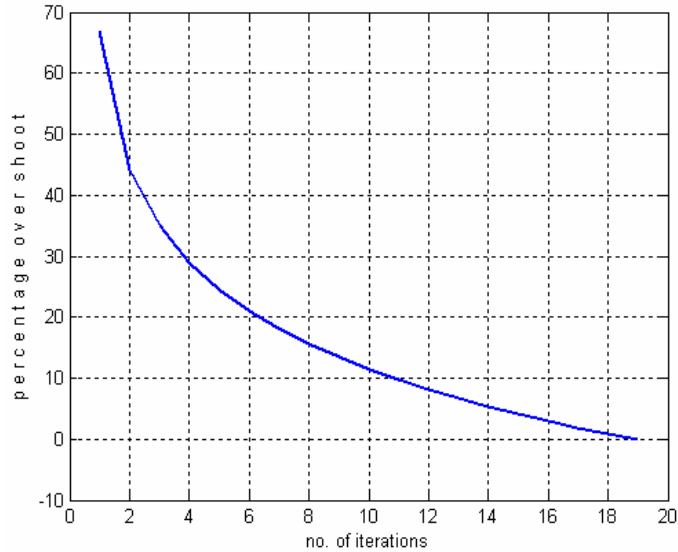


Figure 4.24: Percentage overshoot vs. number of iterations.

Using a value of  $\mu_{sse} = 0.01$  and  $\mu_{pos} = 0.01$  in equation (4.12) and (4.13), the value of  $a_j(5)$  and  $a_j(6)$  were learnt after 10 and 13 iterations. The plot of the history of  $a_j(5)$  and  $a_j(6)$  during this learning phase is given in figure 4.25.

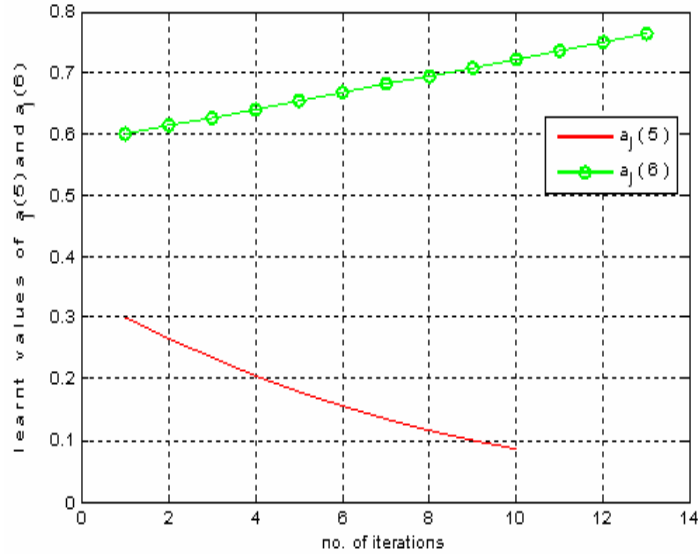


Figure 4.25: Learning values of  $a_j(5)$  and  $a_j(6)$  as iterations increase.

The output of the system, as steady state error was progressively reduced, is shown in figure 4.26.

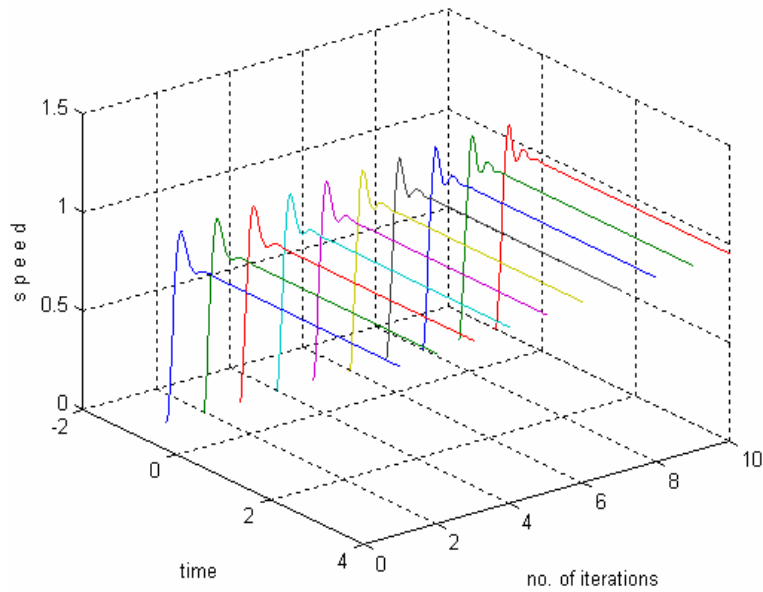


Figure 4.26: Plant output as iterations increase, while learning  $a_j(5)$ .

After the values of  $g_u$ ,  $a_j(5)$ ,  $a_j(6)$  are learnt, the control surface, input membership functions and the step response of the overall scheme are shown in figure 4.27, 4.28 and 4.29.

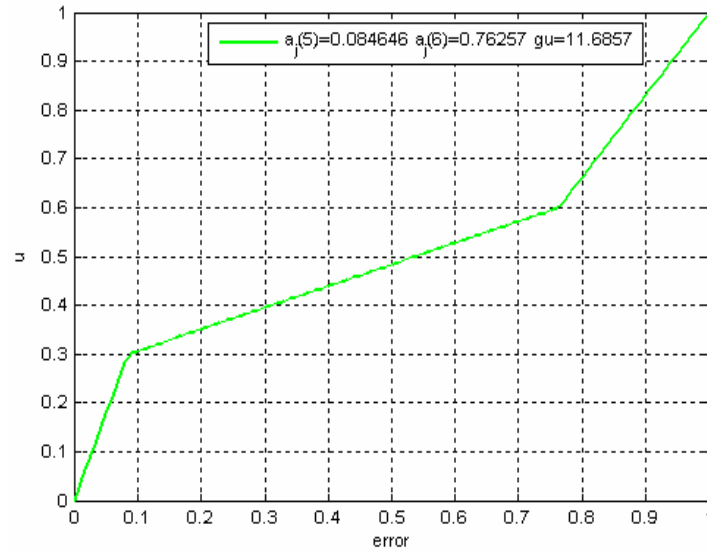


Figure 4.27: Control surface learnt by the system.

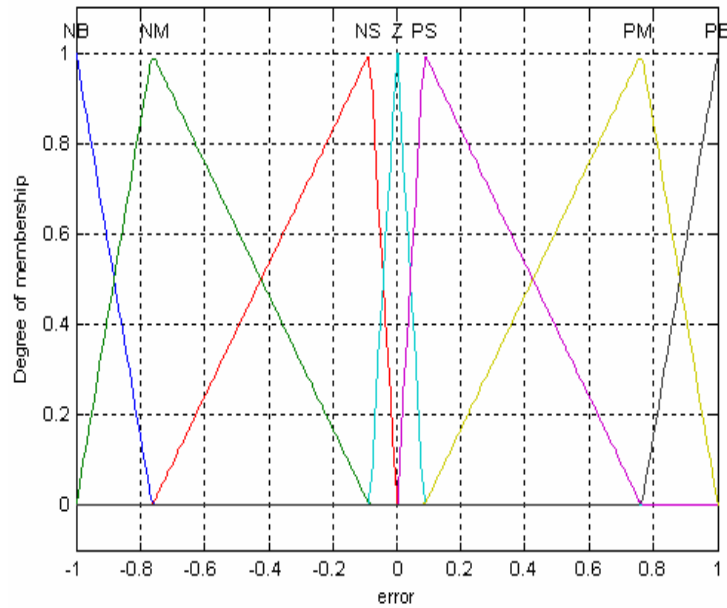


Figure 4.28: Learnt membership functions.

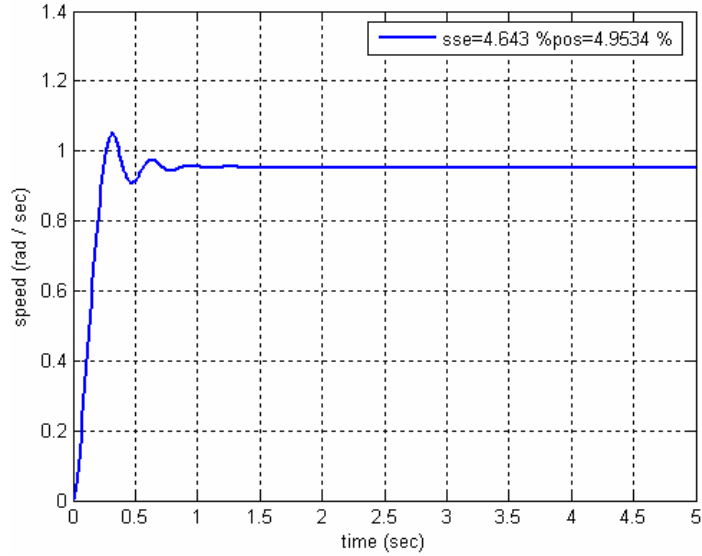


Figure 4.29: Step response after learning.

Figure 4.27 shows that a piece-wise-linear control surface was learnt by the ILFC scheme. Figure 4.28 shows the end points of the input membership functions. The final response in figure 4.29 shows that both the design requirements have been met without the introduction of integral or derivative of error. The behaviour of error is shown in figure 4.30.

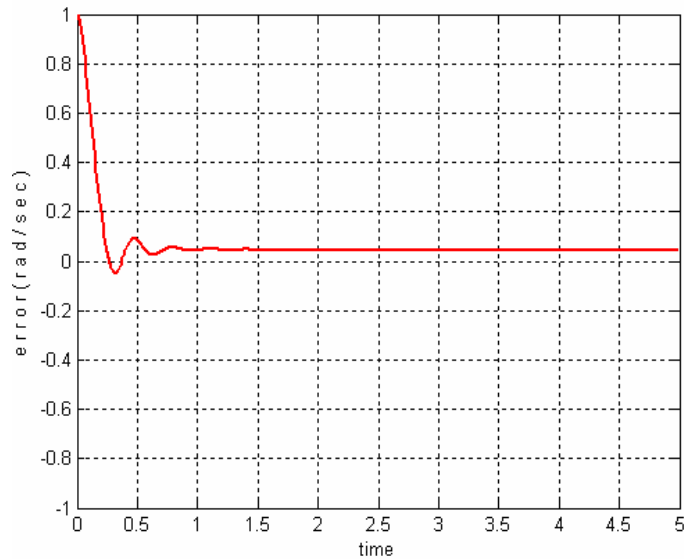


Figure 4.30: Behaviour of error.

The system was able to learn from an initial values of  $g_u = 1.0$ ,  $a_j(5) = 0.3$  and  $a_j(6) = 0.6$  to a final values of  $g_u = 11.68$ ,  $a_j(5) = 0.0846$  and  $a_j(6) = 0.7626$ . Looking at figure 4.27, we can conclude that there are 3 regions of interest for positive values of error. Similar 3 regions exist for negative values of error. The effective gain which is the slope of the lines in these regions is

$$K_p = \left\{ \begin{array}{ll} 41.4 & \text{if } 0 \leq e \leq a_j(5) \\ 4.67 & \text{if } a_j(5) \leq e \leq a_j(6) \\ 19.6 & \text{if } a_j(6) \leq e \leq 1 \end{array} \right\} \quad (4.24)$$

The combination of ILC and fuzzy has transformed the controller as if it were a combination of three controllers, for positive values of error. If we take both positive and negative values, it's a six controller in one. Each controller is effective within its defined range of error only. This gives the capability to increase or decrease the gain in small intervals to meet our design requirements.

#### 4.4.1.2.2 *A non-linear system*

Another set of simulations involved a second order coupled system used by [118]. The system equations are given as

$$x_1(k+1) = x_1(k) + 0.01x_2(k) + 0.01u(k) \quad (4.25)$$

$$x_2(k+1) = 0.1x_1(k) + 0.97x_2(k) \quad (4.26)$$

$$y(k+1) = x_1(k+1) \quad (4.27)$$

The plant was required to follow a step reference signal. The initial values of  $x_1$  and  $x_2$  were set as  $x_1(0) = 1$  and  $x_2(0) = 1$ .

Figure 4.31 shows the learning history of  $g_u$ , while figure 4.32 shows the learning history of  $a_j(5)$  and  $a_j(6)$ .

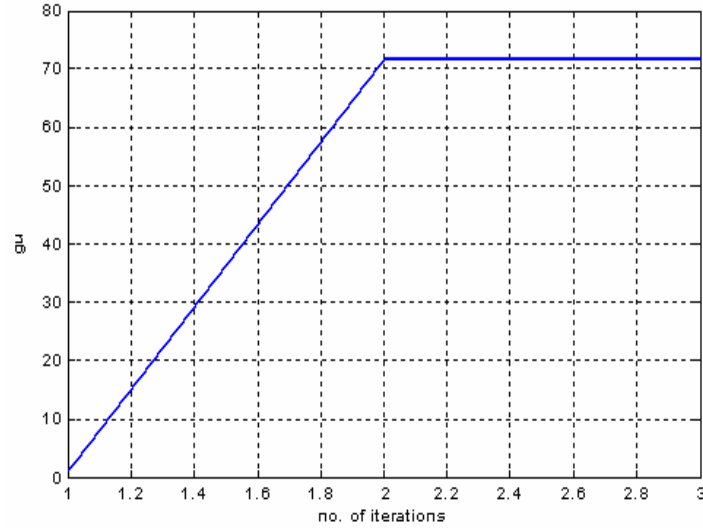


Figure 4.31: Learning values of  $g_u$  as iterations increase.

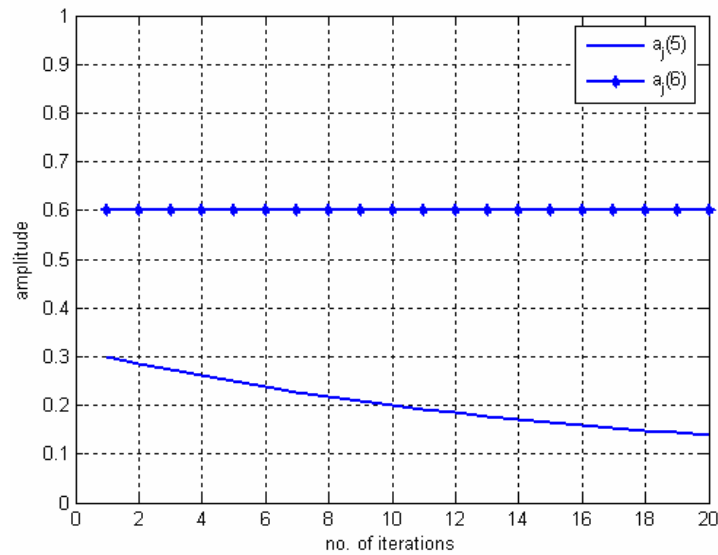


Figure 4.32: Learning values of  $a_j(5)$  and  $a_j(6)$  as iterations increase.

It took 2, 20 and 1 iterations to learn  $g_u$ ,  $a_j(5)$  and  $a_j(6)$  respectively.

After these parameters were learnt, the system produced the response as indicated in figure 4.33.

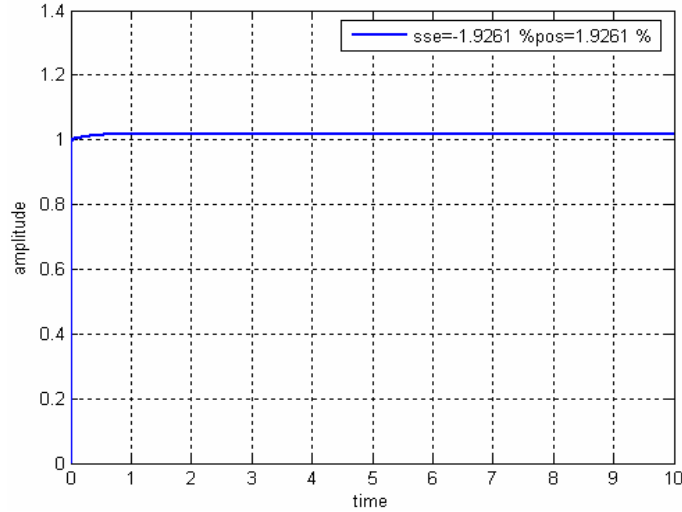


Figure 4.33: Step response after learning.

The response shows that both the design requirements are met. Furthermore, the uncertainty in membership function design is also catered for. The learnt control surface and the learnt membership functions are shown in figures 4.34 and 4.35.

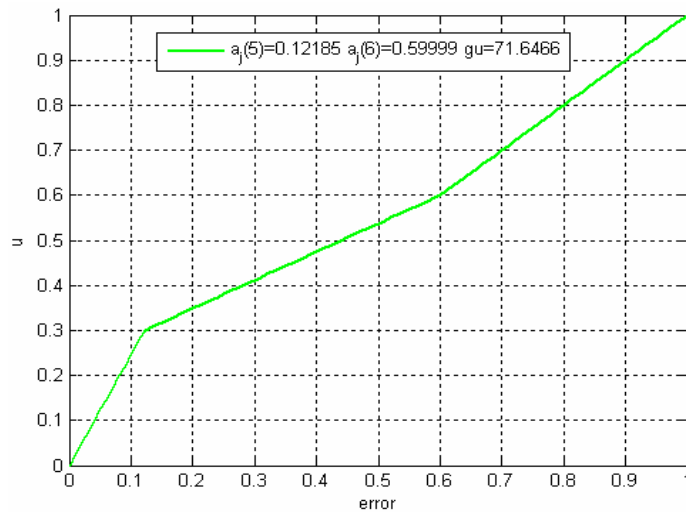


Figure 4.34: Control surface learnt by the system.

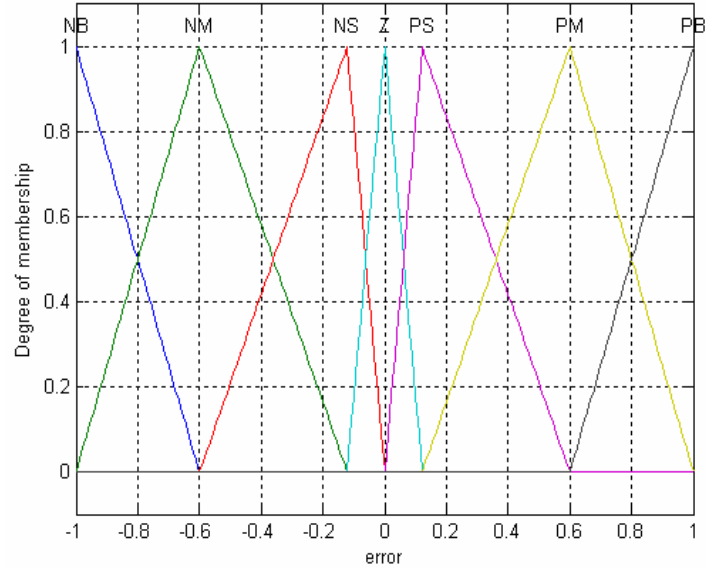


Figure 4.35: Learnt membership functions.

Again, the control surface is nonlinear and the membership functions learnt are squeezed together in the middle and expand away at the ends. The behaviour of error as shown in figure 4.36 below indicates a sharp decrease in error with time.

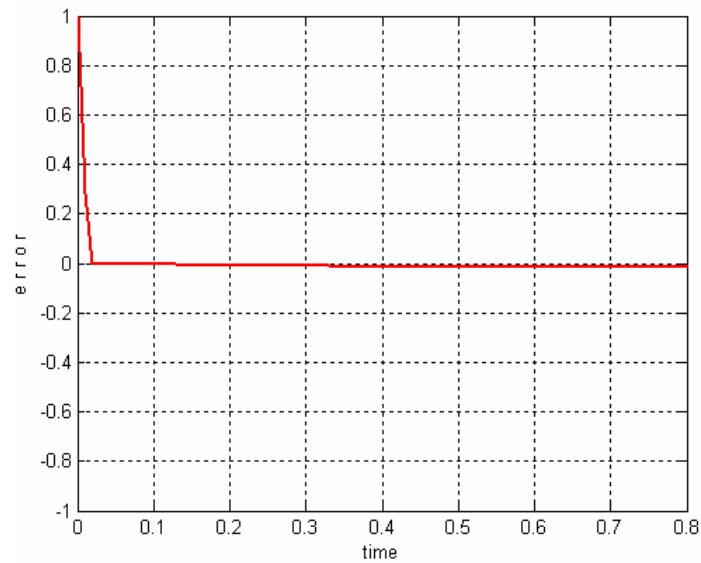


Figure 4.36: Trend of error vs. time.



Determining stability and convergence criteria for fuzzy based systems is a very challenging task. In the following section we present a novel approach to judge the stability of the Iterative Learning Fuzzy Tuner (ILFT).

#### 4.4.2 Stability and convergence

The proposed approach (ILFT), with 7 input and output membership functions, behaves as if there are 6 proportional controllers, which operate one at a time, for different ranges of errors. As three of the controllers, for negative values of errors, have same gains as those for positive error values, we can assume that basically there are three switch able proportional controllers. The ranges for positive errors are  $[0, a_j(5)]$ ,  $[a_j(5), a_j(6)]$  and  $[a_j(6), 1]$ . These ranges are named region1, region2 and region 3. A conceptual block diagram alternative for the approach in figure 4.13, with positive error values is described in figure 4.37.

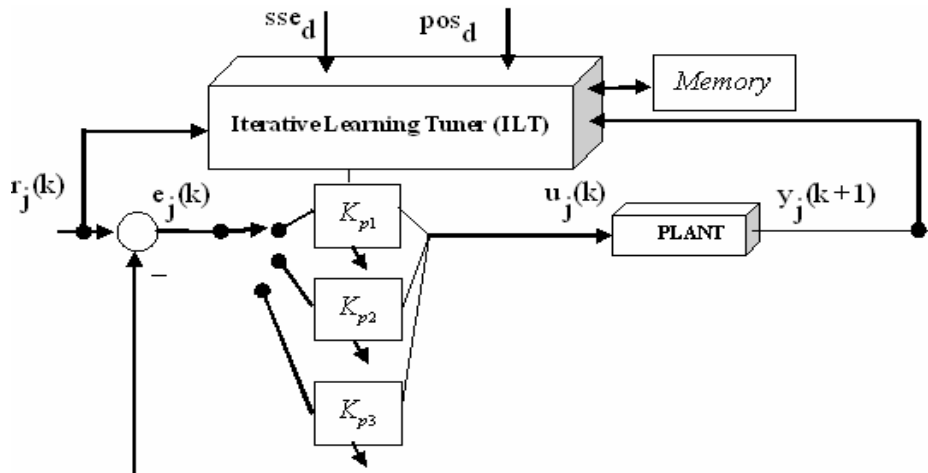


Figure 4.37: Conceptual view of the approach.

Here  $K_{p1}$  is the gain when error has values  $[0, a_j(5)]$ ,  $K_{p2}$  is the gain when error has values  $[a_j(5), a_j(6)]$  and  $K_{p3}$  is the gain when error has values  $[a_j(6), 1]$ .

Mathematically

$$K_p = \left\{ \begin{array}{ll} K_{p1} & \text{if } 0 \leq \text{abs}(e) \leq a_j(5) \\ K_{p2} & \text{if } a_j(5) \leq \text{abs}(e) \leq a_j(6) \\ K_{p3} & \text{if } a_j(6) \leq \text{abs}(e) \leq 1 \end{array} \right\} \quad (4.28)$$

For the motor speed control system, as seen from equation (4.24), the gains learnt are 41.4 in  $[0, a_j(5)]$  range, 4.67 in  $[a_j(5), a_j(6)]$  range and 19.6 in  $[a_j(6), 1]$  range. The gain is relatively high for very large error, to quickly reduce the error. As the error is reduced, the gain is lowered to avoid overshoot. Once the error is low enough, high gain is applied to reduce the steady state error. All this is automatically calculated by the ILT.

This conceptual view gives us the ease to use the well known methods like Root locus, Nyquist stability criteria, Routh's stability criteria, Phase Plane method etc. to determine the stability. Discrete Root locus of the DC motor model of equation (4.14) is described in figure 4.38.

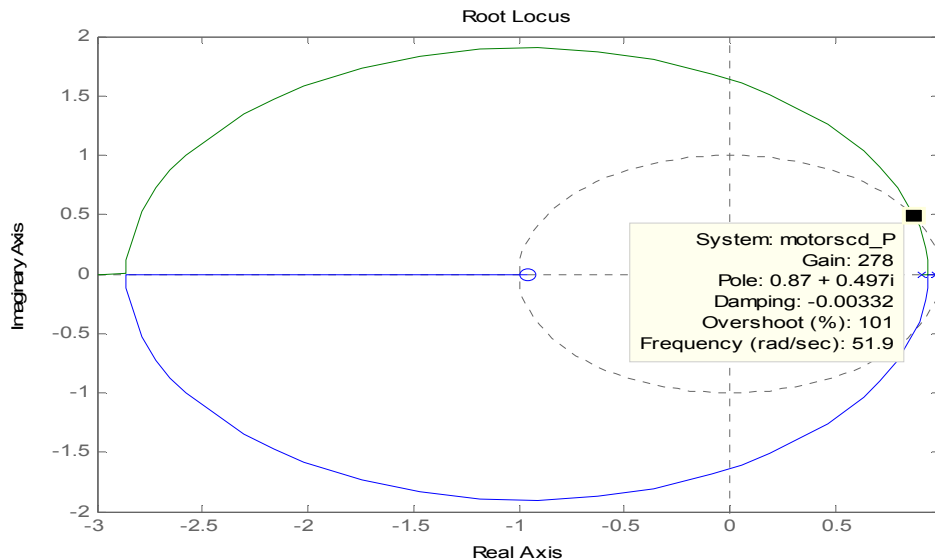


Figure 4.38: Root locus of the system.

The plot shows that there is no stability problem with  $K_p = [-278, 278]$  range.

A universal reliable method to determine the stability of a fuzzy control system has not yet been developed though some methods have been proposed based on Liapunov second method. We now use the following assumptions:-

- (a) The reference signal  $r_j(k)$  is repeatable over a finite interval  $[0, T]$ , where  $T$  is a finite positive constant.
- (b) The system is stable under fuzzy control before the iterative learning algorithm is introduced.
- (c) The value of  $a_j(5)$  is bounded between  $[0, a_j(6)]$  and the value of  $a_j(6)$  is bounded between  $[a_j(5), 1]$ .

Using the above assumptions, if the conventional controller of figure 4.21 is stable for  $K_p = K_{p1}$ ,  $K_p = K_{p2}$  and  $K_p = K_{p3}$ , our proposed approach will also be stable.

It was noted that lower values of error had more impact on steady state error while higher values of error had more impact on percentage overshoot. The behaviour is shown in figure 4.39.

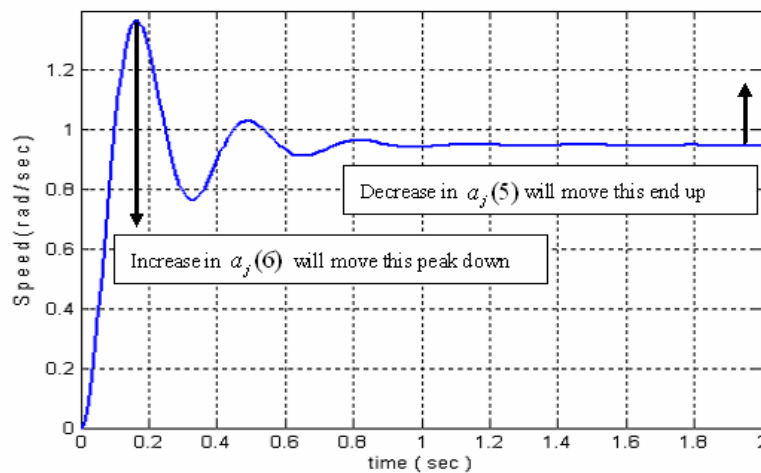


Figure 4.39: Effect of moving end points  $a_j(5)$  and  $a_j(6)$ .

This effect is discussed w.r.t. regions below:-

**Region 1:** From 0 to  $a_j(5)$

If the value of  $a_j(5)$  is moved towards zero the slope of the control surface in this region increases. In this region we try to reduce steady state error. The end point  $a_j(5)$  is

selected to reduce this error. A decrease in  $a_j(5)$  moves the system response in the steady state region upwards, while an increase in  $a_j(5)$  moves the response downwards (figure 4.39). Hence the ILT is always selecting  $a_j(5)$  to reduce steady state error.

**Region 2:** From  $a_j(5)$  to  $a_j(6)$

Reducing  $a_j(5)$  reduces the slope of the control surface while reducing  $a_j(6)$  increases the slope of the control surface in this region. This region has no or very negligible impact on steady state error. The gain should be low enough to avoid overshoot.

**Region 3:** From  $a_j(6)$  to 1

Increasing  $a_j(6)$  increases the slope of the control surface in this region. When this region is effective the error is large. The end point  $a_j(6)$  is selected according to equation (4.13) so as to give a speedy initial response.

Similar arguments can be made for negative values of error. Hence, if the assumptions are fulfilled, the system will not be unstable because of ILT and will try to converge according to conventional control theory [135].

The effectiveness of ILFT in tracking different trajectories, in real time, is discussed in the next section.

### 4.4.3 Tracking a desired trajectory in real time

Tracking is an important practical problem. It becomes a difficult problem, especially when the target changes its position continuously. To demonstrate the effectiveness of this approach, different trajectories were tracked in real time; one of them was a sinusoidal signal. This tracking was done after the learning of the parameter had been achieved as discussed before. For this demonstration, system of equation (4.14) was used and was given a task to follow sinusoidal speed requirements. These speed

requirements meant that the motor had to move in one direction with a maximum of 1 rad/sec speed and then in the opposite direction, to achieve the same speed. The plot in figure 4.40 below shows the results.

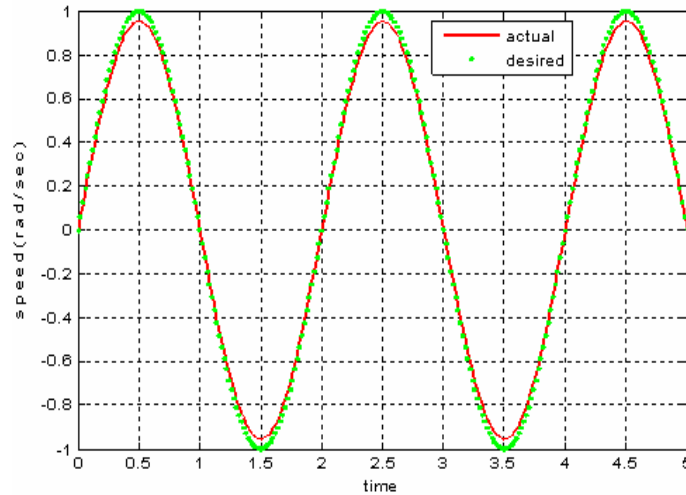


Figure 4.40: Tracking a sinusoidal reference trajectory.

Figure 4.40 shows that the system was quite effectively able to track the desired response. This should be noted that the scheme learnt the parameters for  $< 5\%$  steady state error. For a system with lesser steady state error requirements, even better performance was achieved. A plot of error vs. time is shown in figure 4.41.

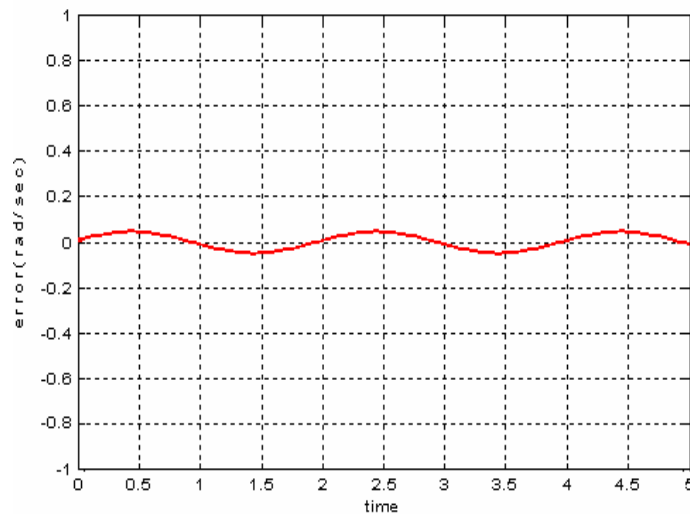


Figure 4.41: Plot of error with time.

The same system with the same learnt parameters was made to track other arbitrary trajectories also. One of the results from such tracking is shown in figure 4.42 below.

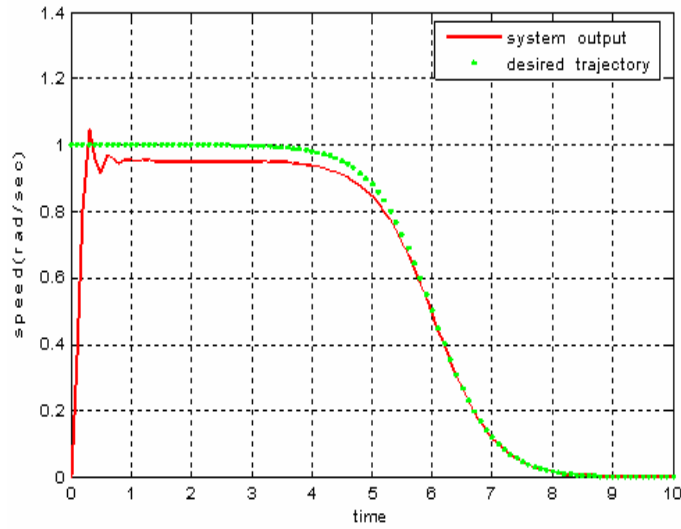


Figure 4.42: The output of the system and the reference trajectory.

The behaviour of error with respect to time is plotted in figure 4.43. The plot shows that the system was able to track the reference trajectory successfully.

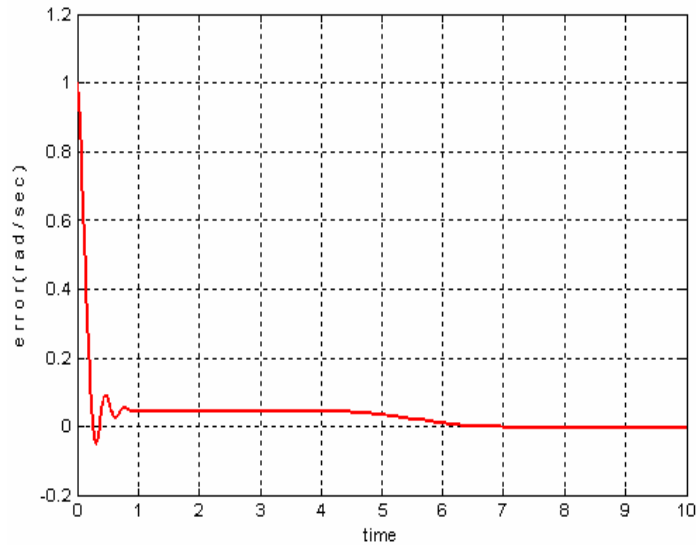


Figure 4.43: Behaviour of error with time.

After an initial error the system was able to track the trajectory perfectly and the initial overshoot is also within our desired limits.

There can be situations where the design requirements can not be fully met by considering the error alone. The best solution in this case is to introduce more inputs, like derivative of error or integral of error. This scenario is discussed now.

#### 4.4.4 Effect of considering derivative of error

We know from conventional PID control design that the derivative part can reduce percentage overshoot and the integral part can reduce steady state error. Similar behaviour was observed for this scheme. For systems whose performance specifications can not be met with only error as input membership function, it is proposed to introduce derivative of error, integral or sum of error also, as per requirement. Derivative of error should be considered if overshoot requirements are not met and integral of error should be considered if steady state error requirements are not met. Figure 4.44 below shows the block diagram of the approach, when rate of change of error is considered.

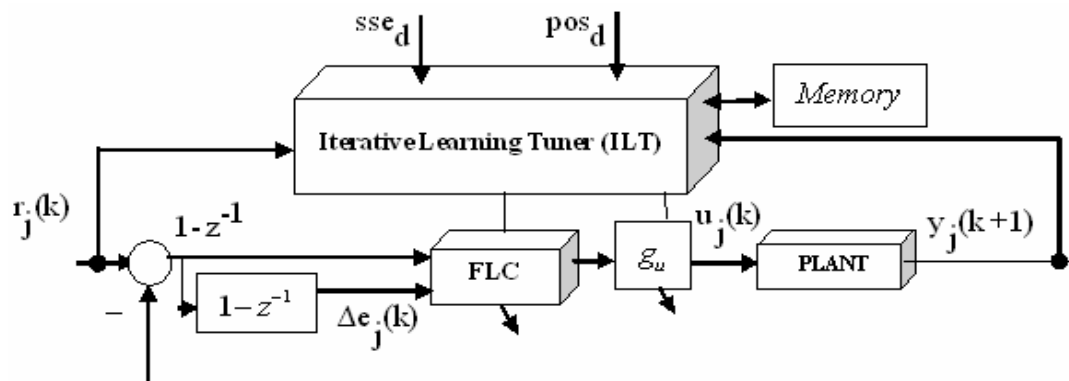


Figure 4.44: Block diagram of the proposed controller considering rate of change of error also.

The only change as compared to figure 4.13 is the introduction of another input i.e. change of error. This increases the type of the fuzzy controller.

There can be five different combinations of  $e$  and  $\Delta e$ .

**Case 1:**  $e$  is +ve and  $\Delta e$  is +ve

This implies that the present error is greater than old error i.e.  $e_j(k) > e_j(k-1)$ .

The output is going in the wrong direction. The output is lower than the reference signal i.e.  $y_j(k) < r_j(k)$  and moving away from it. Therefore, the output should be moved up in the positive direction hence PB,PM,PS should be dominant.

**Case 2:**  $e$  is +ve and  $\Delta e$  is -ve

This implies that the present error is less than the old error i.e.  $e_j(k) < e_j(k-1)$ .

The output is going in the right direction. The output is lower than the reference signal i.e.  $y_j(k) < r_j(k)$  and moving towards it.

**Case 3:**  $e$  is -ve and  $\Delta e$  is +ve

This implies that the present error is less than the old error i.e.  $e_j(k) < e_j(k-1)$ .

The output is going in the right direction. The output is greater than the reference signal i.e.  $y_j(k) > r_j(k)$  and moving towards it. Therefore if negative input membership function is dominant output should be negative and if positive input membership function is dominant output should be positive.

**Case 4:**  $e$  is -ve and  $\Delta e$  is -ve

This implies that the present error is greater than old error i.e.  $e_j(k) > e_j(k-1)$ .

The output is going in the wrong direction. The output is greater than the reference signal i.e.  $y_j(k) > r_j(k)$  and moving away from it. Therefore the output should be moved in the negative direction i.e. NB,NM,MS.

**Case 5:**  $e$  is 0 and  $\Delta e$  is 0

This implies that the present error is equal to old error i.e.  $e_j(k) = e_j(k-1)$ . The output is the desired output. No change in output is desired. Z membership function should be dominant.

Using the discussion above the following rule base is proposed.



$\Delta e \backslash e$	<b>NB</b>	<b>NM</b>	<b>NS</b>	<b>Z</b>	<b>PS</b>	<b>PM</b>	<b>PB</b>
<b>NB</b>	NB	NB	NB	NB	NM	NS	Z
<b>NM</b>	NB	NB	NB	NM	NS	Z	PS
<b>NS</b>	NB	NB	NM	NS	Z	PS	PM
<b>Z</b>	NB	NM	NS	Z	PS	PM	PB
<b>PS</b>	NM	NS	Z	PS	PM	PB	PB
<b>PM</b>	NS	Z	PS	PM	PB	PB	PB
<b>PB</b>	Z	PS	PM	PB	PB	PB	PB

Table 4.8: Proposed rule Base for the controller.

The introduction of rate of change as the second input, suppresses the overshoot. This is obvious in the response of the system as  $a_j(5)$  is learnt. This response as  $a_j(5)$  was learnt against different number of iterations is shown in figure 4.45.

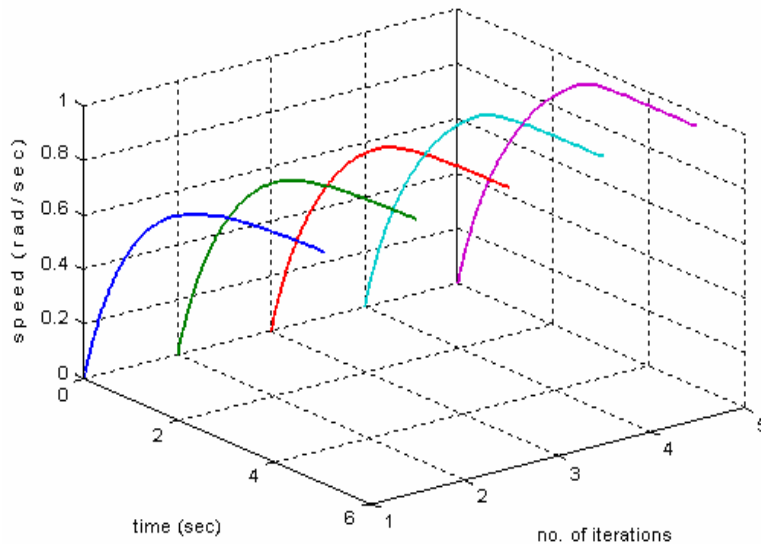


Figure 4.45: Plant output as iterations increase, while learning  $a_j(5)$ .

It took just 5 iterations to learn  $a_j(5)$  and no learning was required for  $a_j(6)$  as introduction of the derivative term reduced the overshoot within the required limit. After the learning process was finished the control surface learnt is shown in figure 4.46 below.

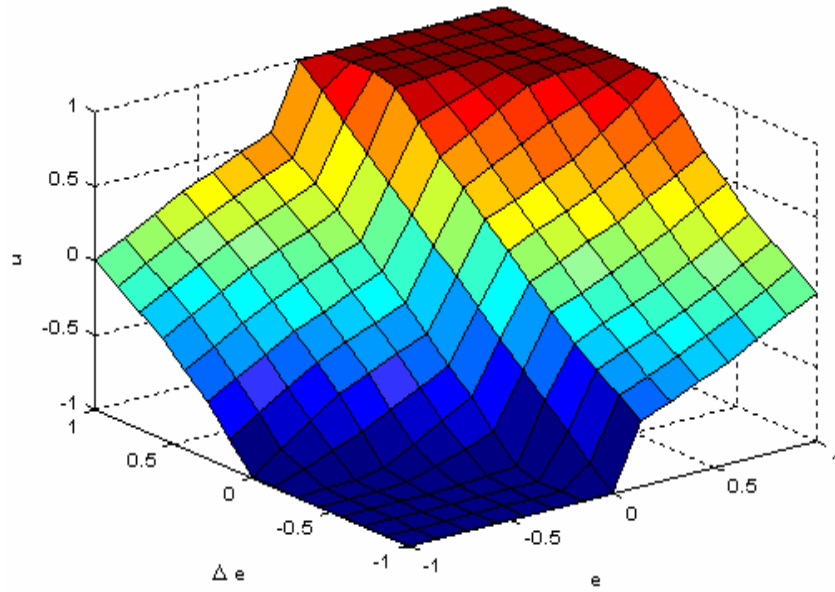


Figure 4.46: Control surface of the controller.

A highly nonlinear control surface is learnt automatically by the proposed scheme. With this controller the behaviour of error for a desired response of 1 rad/sec is plotted in figure 4.47.

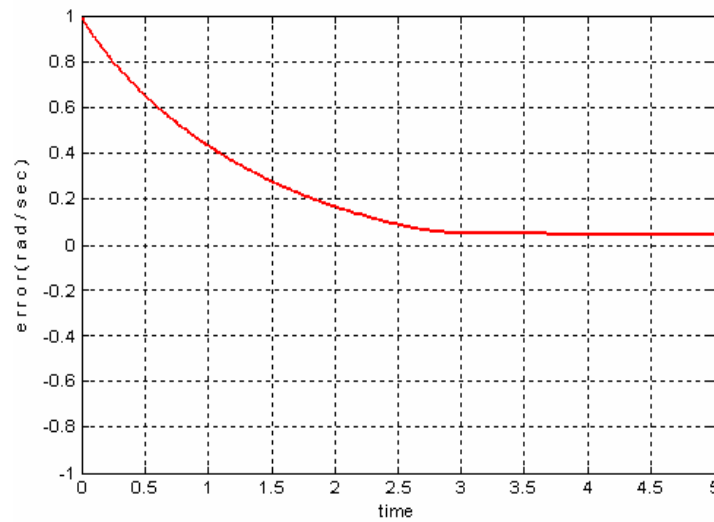


Figure 4.47: Behaviour of error.

A comparison of this plot with that in figure 4.30 shows that even the behaviour of error is non oscillatory due to introduction of rate of change of error. The parameters learnt

during the learning phase were  $g_u = 11.68$ ,  $a_j(5) = 0.0838$  and  $a_j(6) = 0.6$ . No learning was required for parameter  $a_j(6)$ .

Number of simulations, also considering rate of change of error and integral of error on different systems led to the formation of a procedure for the design of ILFT. The procedure is described in figure 4.48.

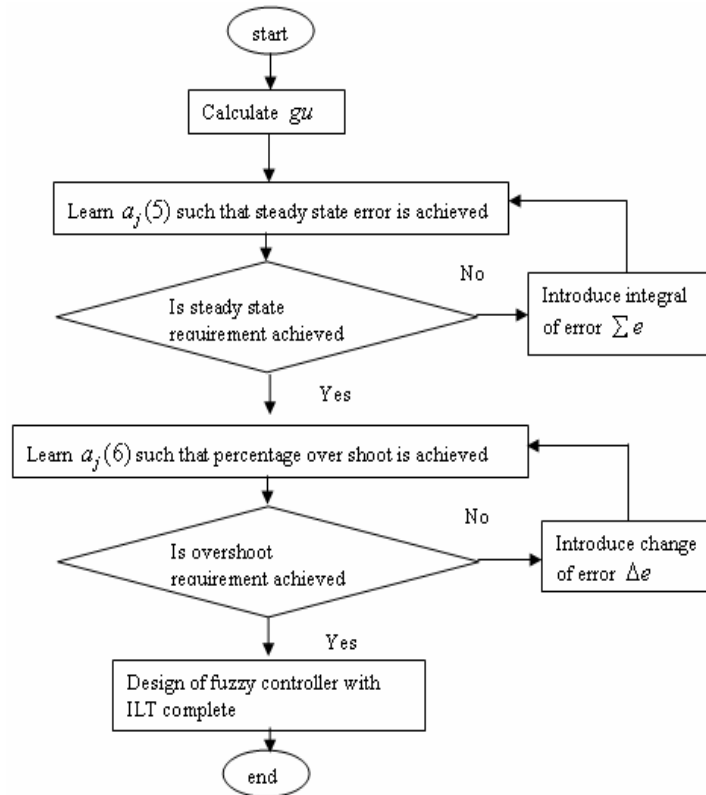


Figure 4.48: Flow chart showing the procedure for selecting rate and/or integral of error.

If rise time is also an issue care must be taken in introducing  $\Delta e$  as it has an impact on rise time of the system also.

#### 4.4.5 Stability using linguistic trajectory

Fuzzy systems are non-linear systems. From conventional non-linear control theory we know that the stability of a non-linear system not only depend on shape and

amplitude of input signal, but also on location of initial conditions. One method used for stability analysis of non-linear systems is the phase plane method. Control systems designers have been using phase plane analysis to generate motion trajectories of the system in the state space domain. The qualitative analysis of the trajectories gives information concerning stability, robustness and convergence. The main advantage of the phase plane analysis is its graphical nature which allows us to visualize what goes on in a non-linear system. These graphs are plotted for different initial conditions without having to solve any equation analytically.

This method can also be used to study the stability of fuzzy based systems by observing the sequence in which the rules are fired. If we draw this as a trajectory on a rule table, the error and change of error should be decreasing during the rule firing process [101, 103]. This means that we have to move along this trajectory from the edges of the table to its centre. For an unstable system there is an inverse trajectory or at least this trend is not present.

Let  $x = (x_1, x_2)^T$  be the state. We consider the following fuzzy control system

$$\dot{x} = f(x) + bu \quad (4.29)$$

$$u = \varphi(x) \quad (4.30)$$

Here  $f(x)$  represents plants dynamics and is a nonlinear vector function,  $b$  is a two-dimensional vector,  $u$  is a scalar control variable and  $\varphi(x)$  is a two input one output fuzzy system. We define  $N_1$  and  $N_2$  fuzzy sets to cover the domains of  $x_1$  and  $x_2$ . We also suppose that the fuzzy rule base of  $\varphi(x)$  consists of  $N_1 \times N_2$  rules. Let the rule be of the form in equation (4.2). We say that the point  $(x_1, x_2)$  in the phase belongs to rule  $l^*$  if it holds that

$$\mu_{A_1}^{l^*}(x_1) * \mu_{A_2}^{l^*}(x_2) \geq \mu_{A_1}^l(x_1) * \mu_{A_2}^l(x_2) \quad (4.31)$$

For all  $l \neq l^*$ , where  $*$  represents t-norm.

The tangent vector of the state trajectory equals the summation of vector field  $f(x)$  and vector field  $b\varphi(x)$  as shown in figure 4.49.

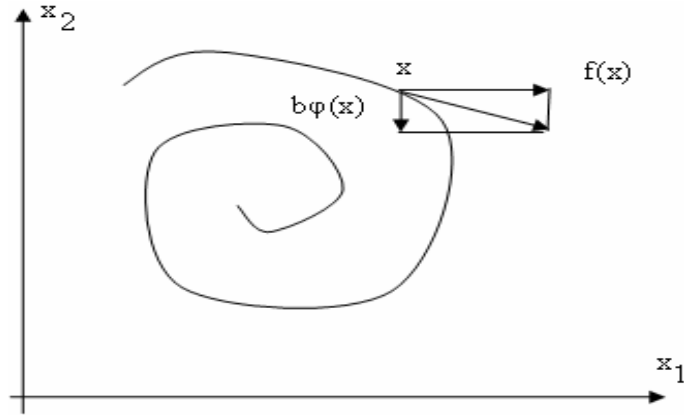


Figure 4.49: State trajectory moves along the direction of  $f(x) + b\varphi(x)$

The subspace  $\varphi(x) = 0$  is a line called the switching line. When the open loop system  $\dot{x} = f(x)$  is stable and the control  $u = \varphi(x)$  tries to lead the system trajectory towards the switching line, the plant component  $f(x)$  has a greater influence which makes the trajectory converge to the equilibrium point. On the other hand if the open loop system is unstable and the control tries to stabilize the system. At this moment if the state trajectory moves near the switching line the unstable plant component  $f(x)$  has a greater influence which makes the state trajectory diverge away from the equilibrium point. This interaction between the control and the plant component makes the state oscillate around the equilibrium point and a limit cycle is formed.

Considering  $b = 1$  the equilibrium point of the fuzzy system described by equations (4.29) and (4.30) is determined by

$$\dot{x} = f(x) + \varphi(x) = 0 \quad (4.32)$$

Since  $f(0) = \varphi(0) = 0$ , the origin is an equilibrium point. For this condition to be stable a sufficient condition is [103].

$$\frac{d}{dx}[f(x) + \varphi(x)]|_{x=0} = f'(0) + \varphi'(0) < 0 \quad (4.33)$$

Consequently the closed loop system in equation (4.29) and (4.30) is globally stable if the following two conditions are satisfied [103].

$$f'(0) + \phi'(0) < 0 \tag{4.34}$$

$$|\phi(x)| < |f(x)|, \forall x \neq 0 \tag{4.35}$$

The linguistic trajectory for the rule base in table 4.8 and a desired speed of 1 rad/sec is shown in table 4.9 below.

$e$ \ $\Delta e$	<b>NB</b>	<b>NM</b>	<b>NS</b>	<b>Z</b>	<b>PS</b>	<b>PM</b>	<b>PB</b>
<b>NB</b>	NB	NB	NB	NB	NM	NS	Z
<b>NM</b>	NB	NB	NB	NM	NS	Z	PS
<b>NS</b>	NB	NB	NM	NS	Z	PS	PM
<b>Z</b>	NB	NM	NS	<b>Z</b> ▼	PS	PM	PB
<b>PS</b>	NM	NS	Z	PS	PM	PB	PB
<b>PM</b>	NS	Z	PS	PM	PB	PB	PB
<b>PB</b>	Z	PS	PM	PB	PB	PB	PB

Table 4.9: Linguistic trajectory for simulation results in section 4.4.1.6.

The trajectory shows smooth reduction of error. Also there are no stability issues as the trajectory is always moving toward the centre Z membership function. The rule base in table 4.8 can be modified if a different trajectory needs to be followed. The linguistic trajectory plot also makes this modification easier.

## 4.5 A Real Time Tracker Using ILT

A classical controller can become highly complex when a system is non linear or has complex dynamics. On the other hand, humans tend to take care of such complex problems like tracking, fairly easily. Moreover, humans are unaware of the mathematical model of the system. Even while driving a car, humans do not have a model of the car, driver or distance in mind, but just an idea of the error or its range. This proposed ILT

based controller discussed earlier can be easily applied to systems where the mathematical model is unavailable.

This research presents a real time tracker based on ILFT scheme discussed earlier and shows its effectiveness using simulations as well as practical demonstrations.

The block diagram of the real time tracker is presented in figure 4.50.

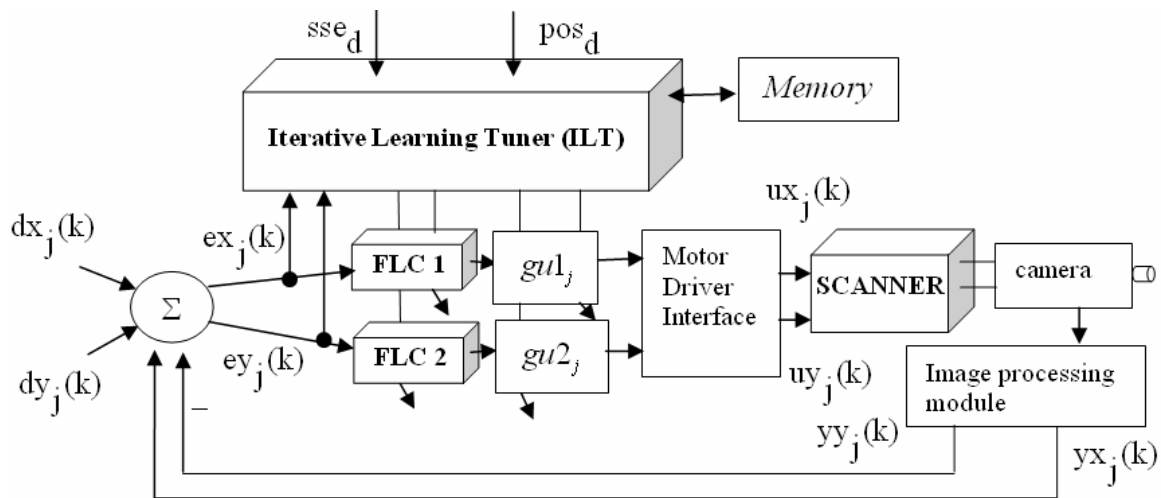


Figure 4.50: Block diagram of the real time tracker.

Here  $dx_j(k)$  and  $dy_j(k)$  are the desired x and y positions of the Scanner for  $k = 1 \dots N$  and  $j = 1 \dots \infty$ . Variable  $j$  represents the iteration number and  $k$  represents the samples. During the tracking mode, the value of  $k$  is 1 as each sample is taken as new iteration with its initial state as the final state of the previous iteration. The errors in the x and y plane are represented by  $ex_j(k)$  and  $ey_j(k)$ . These error values are fed to two FLC blocks. FLC1 controls the movements of Scanner for x axis and FLC2 controls the movements for y axis. The output of these blocks is converted into appropriate signals for the Scanner, using the output scaling factors  $gu1_j$ ,  $gu2_j$  and the Motor Driver Interface card. The inputs to the Scanner are  $ux_j(k)$  and  $uy_j(k)$ . Each input is used to control one axis of the XY plane. The Scanner has a camera mounted on it which takes pictures of the scene regularly. The Image processing module detects the target and locates its current x

and y positions marked  $yx_j(k)$  and  $yy_j(k)$ . Using this target position the error signal is generated.

The acceptable or desired steady state error ( $sse_d$ ) and the acceptable or desired percentage over shoot ( $pos_d$ ) is fed to the ILT block. The ILT adjusts  $gu1_j$ ,  $gu2_j$  and tunes the fuzzy logic controllers (FLCs) by adaptively adjusting the membership function end points. The aim is to converge with respect to given desired steady state error and percentage overshoot. The learnt values of  $gu1_j$ ,  $gu2_j$  and membership function end point values are stored in memory to be used in future iterations. The ILT block uses the error signals and learning laws to adjust these parameters.

The Scanner for simulation purposes is a 2 Degree of freedom tracking platform (2DOFTP), the details of which are given in section 2.9. The camera used had selectable pixel resolution of 640x480 or 320x240. The pixel coordinates for a 320x240 camera resolution are shown in figure 4.51 below

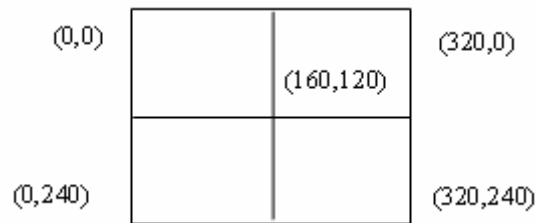


Figure 4.51: Camera pixel coordinates.

These camera pixel coordinates were converted to a virtual coordinate for mathematical convenience as in figure 4.52.

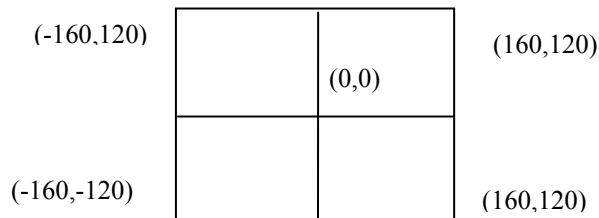


Figure 4.52: Virtual coordinates and Centre of Camera (COC) i.e. point (0,0).

The aim is to have the target at (0,0) i.e.



$$dx_j(k) = 0 \text{ and } dy_j(k) = 0$$

The errors  $ex_j(k)$  and  $ey_j(k)$  are normalized before being supplied to the FLC blocks using the equations

$$ex_j(k) = \frac{dx_j(k) - yx_j(k)}{160} \quad (4.36)$$

$$ey_j(k) = \frac{dy_j(k) - yy_j(k)}{120} \quad (4.37)$$

The desired x, y positions  $y_j dx(k)$ ,  $y_j dy(k)$  the current x,y positions  $yx_j(k)$ ,  $yy_j(k)$  and the errors in x,y directions  $ex_j(k)$ ,  $ey_j(k)$  all are in pixels. The 7 input MFs for error and 7 output membership functions for voltage were of the form in figure 4.14 for both FLC blocks. The membership function end points were renamed as

$$a_j(1) = ax_j(1), \quad a_j(2) = ax_j(2), \quad a_j(3) = ax_j(3), \quad a_j(4) = ax_j(4), \quad a_j(5) = ax_j(5), \\ a_j(6) = ax_j(6) \text{ and } a_j(7) = ax_j(7) \text{ for FLC1 block and}$$

$$a_j(1) = ay_j(1), \quad a_j(2) = ay_j(2), \quad a_j(3) = ay_j(3), \quad a_j(4) = ay_j(4), \quad a_j(5) = ay_j(5), \\ a_j(6) = ay_j(6) \text{ and } a_j(7) = ay_j(7) \text{ for FLC2 block.}$$

The ILT learning procedure, described in equations (4.11), (4.12) and (4.13) is modified for FLC1 block as:-

if percentage over shoot < 0.0

$$gu1_{j+1} = gu1_j + \mu_{gu1}(r_j(k) - y_j(k)) \quad (4.38)$$

else

$$gu1_{j+1} = gu1_j - \mu_{gu1}(r_j(k) - y_j(k)) \quad (4.39)$$

if  $ydx_j(k) - yx_j(k) > sse_d$

$$ax_{j+1}(5) = ax_j(5) - \mu_{sse}(r_j(k) - y_j(k)) \quad (4.40)$$

else

$$ax_{j+1}(5) = ax_j(5) + \mu_{sse}(r_j(k) - y_j(k)) \quad (4.41)$$

ILT learning equation for percentage overshoot is modified as :-

if  $pos_j > pos_d$

$$ax_{j+1}(6) = ax_j(6) + \mu_{pos}(r_j(k) - y_j(k)) \quad (4.42)$$

else

$$ax_{j+1}(6) = ax_j(6) - \mu_{pos}(r_j(k) - y_j(k)) \quad (4.43)$$

Here  $\mu_{pos}$  is the step size parameter for correcting percentage over shoot and  $pos_j$  is the percentage overshoot value for the current iteration. Similar equations can be derived for FLC2 block tuning.

#### 4.5.1 Simulation results

Using the motor of equation (4.14), simulations were run to test the real time tracker. The aim is to make the ILT based controller tune for less than 2% steady state error and less than 5% percentage over shoot. This amounts to a steady state error of  $\pm 3.2$  pixels in x direction and  $\pm 2.4$  pixels in y direction. The permissible overshoot in x direction is  $\pm 8$  pixels and  $\pm 6$  pixels in y direction. The learnt values of the two output scaling factors and the membership function endpoints were  $gu1_j=11.68$ ,  $ax_j(5)=0.0846$ ,  $ax_j(6)=0.7625$ ,  $gu2_j=11.68$ ,  $ay_j(5)=0.0846$  and  $ay_j(6)=0.7625$ .

Figures 4.52 shows the Scanner tracking a static target at  $x=200$  and  $y = 100$ . The target is tracked with in one second of operation.

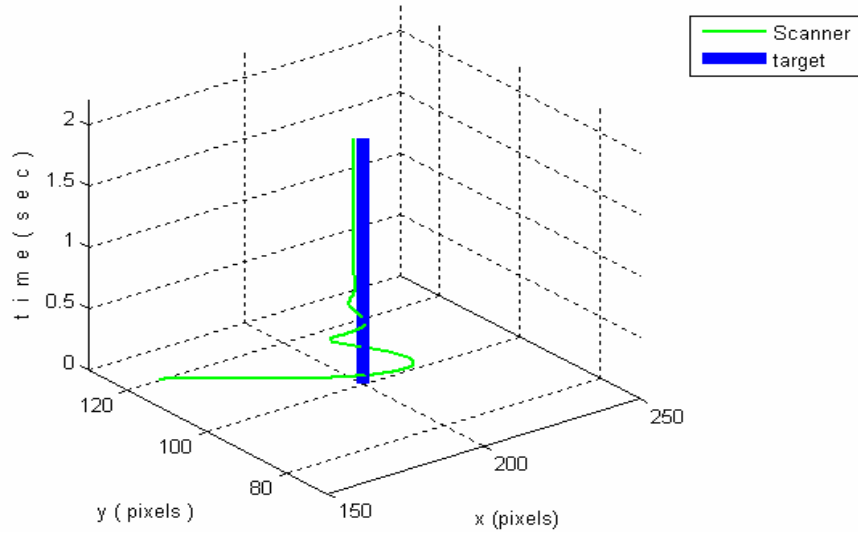


Figure 4.52: Response of the Scanner against a static target.

The behaviour of errors in the x and y plane are shown in figure 4.53 and 4.54.

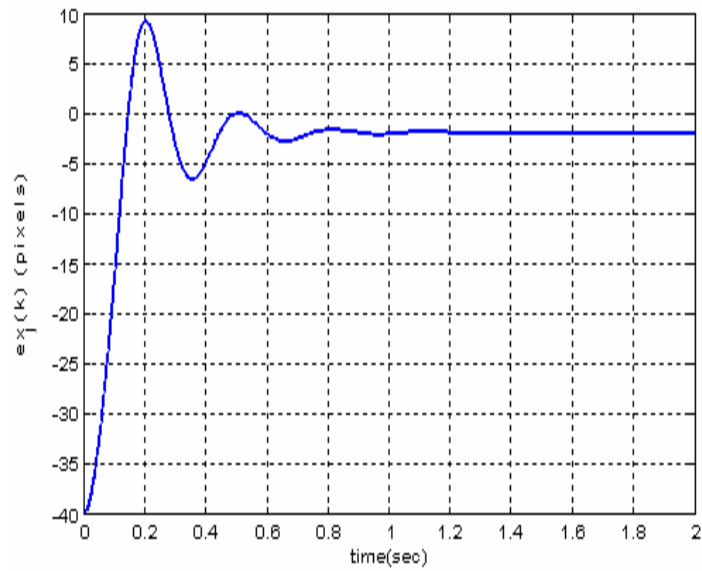


Figure 4.53: Behaviour of error in the x axis as the object is tracked.

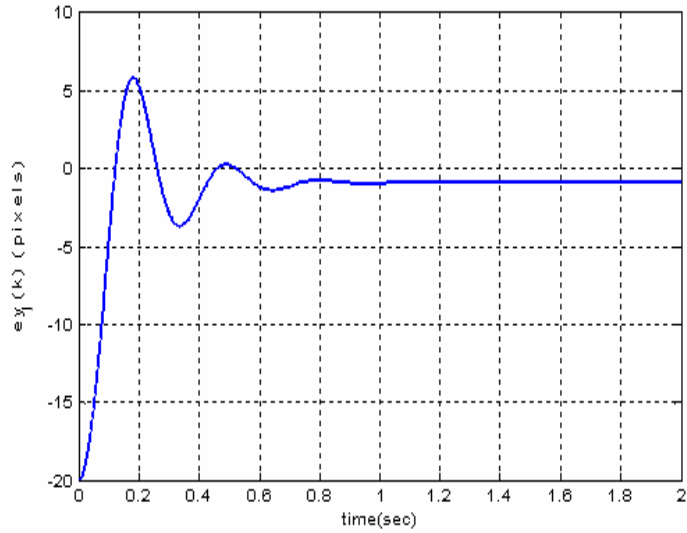


Figure 4.54: Behaviour of error in the y axis as the object is tracked.

Both figures show quick decrease in error. The Scanner output w.r.t. the desired target positions are shown in figure 4.55 and 4.56.

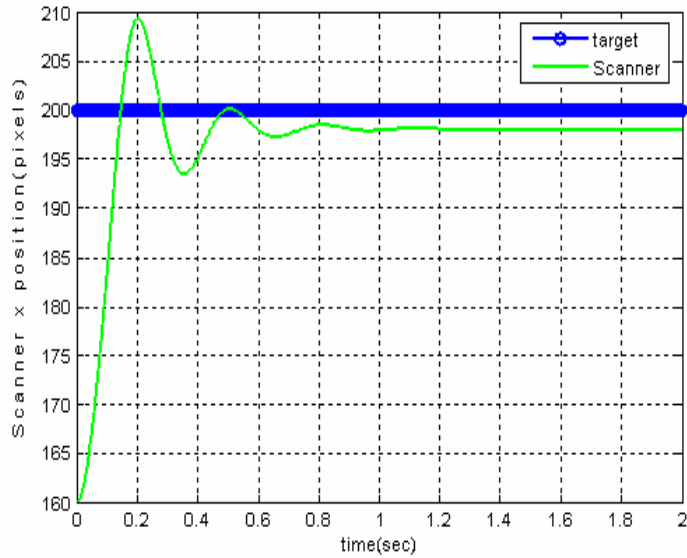


Figure 4.55: Position of Scanner in x axis as the object is tracked.

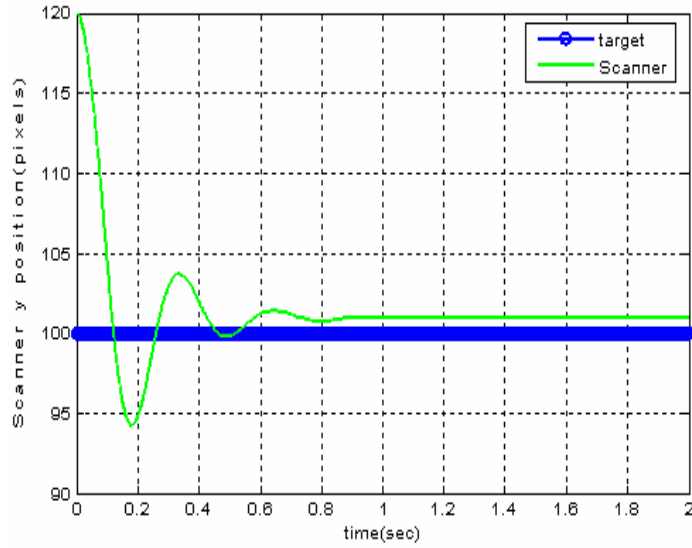


Figure 4.56: Position of Scanner in y axis as the object is tracked.

The trajectory followed by the Scanner as it tries to track the target is shown in figure 4.57. The trajectory has a converging shape.

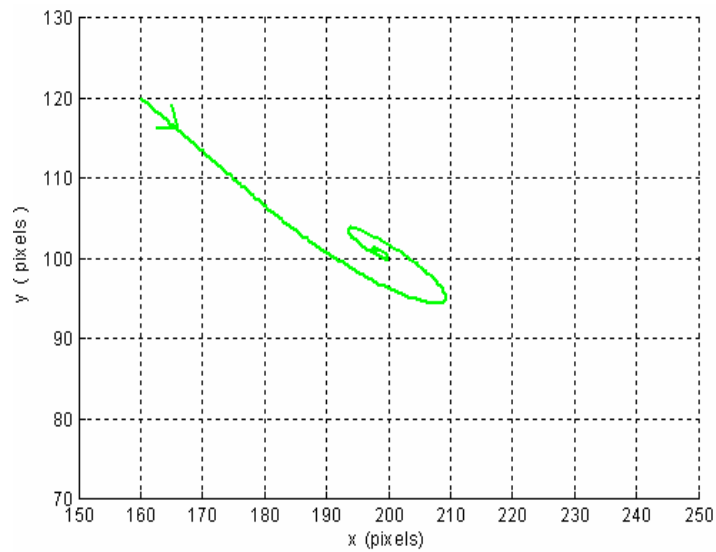


Figure 4.57: Trajectory of the Scanner in the XY plane.

Moving targets were also successfully tracked using this setup. One such tracking performance is exhibited in figure 4.58.

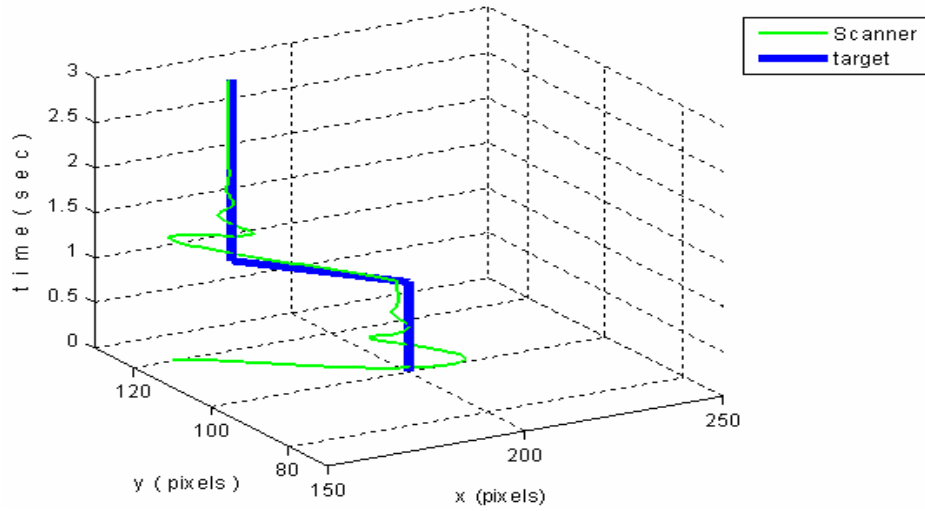


Figure 4.58: Response of the Scanner against a moving target.

For this moving target the behaviour of errors in the x and y plane are shown in figure 4.59 and 4.60.

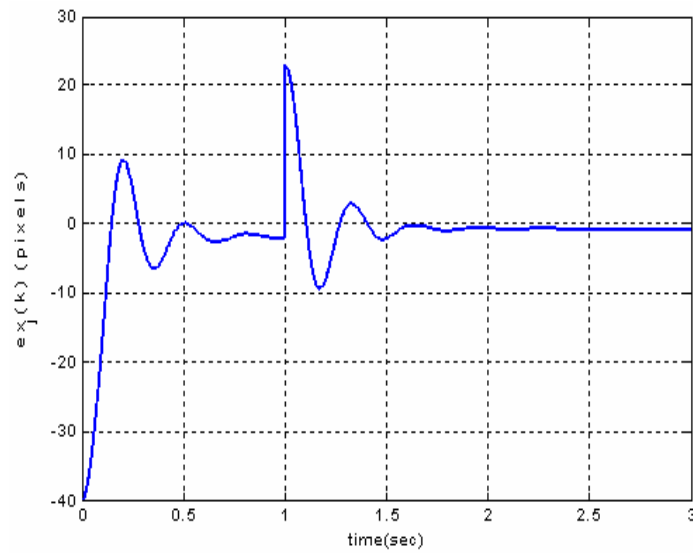


Figure 4.59: Behaviour of error in the x axis as the object is tracked.

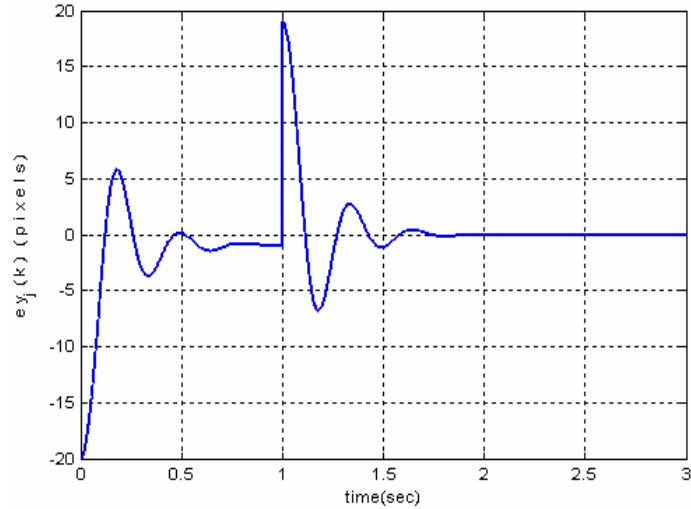


Figure 4.60: Behaviour of error in the y axis as the object is tracked.

The hump shows the time when the target moved i.e. after 1 sec. A plot of the output w.r.t. the desired target positions in the x axis is shown in figure 4.61.

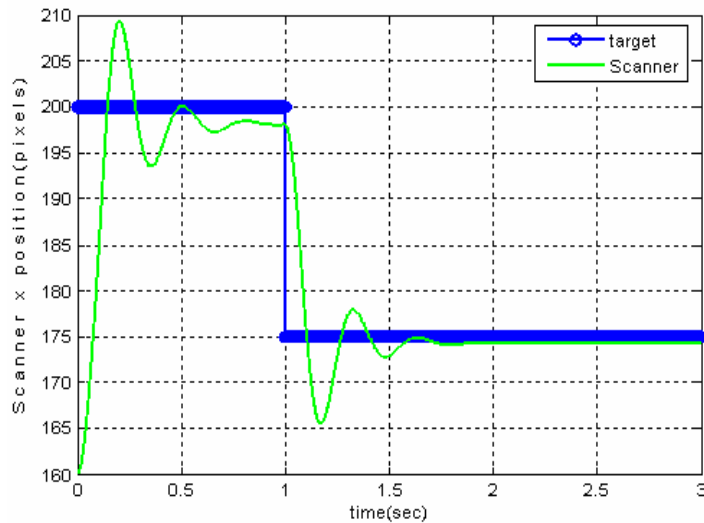


Figure 4.61: Position of Scanner in x axis as the object is tracked.

The trajectory taken by the Scanner as the moving target is tracked is shown in figure 4.62 below. The figure clearly shows the real time tracker converging at the target.

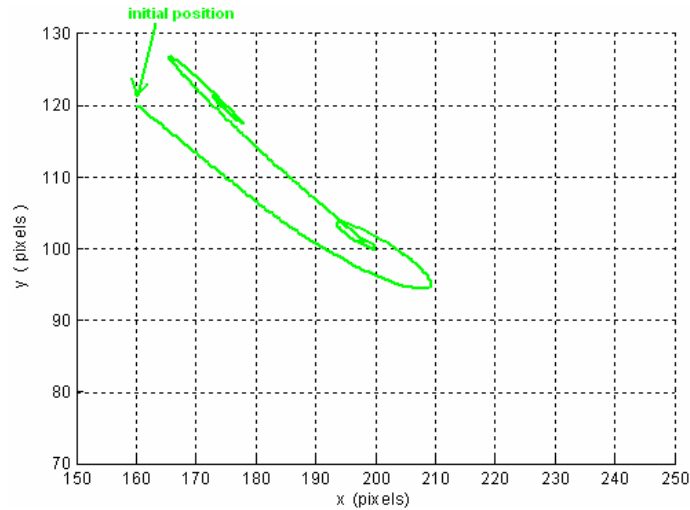


Figure 4.62: Trajectory of the Scanner in the XY plane.

Moving targets were effectively tracked by the ILFT.

Researchers over the years have developed some innovative 2DOF devices like [3,4] where they propose to put both actuators in the base. Camera on the other hand has also been used by some [87,88] to track objects like human head. The camera can also have zoom facility to increase range and reduce error. We now describe the construction of such a device. This device can, not only be used to test ILFT but also opens up many industrial and commercial applications.

#### 4.5.2 Experimental setup

A practical setup was developed to test the performance of the controller in real word situation. The practical setup described here consists of 3 blocks, a target capturing and processing block, a controller and an electro mechanical mechanism to manipulate the position. The proposed tracking system consists of a camera and Image processing module for capturing and processing. It uses the ILT based controller discussed previously as a controller and uses a device called Scanner S-101 for mechanical movements. The S-101 was specially manufactured as a verifying tool for these experiments.



### **4.5.2.1 Real time tracking system**

The constructed real time tracking system consists of a device called Scanner S-101, an interface card, a supply unit, a camera, simulated target board and control software. It is able to scan a predefined area for the presence of a target. It has a positioning accuracy of  $\pm 1$  pixels. This means that for a fixed object position the device is able to move at least  $\pm 1$  pixels in either direction of the target. A webcam is used to capture the image of the area and that image is processed to identify the target.

The S-101 Scanner is a two degree of freedom (2 DOF) device. Travel ranges are  $\pm 90^\circ$  in both the directions from its centre point. Movements in two degrees of freedom are accomplished using 2 DC mini motors, which are connected to the interface card through a connector. Limit switches are attached to the mechanical part to limit the movement of the Scanner. A reed switch and magnet mechanism is used to help the platform move to its initial position. The initial position is the centre position of the XY plane. The status signals from the limit switches and the reed switches are read through the same interface card.

The Scanner is controlled by the ILT based controller discussed earlier. The interface card accepts commands via a parallel communication link from the host PC. On command, pulse width modulated signals are passed to the motors.

### **4.5.2.2 The S-101**

The S-101 weighs about 655gms without the camera. It is shown in figure 4.63.

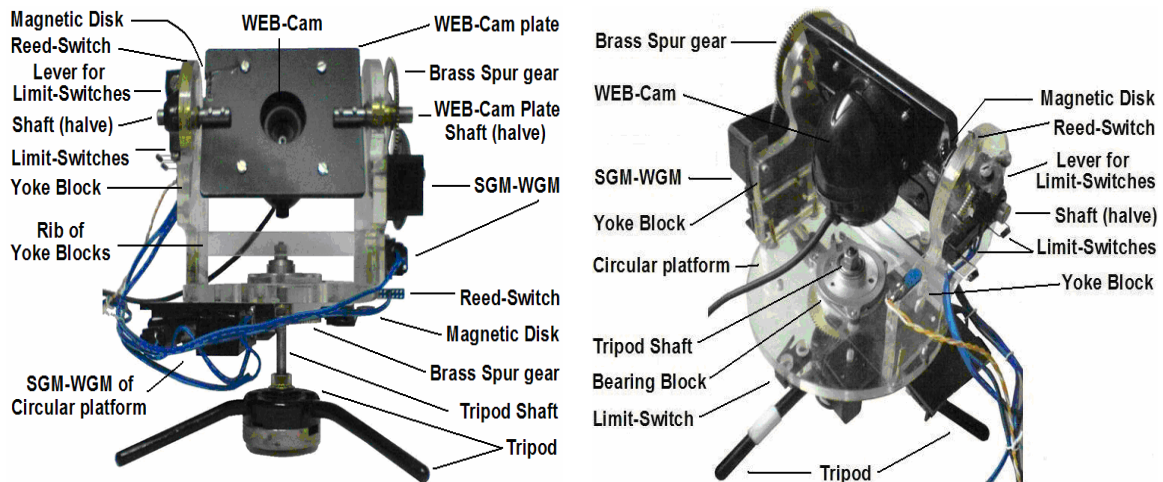


Figure 4.63: The Scanner S-101.

Major construction of the S-101 was carried out with 8mm Plexiglas (Acrylic glass) sheet. The main structure is attached with a circular platform ( $\Phi$  123mm). Two round armed yoke brackets of Plexiglas were also fitted on this circular platform using self threading screws. These brackets are 100mm apart and parallel to each other. The yoke brackets hold the WEB-Cam plate (80x93x6mm). The WEB-Cam plate holds the camera. To assemble this structure a shaft ( $\Phi$  8mm x 153mm) was machined into the WEB-Cam plate. Four 3.5mm holes were drilled into the shaft and threaded with 3mm threading tap. The WEB-Cam plate was screwed in the holes with four 3mm countersunk screws. After proper fixing and marking, the plate was unscrewed from the shaft. The shaft was then cut in such a way that 80mm portion was removed from the middle. The WEB-Cam plate was fixed firmly on the shaft again. The shaft preparation sequence and WEB-Cam plate fixation is shown in figure 4.64. This procedure helps to achieve coaxial alignment of both halves of the shaft.

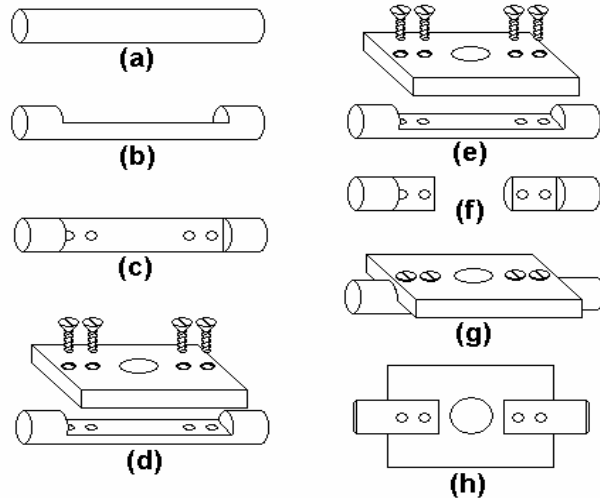


Figure 4.64: ( a= shaft, b= groove making, c= drilling and threading, d=fixing of the WEB-Cam plate, e=dismantling, f=cutting into two halves, g=reassembling, h=actual configuration )

One half of the WEB-Cam plate shaft was fitted with a brass spur gear while the other was fitted with lever for limit switches.

Two spur gear modules (SGM) were fabricated with commonly available gears from used VCRs. The SGMs are derived from worm gear module (WGM). Worm gears are used as driver gears to increase power and reduce speed of the motors. A gear box designed with worm gears have a considerably smaller volume than that designed from spur gears alone. Worm gears also integrate the torque value and have self locking or self braking property which is essential to reduce skew and to give more control. The SGM and WGM are independently adjustable to help reduce the latch.

The gear ratio between the SGM and brass spur gear is 15:1 giving a reduction in speed by a factor of 15 while increasing the torque by the same amount.

The Scanner has a tripod stand for better balance. To keep the centre of mass lower, for balance, the tripod assembly is metallic. This helps the Scanner to keep its ground as it moves while tracking a target. Connected to the tripod centre is a shaft on which the whole assembly is mounted. The shaft was welded vertically to a tripod. The free end of the shaft was machined for 6M threads. A brass spur gear was fixed to the shaft at a desired height. A bearing block taken out from an old floppy drive was fixed in

the middle of the circular platform. The circular platform also holds a gear module similar to the one attached with the yoke bracket. When the WGM rotates the circular platform rotates. All the metallic gears used in the construction were machined to reduce weight by reducing the thickness of the gears and cutting holes in them.

The two yoke blocks were fitted with a rib for rigidity. The two yolk blocks were provided with plain bush bearing for the 8mm shaft of the WEB-Cam plate. The plain bush bearing are light weight and give jerk free movement. A M.S. sheet mounting is fabricated to fix the camera on the WEB-Cam plate.

The SGM consists of both metallic and plastic gears to compensate for spur gear back lash as well as to give better performance even with less lubrication. The metal gears are of  $\Phi 53$  mm and  $\Phi 47$  mm. Holes are cut through the metal gears to reduce weight. The two gear box assemblies are shown in figure 4.65.

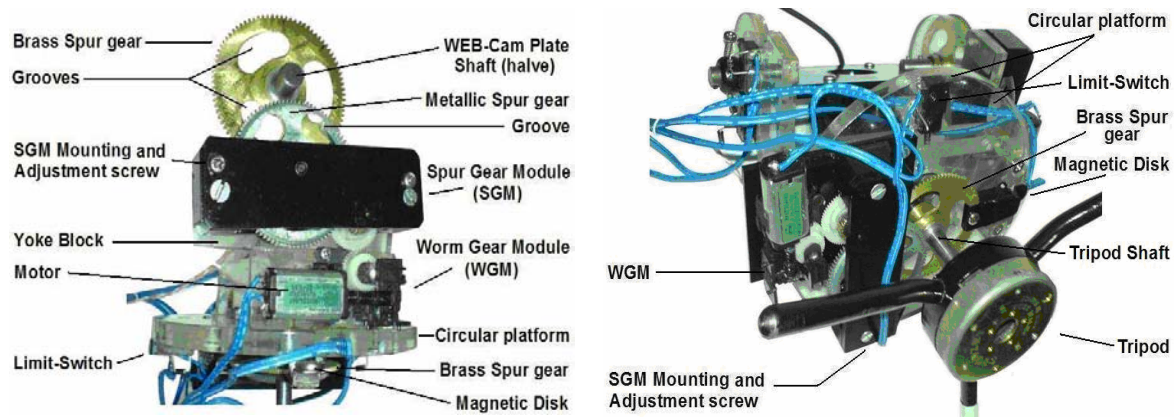


Figure 4.65: Gear boxes assemblies.

Two limit switches are used, for each degree of freedom to limit the movement. The limit switches cut off the supply immediately to protect the mechanical system. The limit switches and the associated electrical circuit is shown in figure 4.66.

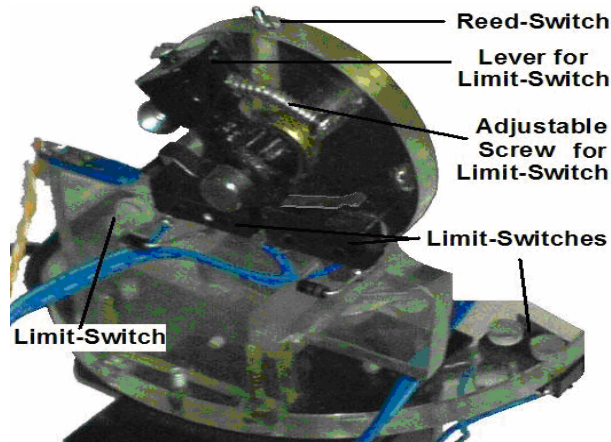


Figure 4.66: Limit switches to protect the assembly.

The compact Scanner is capable of a wide range of rotational movements with great accuracy. Two MITSUMI M15E-2 DC mini motors are mounted at the platform. These DC mini motors have a voltage range of 1.6 to 8.5 V and a no load rotational speed of 9100rpm. A 40mA current at no load produces a torque of 3.92 mNm. These motors have dimensions of only 26.5x 12mm keeping the total assembly weight and size small.

The Scanner also has a mechanism to initialize itself. Two reed switches and bar magnet assemblies are positioned so as to generate signals just at the moment when base plate and the WEB-Cam plate are at the centre. The components are shown in figure 4.67.

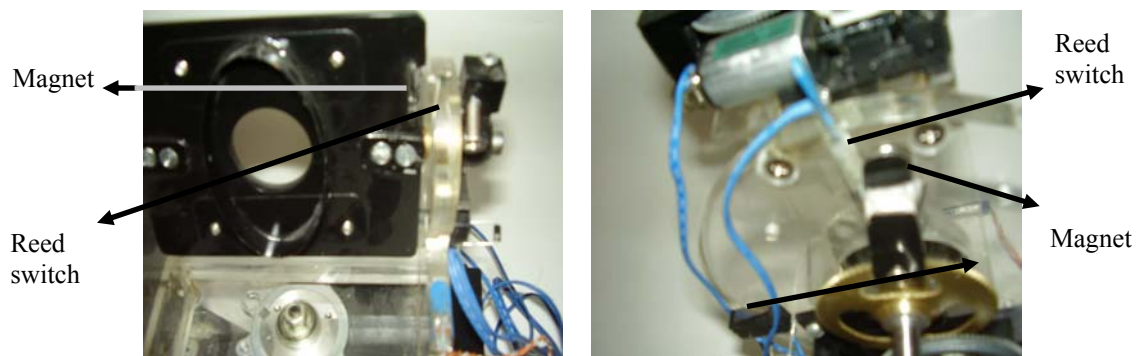


Figure 4.67: Reed switch and magnet assembly.

The camera used is a normal webcam (Creative Live pro) which has a weight of 300 gms.

### 4.5.2.3 Target Simulation Board (TSB)

For target generation, a Light Emitting Diode (LED) based board was developed. Thirty six LEDs of red colour were placed on a wooden square board at regular intervals to simulate potential targets. The target board also has a green LED at the centre for quick Scanner initialization. This initialization support is in addition to the reed relay and magnet structure present in the S-101. The target board also has a switch board (remote console) to switch on the LEDs. The diagram of the target board is presented in figure 4.68.



Figure 4.68: Target Simulation Board.

### 4.5.2.4 Interface card

Communication between PC and S-101 was done through the Interface card. The card also supplied the power to the S-101. A picture of the card is described in figure 4.69.

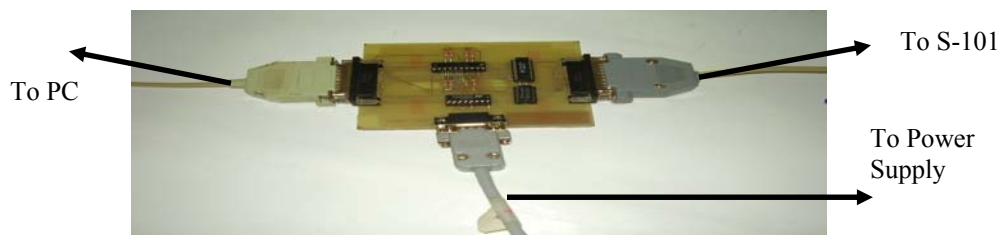


Figure 4.69: Interface card.

### 4.5.2.5 Scanner coordinate system

The two degrees of freedom of the scanner are explained in figure 4.70 below.



Figure 4.70: Geometrical coordinates of the S-101.

The origin of the coordinate system XYZ, is located at the intersection of the centre of the base disc and the centre of the camera. This is the initial position of the S-101. The  $w$  rotation is called the yaw rotation and the  $v$  rotation the pitch.

### 4.5.2.6 Control software

The control software was written using MATLAB. The camera took software trigger based pictures regularly. The status signals from the limit switches and also from the reed switches were read using the parallel (LPT) port. The same port was also used to give commands to the motors. A Graphical User Interface (GUI) was also developed. It is exhibited in figure 4.71.

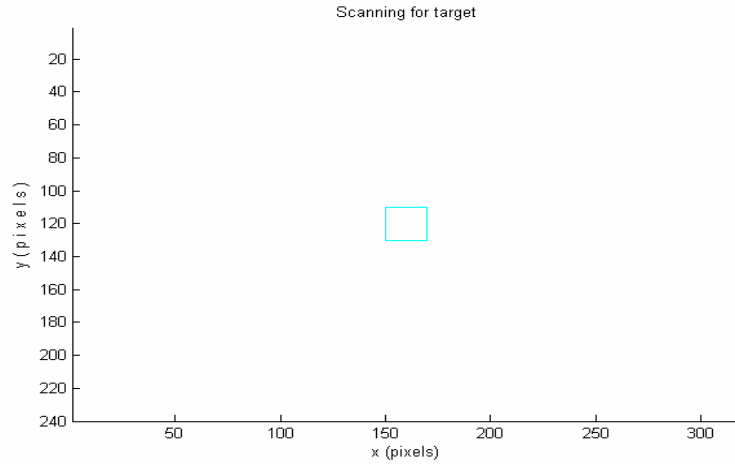


Figure 4.71: GUI of the software.

The GUI shows the camera pixels both in x and y directions. This snapshot was taken with a camera resolution set at 320x240 pixels. Any object detected by the image processing module will be displayed in this GUI. The small rectangular block marks the locking area. An object with in this area is considered locked.

### 4.5.2.7 The complete setup

The complete setup is shown in figure 4.72.

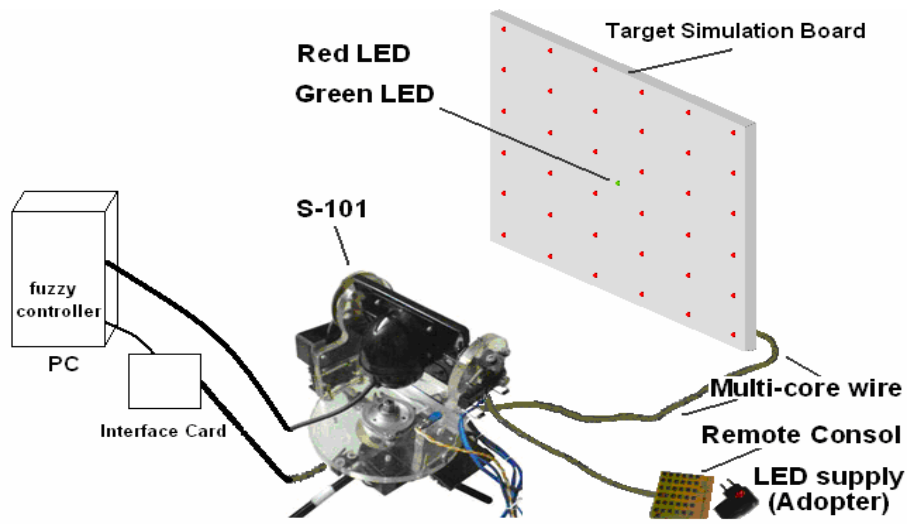


Figure 4.72: Real time tracking system in operation.



The S-101 is facing the target simulation board. The board has its own remote console to simulate target movement. The S-101 is connected to the PC through an interface card. The camera is also connected to the PC. This camera closes the feedback loop of the system.

#### 4.5.2.8 Experiments using 3 input and 3 output MFs

In this experiment, 3 MFs were used to define both errors ( $e_{x_j}(k), e_{y_j}(k)$ ). The membership functions are shown in figure 4.73 and 4.74.

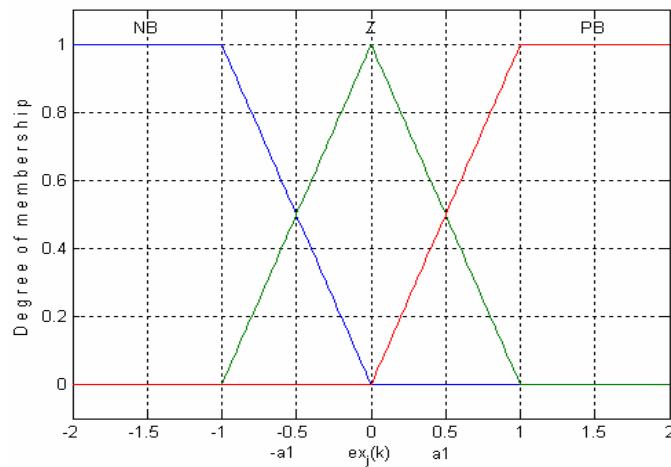


Figure 4.73: Input MFs for FLC 1.

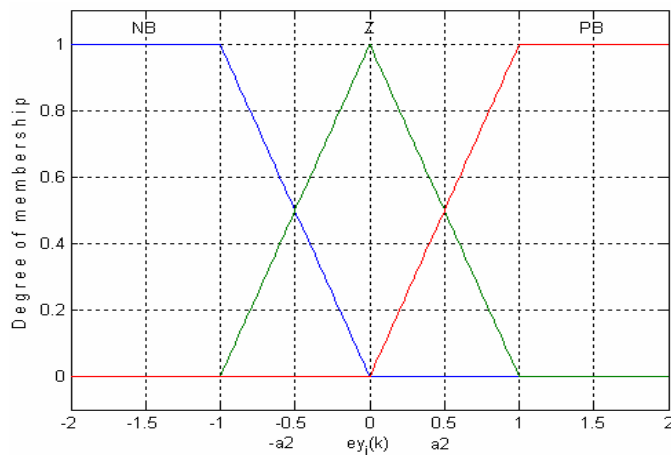


Figure 4.74: Input MFs for FLC 2.

The output membership functions for both the FLC blocks are defined in figure 4.75.

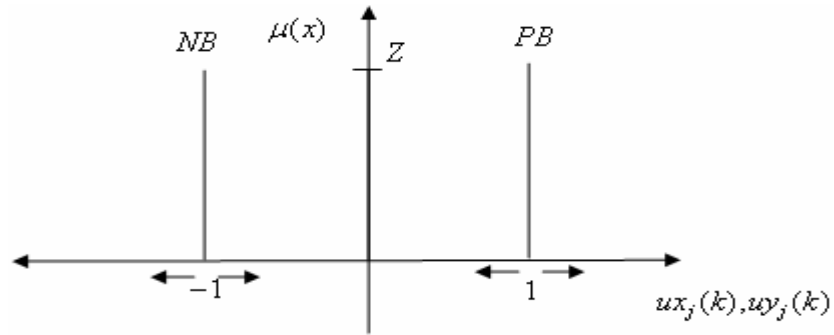


Figure 4.75: Output Membership functions for both FLC 1 and FLC 2.

The ILT is programmed to find values of  $a_1$  and  $a_2$  (see figures 4.73 and 4.74) such that the system can track the target within  $\pm 3$  pixel resolution at less than 1 sec. The ILT learns the values of  $a_1$  and  $a_2$  for ensuring the desired performance requirements. From starting values of  $a_1 = 1.0$  the system learns the values  $a_1 = 0.52$ . The learning history of parameter  $a_1$  is shown in figure 4.76.

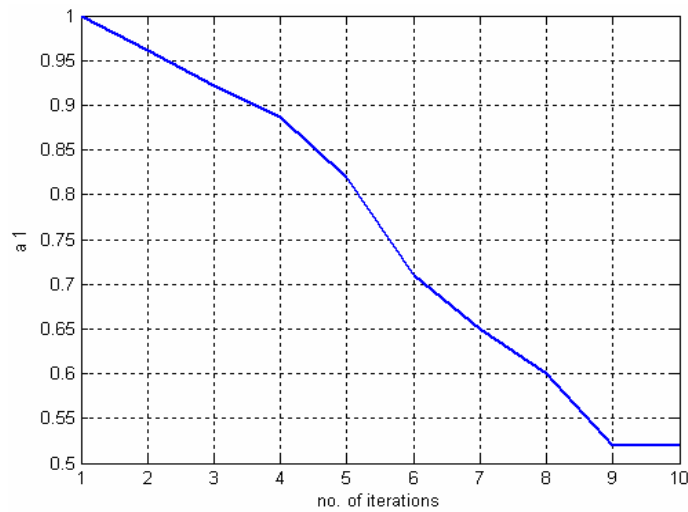


Figure 4.76: Learning behaviour of  $a_1$  as iterations increase.

After learning  $a_1$  the S-101 is made to track a target, the tracking performance is shown in figure 4.77.

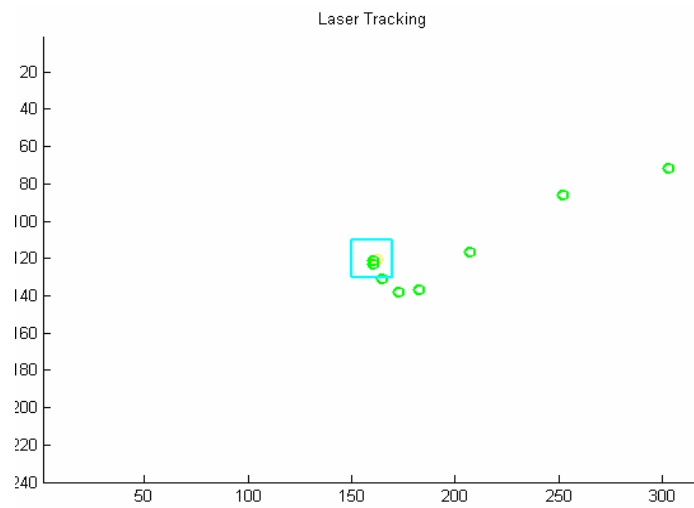


Figure 4.77: Snap shot of the target being tracked by S-101.

There is over shoot in the y axis. To reduce over shoot,  $a_2$  was learnt. The learnt value for acceptable overshoot was  $a_2 = 1.57$ . The learning history of  $a_2$  and the response with these values is presented in figure 4.78 and 4.79.

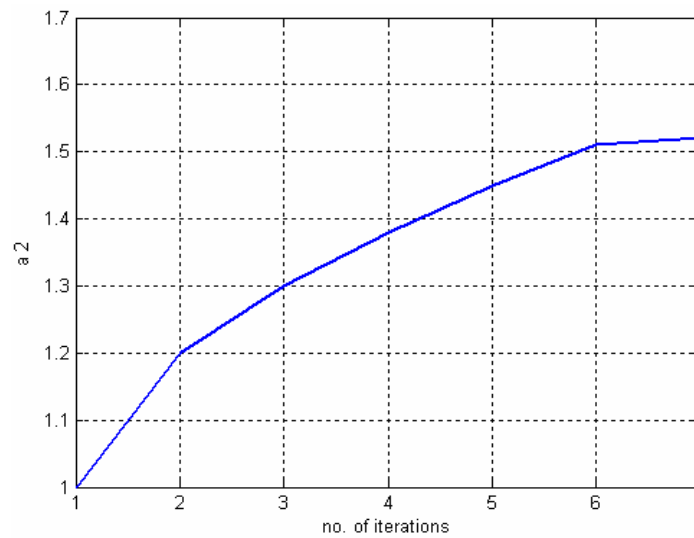


Figure 4.78: Learning behaviour of  $a_2$  as iterations increase.

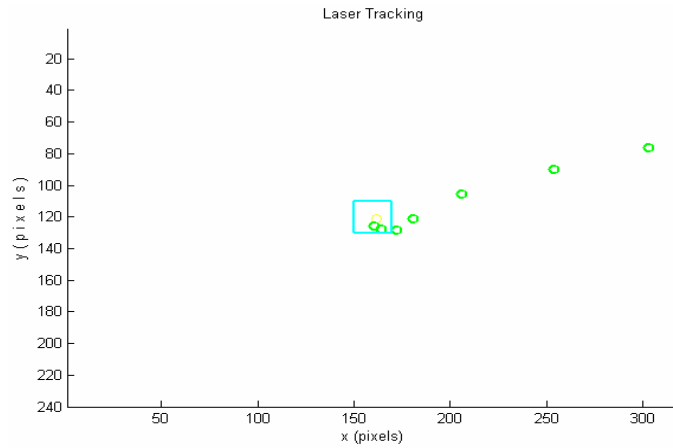


Figure 4.79: Snap shot of the target being tracked by S-101.

The plot of errors in x and y positions of the scanner while tracking the target is shown in figure 4.80.

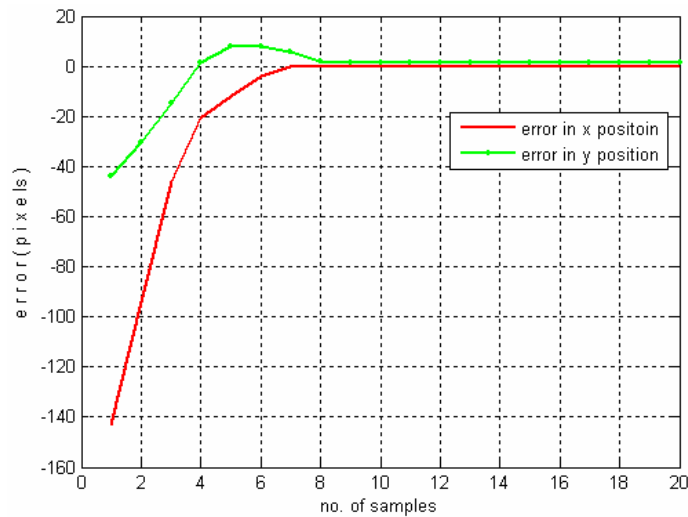


Figure 4.80: Plot of the errors as target is being tracked.

The plot shows that the error settles to within  $\pm 3$  pixels in both the x and y axis. For a frame rate of 20 frames per second (no. of frames is labelled no. of samples in the figure) the S-101 was able to track the object within 0.5 sec.

The control surfaces learnt for the two FLC blocks are shown in figure 4.81 below.

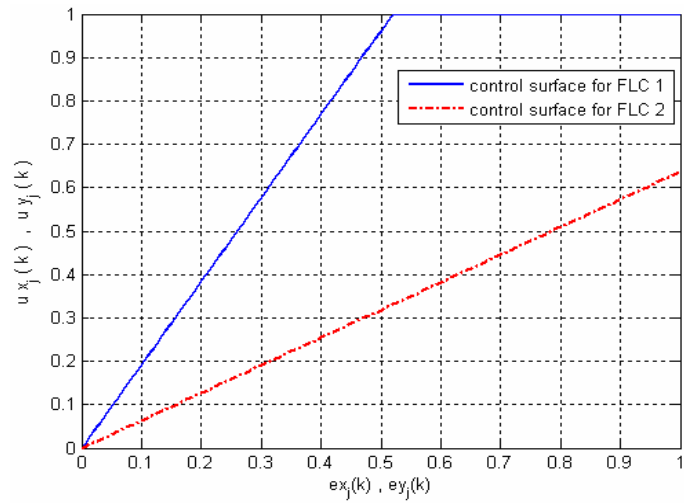


Figure 4.81: Control surface of the two FLCs after learning.

The input membership functions for FLC1 and FLC2 after learning are given in figure 4.82 and 4.83.

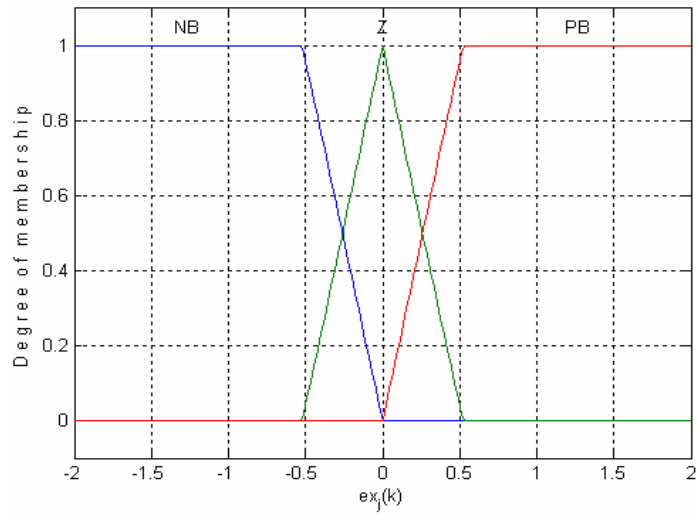


Figure 4.82: Input MFs for FLC 1 after learning.

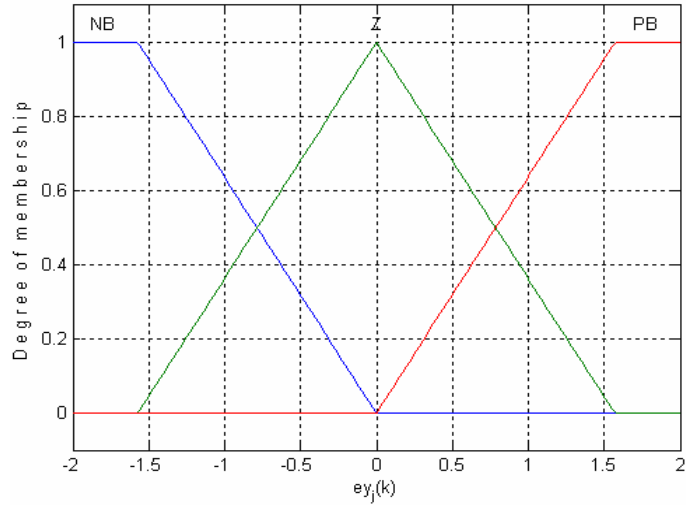


Figure 4.83: Input MFs for FLC 2 after learning.

The ILT based controller successfully tracked moving targets also. Figure 4.84 shows the GUI as the target is moving and as it is being tracked. The trail is intentionally left visible so that we can see the performance. The target was initially at position 1, it then moved to position 2, then to position 3 and ultimately to position 4.

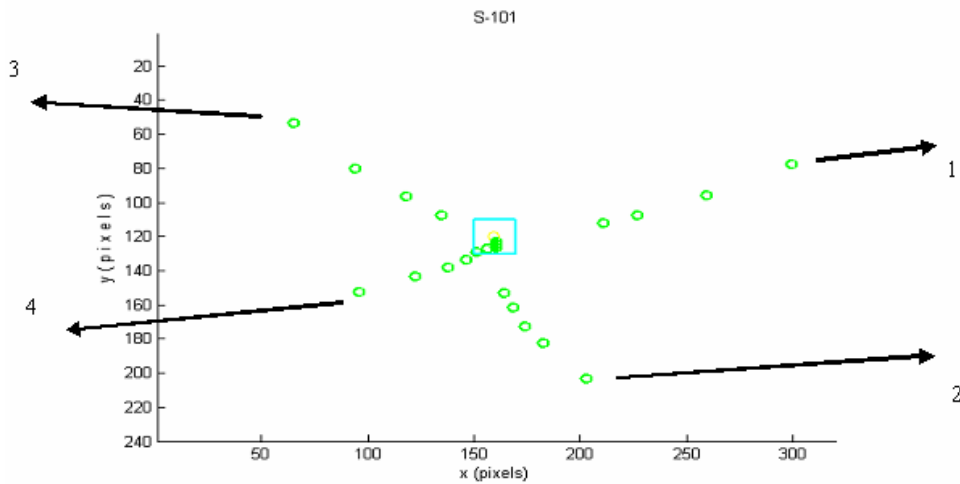


Figure 4.84: Moving target being tracked.

The plot of errors in both dimensions as the target is tracked is shown in figure 4.85. Plot shows that the Controller is able to successfully track moving objects.

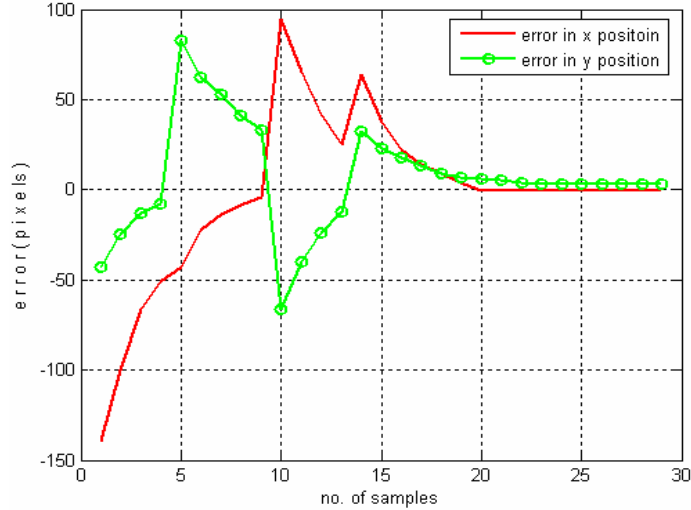


Figure 4.85: Plot of the error as target is being tracked.

Using a frame rate of 20 frames per sec. the device showed good accuracy and speed while tracking a moving target.

#### 4.5.2.9 Experiments using 7 input and 7 output MFs

Another set of experiments used the same physical setup, with 7 input and 7 output membership functions for both FLCs. The membership functions were of the form in figure 4.14. The design requirements were tightened with less than 2 pixel steady state error and with an overshoot of less than 7 pixels. The ILT learnt the values of  $ax_j(5)$ ,  $ax_j(6)$ ,  $ay_j(5)$  and  $ay_j(6)$  as  $ax_j(5) = 0.1545$ ,  $ax_j(6) = 0.6823$ ,  $ay_j(5) = 0.3$  and  $ay_j(6) = 0.83$ . It took 3 iterations to learn  $ax_j(5)$ ,  $ax_j(6)$  and  $ay_j(6)$ . The learning history plot of these parameters is shown in figure 4.86 and 4.87.

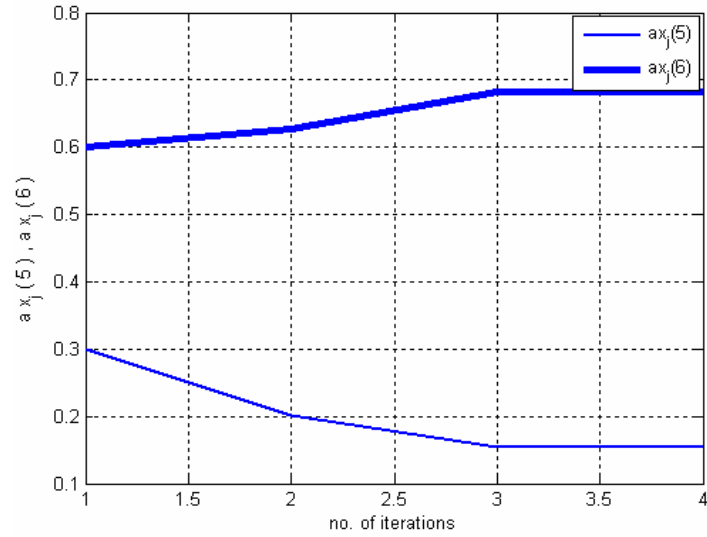


Figure 4.86: Learning behaviour of  $ax_j(5)$  and  $ax_j(6)$  as iterations increased.

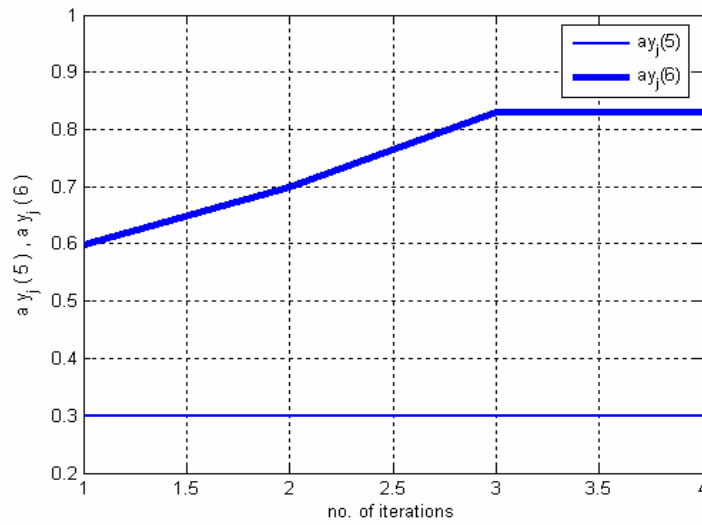


Figure 4.87: Learning behaviour of  $ay_j(5)$  and  $ay_j(6)$  as iterations increased.

Figure 4.88 shows a GUI snap shot of a target being tracked by S-101, using 7 input 7 output MFs.



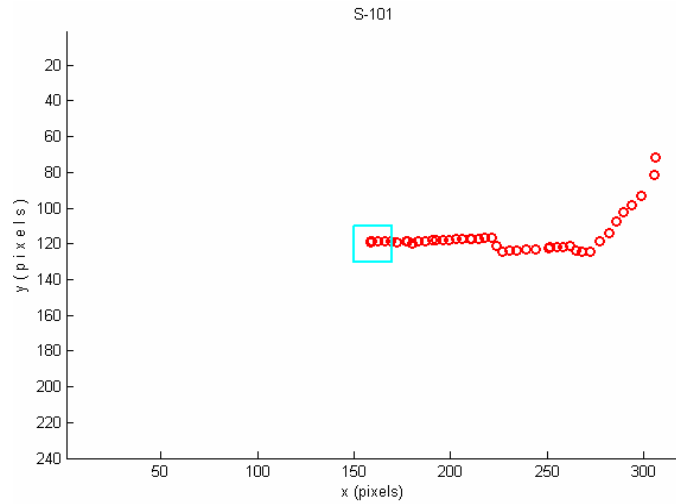


Figure 4.88: Snap shot of the target being tracked.

The camera frame rate was set at 20 frames per second. A 3D plot of the behaviour of S-101 as the target is being tracked is shown in figure 4.89.

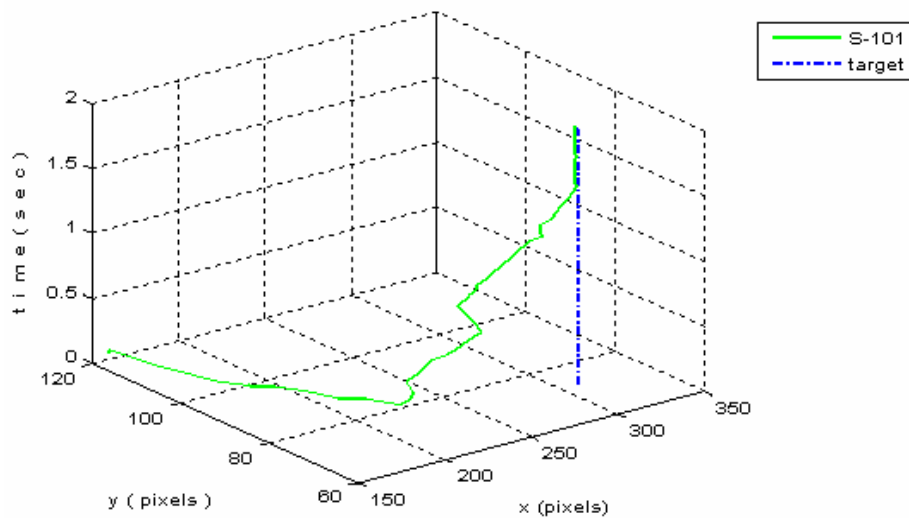


Figure 4.89: Response of S-101 against a static target.

The two dimensional plot as tracking was achieved in both x and y directions against time is presented in figures 4.90 and 4.91. The plots show in greater detail, the behaviour of S-101.

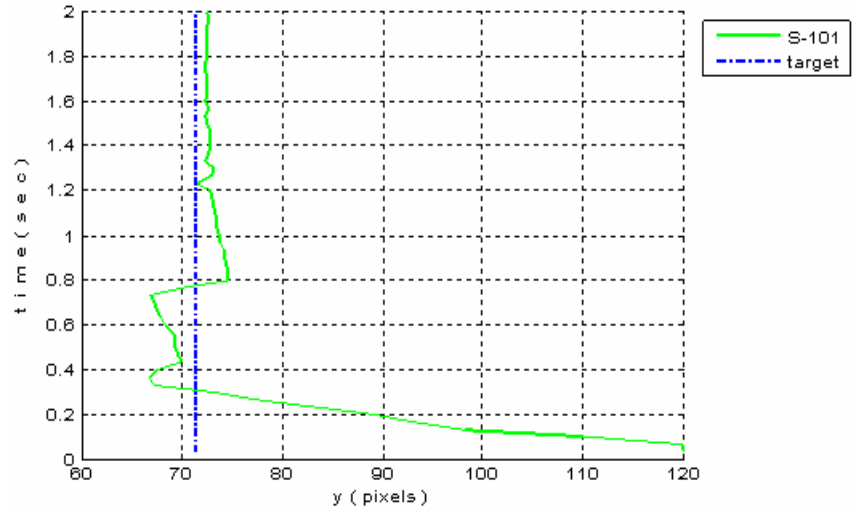


Figure 4.90: Position of S-101 y axis as the object is tracked.

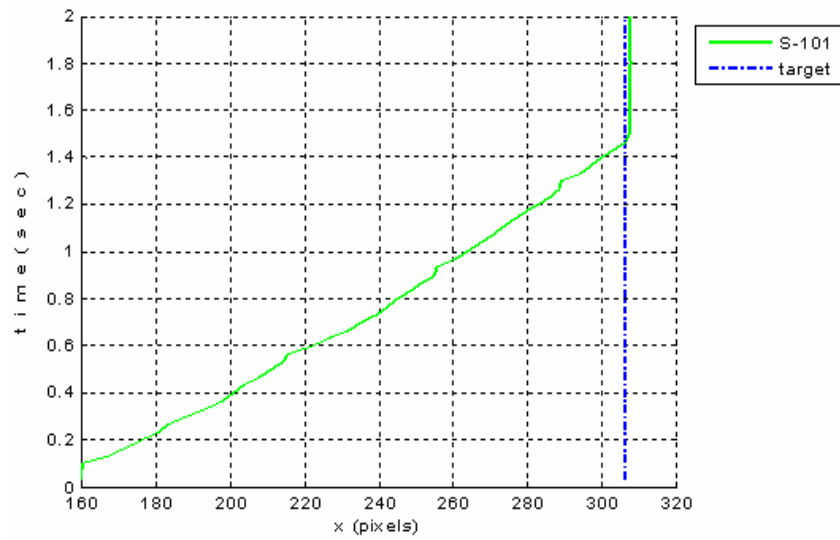


Figure 4.91: Position of S-101 along x axis as the object is tracked.

For both degrees of freedom, the elimination of error is plotted in figure 4.92.

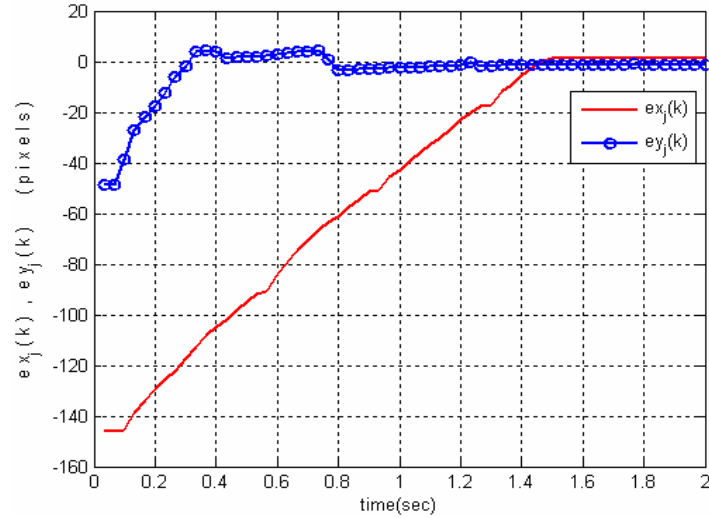


Figure 4.92: Plot of the errors as target is being tracked.

Moving targets were also tracked successfully with this setup. The response of S-101 while trying to track a moving target is shown in figure 4.93. Again the circular marks, showing the target, are intentionally left visible to show the tracking performance.

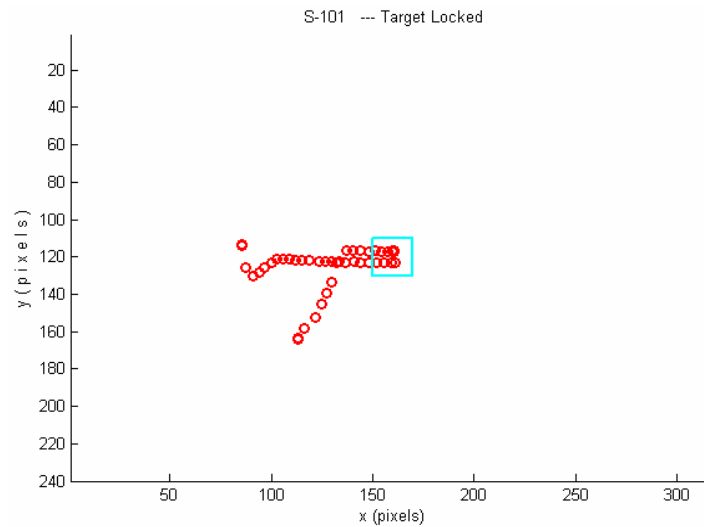


Figure 4.93: Snap shot of the GUI while tracking a moving target.

A 3-D plot, as S-101 tries to track this moving target is shown in figure 4.94 and its corresponding error plot is shown in figure 4.95.

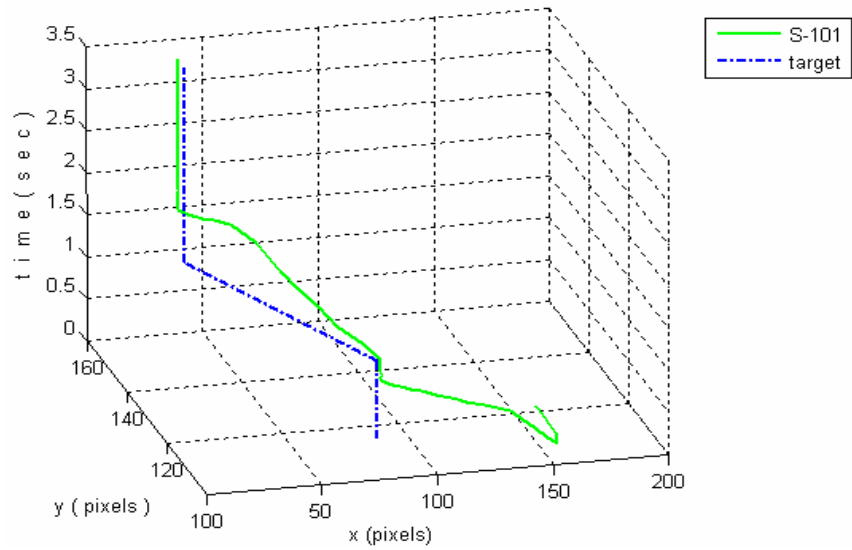


Figure 4.94: Response of S-101 against a moving target.

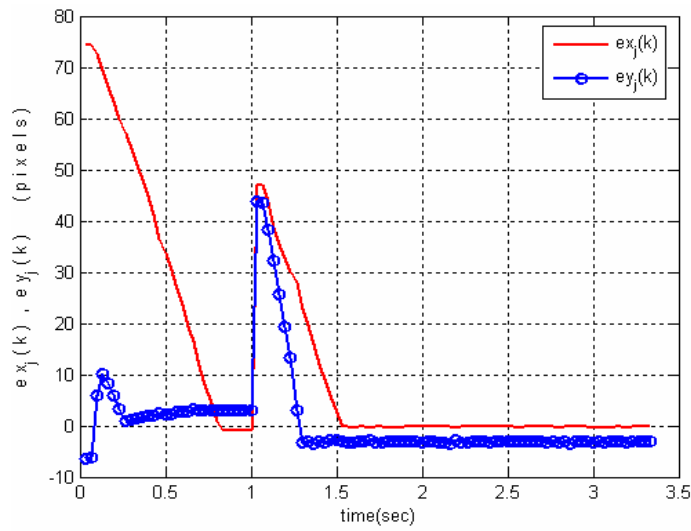


Figure 4.95: Plot of the errors as target is being tracked.

For a closer look at the performance, figure 4.96 shows a 2-D view of the S-101 tracking the target. It shows S-101 movements in x-direction only.

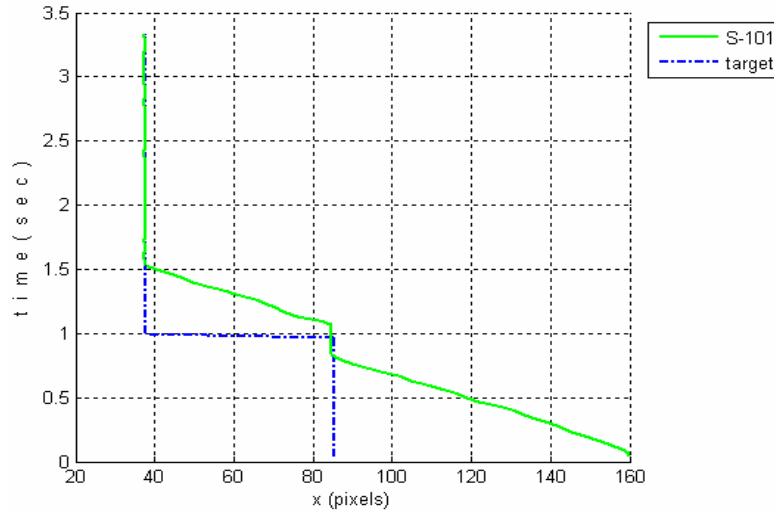


Figure 4.96: Position of S-101 along x axis as the object is tracked.

This result was taken at 10 frames per second. The tracking performance can be increased by increasing the frame rate. It was observed that 20 frames per second produced the best results.

## 4.6 Summary

Zadeh in his foreword remarks in [101] points out that the issue of key importance in the design of fuzzy controller is the tuning of controller parameters and the induction of rules, which are mostly done by trial and error. This tuning is more important because of the uncertainties associated with the linguistic variables, upon which the fuzzy controllers are based. Use of type-1 fuzzy set to model these linguistic variables is scientifically incorrect because a word is uncertain while a type-1 set is certain [72].

To tackle these uncertainties, type-2 FLCs are being developed [31] but type-2 systems are very complex to perceive and very difficult to implement. Ideally, we have to use type- $\infty$  fuzzy sets to represent uncertainty, in totality [73].

This chapter proposes to tackle linguistic uncertainties through adaptability in membership functions, thereby still remaining in the framework of Fuzzy type-1. This adaptability is achieved using an Iterative Learning Fuzzy Tuner (ILFT). The Controller

contains an Iterative Learning Tuner (ILT), which iteratively tunes the Membership Functions (MFs). This results in an adaptive rule base. This ILFT has the capability to achieve desired steady state error and percentage overshoot design requirements. Results from different linear and non-linear systems are presented.

A universal reliable method to determine the stability of a fuzzy system is still not available. By dividing the error into different ranges, a novel method for stability and convergence is discussed. Stability and convergence is also discussed using linguistic trajectories.

The ILFT is made to track different practical trajectories with excellent results. The controller can take derivative and integral of errors as input as well. These inputs should only be used if the design requirements are not met using error alone. A step by step procedure for the selection of number of inputs, for the controller, is also presented in this chapter.

For a practical setup, a Tracker called S-101 was developed. The device has two degrees of freedom and is a low cost alternative to the six degrees of freedom Hexapod, used earlier in chapter 2. It has a camera mounted on it for target recognition. Using this tracker the ILT based controller was able to track targets with in the desired accuracy level in real time. To achieve exact replication of target path, a Target Simulation Board (TSB) was also developed.

The simulation and practical results show that because of inherent non-linearity in fuzzy systems, we were able to learn non-linear control surfaces even with a single input single output controller. If there is any change in plant parameter, desired steady state error or desired percentage over shoot, the ILFT has the ability to readjust.

Though fuzzy logic has contributed in thousands of applications, the most used controllers are still Proportional Integral (PI) and Proportional Integral Derivative (PID). The problem with these conventional controllers is the requirement of a mathematical model and single point operation excellence. Fuzzy logic, with the learning capability, developed in this chapter, can be used to schedule the gains of the PI and PID controllers. This will make the controller performance independent of the model and will be able to operate at different operating points.

The next chapter develops and tests such an approach.

## **5 ITERATIVE LEARNING FUZZY GAIN SCHEDULER**

A simple yet robust alternative to Proportional-Integral-Derivative (PID) controller is developed using fuzzy based gain scheduler. The fuzzy based system is further tuned using an iterative learning approach.

Gains of a conventional PID controller are usually fixed. This results in a control surface which is some times unable to meet our design requirements. Designing fuzzy controllers with desired performance specifications is not a trivial task either. Even the specification of linguistic variables, key to the concept of fuzzy design, can be different from different experts. This chapter lays out an adaptive procedure for designing fuzzy controllers through iterative learning process to schedule gain values.

### **5.1 Introduction**

The best known and most used controllers in industrial control processes are proportional-integral (PI) and proportional-integral-derivative (PID) controllers. Designing and implementing these controllers have difficulties associated with them [29], namely:-

- (a) It is usually based on accurate mathematical model of the system which is usually not known.
- (b) Variation of plant parameters can cause unexpected performance variations.
- (c) They usually show high performance, for one unique action point.

Extensive efforts have been devoted to develop methods to reduce the time spent on optimizing the choice of controller parameters like proportional gain, integral gain and derivative gain of these controllers [7]. The PID controllers in the literature can be divided into two main categories. In the first category, the controller parameters are fixed after they have been tuned or chosen in a certain optimal way. The parameters of the

controllers of the second category are adapted, based on some parameters estimation technique, which requires certain knowledge of the process. In most practical cases, the model of the system is not known and hence conventional PID control schemes can not achieve high performance values. Also, the dynamics of a system even for a reduced mathematical model is usually non-linear, making tuning of these controllers even more difficult [16]. Fuzzy logic based control [92, 93, 94] has been shown in numerous studies to be a simpler alternative to conventional PID control [70, 102, 104, 136].

For conventional PID controller design, gain scheduling is often effectively used to give some adaptivity. In conventional gain scheduling (CGS), the controller parameters are scheduled according to some monitored conditions in an open-loop fashion. Its main advantage is that controller parameters can be changed quickly. One serious drawback of CGS is that the parameter change may be rather abrupt, which may result in unsatisfactory or even unstable performance across the transition region. Another problem is that accurate models at various operating points may be too difficult or even impossible to obtain. As a solution, fuzzy has been utilized for gain scheduling to overcome these problems [104, 132]. Other proposed techniques use fuzzy rule base formulation [103][52], neural networks [132] and membership function definitions for PID gains [154]. Still others have simplified existing fuzzy PID schemes with a gain scheduling differential equation [136].

Even with fuzzy based scheduling, the choice of appropriate membership functions, minimum rule base and suitable fuzzifier and defuzzifiers is still a challenging task. Having made these choices, one still needs to tune the fuzzy controller to deliver the desired response. Multiple simultaneous adjustments (rules, membership functions and input/output fuzzy gains) make the optimum tuning even more difficult. Many techniques have been used to overcome this difficulty, including a phase plane technique for rule base design [62], rule modification [63], neural network techniques [29], genetic algorithms [2] and gain phase margin analysis technique [76].

Before any rules can be formulated, membership functions need to be sorted out. But, as discussed in chapter 4, membership functions have uncertainties associated with them. With such uncertainty it is difficult to determine the exact Membership functions (MF) for a fuzzy system (FS) which can give desired performance.



Apart from determining the gains of the P, PI and PID controllers this scheme also tackles this uncertainty using a learning approach to adaptively adjust the membership functions. This iterative learning process is further linked with steady state error and overshoot, which are used to specify design requirements. This helps the controller to adjust automatically, when the plant parameters vary or performance requirements change.

Iterative learning control (ILC) has been successfully used in tasks where the process is repetitive [5, 105]. In this proposed approach we make the controller gains adaptive by adjusting the membership functions using learning laws. These learning laws indirectly adjust the control input to the plant.

The hallmark of our proposed approach is the Iterative Learning Fuzzy Gain Scheduler (ILFGS). It consists of a fuzzy system, iterative learning laws to adjust member ship functions and a mathematical formula to calculate controller gains. The approach is tested through simulation and a motor speed control experiment, using Quanser’s DC Motor Control Kit.

## 5.2 Proposed Approach

A typical block diagram of a conventional proportional-integral (PID) controller using 2-D representation is describe in figure 5.1.

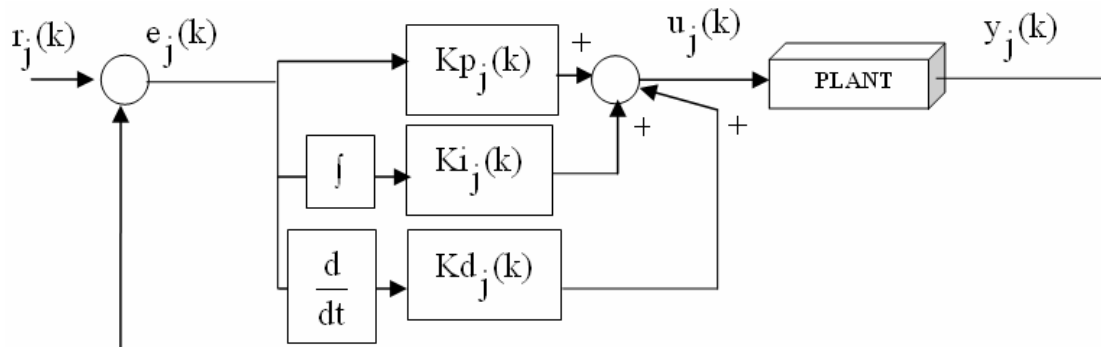


Figure 5.1: Block diagram of a PID controller.

Here  $r_j(k)$  represents the reference signal for  $k=1\dots N$  and  $j=1\dots\infty$ . Variable  $j$ , represents the iteration number and variable  $k$ , represents the samples. The error is represented by  $e_j(k)$ , the input to the plant is  $u_j(k)$  and the next plant output is  $y_j(k)$ . Gains  $Kp_j(k)$ ,  $Ki_j(k)$  and  $Kd_j(k)$  are in most cases fixed but they can be adaptive also. These gains are chosen such as to meet our design requirements of steady state error (sse) and percentage over shoot (pos). Usually, in order to figure out  $Kp_j(k)$ ,  $Ki_j(k)$  and  $Kd_j(k)$  correctly, we have to have some knowledge of the plant. If there is any change in plant parameters or design requirements, recalculation of gains is required.

To overcome these issues we propose an adaptive gain scheduler scheme, the block diagram of which is presented in figure 5.2.

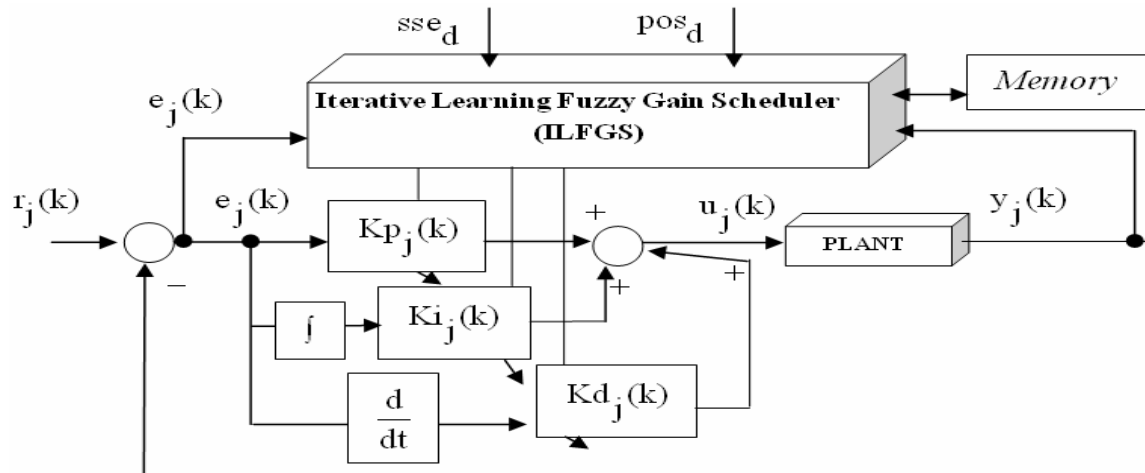


Figure 5.2: Block diagram of the proposed scheme.

Here  $sse_d$  is the desired steady state error,  $pos_d$  is the desired percentage over shoot,  $r_j(k)$ ,  $e_j(k)$ ,  $u_j(k)$  and  $y_j(k)$  are the reference input, error, input to the plant and next plant output at iteration  $j$ . Desired steady state error and percentage over shoot are supplied to Iterative Learning Fuzzy Gain Scheduler (ILFGS). The ILFGS adjusts  $Kp_j(k)$ ,  $Ki_j(k)$  and  $Kd_j(k)$ . It also tunes itself by adaptively adjusting the membership

function end points. The aim is to converge, with respect to, given steady state error and percentage overshoot. The learnt values of membership function end points are stored in memory to be used in future iterations.

Taking a Sugeno type rule processing, the input (error) and output ( $K_p'$ ) membership functions proposed are of the form given in figure 5.3 and 5.4.

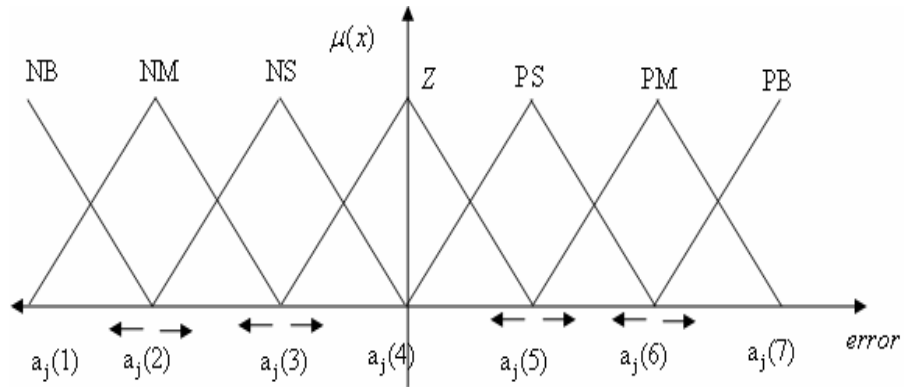


Figure 5.3: Input Membership functions.

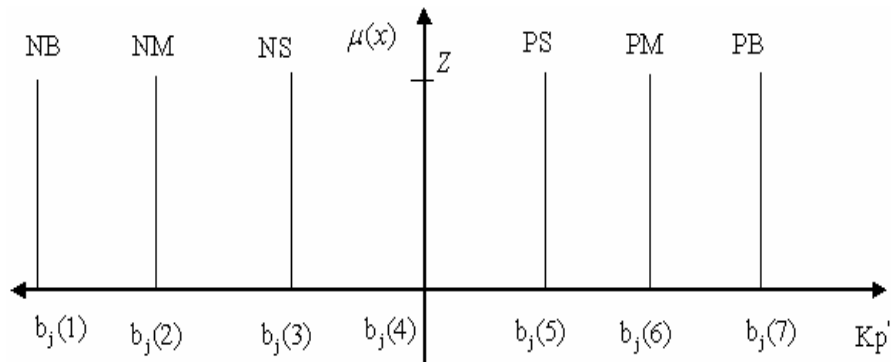


Figure 5.4: Output Membership functions for Sugeno rule processing.

End points of membership functions  $a_j(k)$  and  $b_j(k)$  can be made adjustable to meet steady state error and percentage over shoot requirements.

As discussed in chapter 4, TSK fuzzy system can be viewed as a piece-wise linear function, where the change from one segment to the other is smooth, rather than abrupt. This helps us to overcome the abrupt change in parameters in case of CGS.

The following examples show a method to adjust  $Kp_j(k)$ . Most requirements were met by considering this proportional gain only. If steady state error requirements are not met by considering  $Kp_j(k)$  alone;  $Ki_j(k)$  can also be considered to reduce steady state error further. Similarly, if percentage overshoot requirements are not met by considering  $Kp_j(k)$ , then  $Kd_j(k)$  can also be considered. Without any loss of generality, it is proposed that we start with  $Ki_j(k) = 0$  and  $Kd_j(k) = 0$  initially.

Suppose that the range  $[Kp_{\min}, Kp_{\max}]$  can be determined, where  $Kp_{\min}$  is the minimum and  $Kp_{\max}$  is the maximum value that the proportional gain can have. The values of input and output membership functions can be normalized for convenience. The input membership function end points permissible ranges are tabulated in table 5.1.

End point	Permissible Range
$a_j(1)$	$[e_{\min}, a_j(2)]$
$a_j(2)$	$[a_j(1), a_j(3)]$
$a_j(3)$	$[a_j(2), a_j(4)]$
$a_j(4)$	$[a_j(3), a_j(5)]$
$a_j(5)$	$[a_j(4), a_j(6)]$
$a_j(6)$	$[a_j(5), a_j(7)]$
$a_j(7)$	$[a_j(6), e_{\max}]$

Table 5.1: Permissible ranges of input membership function, end points.

Here  $e_{\min}$  and  $e_{\max}$  are the minimum and maximum values that error can have. The output membership function end points were fixed, as given in table 5.2.

End point	Value
$b_j(1)$	$Kp'_{\min}$
$b_j(2)$	$0.6Kp'_{\min}$
$b_j(3)$	$0.3Kp'_{\min}$
$b_j(4)$	0
$b_j(5)$	$0.3Kp'_{\max}$
$b_j(6)$	$0.6Kp'_{\max}$
$b_j(7)$	$Kp'_{\max}$

Table 5.2: Fixed output membership function end point values.

Here  $Kp'_{\min}$  and  $Kp'_{\max}$  are the minimum and maximum values of the output membership function.

For the proposed scheme, we only move  $a_j(5)$  and  $a_j(6)$ . Though,  $a_j(5)$  and  $a_j(6)$  can take up any initial starting value within their permissible range, it is recommended from research results to start with values that divide the Universe of discourse evenly. One such division is done by  $a_j(5) = 0.3e_{\max}$  and  $a_j(6) = 0.6e_{\max}$ , which divides the range reasonably.

The proposed rule base of the fuzzy controller is shown in table 5.3.

	NB	NM	NS	Z	PS	PM	PB
$e_j(k)$	$e_{\min}$	$a_j(2)$	$a_j(3)$	0	$a_j(5)$	$a_j(6)$	$e_{\max}$
$Kp'$	$b_j(1)$	$b_j(2)$	$b_j(3)$	$b_j(4)$	$b_j(5)$	$b_j(6)$	$b_j(7)$

Table 5.3: Proposed rule base.

The following procedure for the up gradation of  $Kp_j(k)$ ,  $a_j(5)$  and  $a_j(6)$  is proposed.

### 5.2.1 Procedure for the up gradation of parameters

The values of  $Kp_j(k)$ ,  $a_j(5)$  and  $a_j(6)$  should be calculated using the following procedure and sequence:-

1- The iterative learning mechanism first learns the value of  $Kp_j(k)$  until the system is critically damped i.e.

$$\text{pos} \leq \varepsilon \quad (5.1)$$

Where  $\varepsilon$  is some small number and 'pos' stands for percentage over shoot. The learning procedure to learn this value of  $Kp_j(k)$  is:-

if percentage over shoot < 0.0

$$Kp_{j+1}(k) = Kp_j(k) + \mu_{kp} (\|r_j(k)\| - \|y_j(k)\|) \quad (5.2)$$

else

$$Kp_{j+1}(k) = Kp_j(k) - \mu_{kp} (\|r_j(k)\| - \|y_j(k)\|) \quad (5.3)$$

Where,  $Kp_{j+1}$  is the value of gain for the next iteration and  $Kp_j(k)$  is the value of gain for the current iteration. This value of  $Kp_j(k)$  is stored in memory and is called  $Kp_c$  (critically damped proportional gain). Step size parameter for finding  $Kp_j(k)$  is  $\mu_{kp}$ .

2- Change the value of  $a_j(5)$  such that  $sse \leq sse_d$  using the learning law:-

$$\text{if } r_j(k) - y_j(k) > sse_d$$

$$a_{j+1}(5) = a_j(5) - \mu_{sse}(\|r_j(k)\| - \|y_j(k)\|) \quad (5.4)$$

else

$$a_{j+1}(5) = a_j(5) + \mu_{sse}(\|r_j(k)\| - \|y_j(k)\|) \quad (5.5)$$

Here  $\mu_{sse}$  is the step size parameter for correcting steady state error.

**3-** Change the value of  $a_j(6)$  such that  $pos \leq pos_d$  using the learning law:-

if  $pos_j > pos_d$

$$a_{j+1}(6) = a_j(6) + \mu_{pos}(\|r_j(k)\| - \|y_j(k)\|) \quad (5.6)$$

else

$$a_{j+1}(6) = a_j(6) - \mu_{pos}(\|r_j(k)\| - \|y_j(k)\|) \quad (5.7)$$

Here,  $\mu_{pos}$  is the step size parameter for correcting percentage overshoot and  $pos_j$  is the percentage overshoot value for the current iteration. To terminate the learning processes, some small tolerance value,  $\epsilon$ , needs to be defined.

Once the learning is complete, the value of  $Kp_j(k)$  is set by using equation

$$Kp_j(k) = (Kp' * K_{pc})/e_j(k) \quad (5.8)$$

Here  $Kp'$  is the output of the defuzzifier against the current error. This procedure is summarized using a flow diagram in figure 5.5.

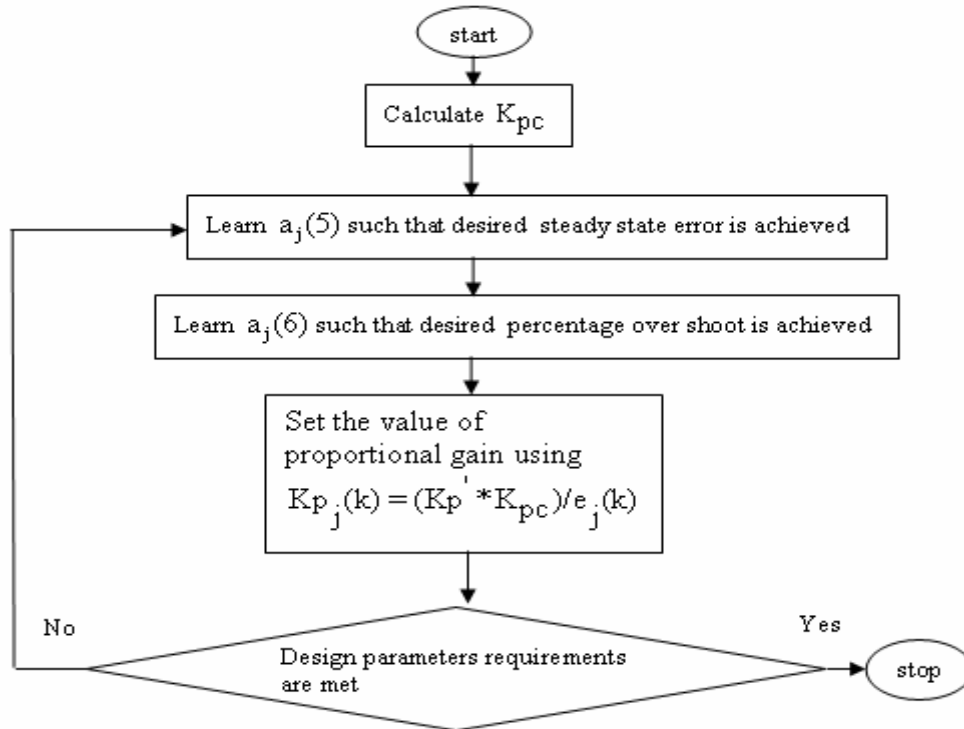


Figure 5.5: Flow chart of the procedure.

The flow chart recommends to learn  $K_{pc}$ ,  $a_j(5)$  and  $a_j(6)$  in order. The order of learning should be maintained even if a second iteration is required for the calculation of these values.

We now present an observer based approach to guarantee the stability of ILFGS.

### 5.2.2 Stability

The parameters of PID controller are function of time and the fuzzy gain scheduler is a non-linear system. Fuzzy mathematics and conventional control mathematics don't mix well. Hence, there is a need for other innovative techniques to guarantee stability and convergence. Recommendations are made for a hierarchical entity, like a supervisor to monitor the performance of the control system. Such multilevel controls structures turn out to be more useful in complex practical systems [103]. To detect instability, using these hierarchical structures, there are some methods



available. Anderson [82] suggests monitoring the magnitude of peaks, and the system is determined to be unstable when peaks are growing in magnitude, for three peaks in a row. A ratio of short term integral of error and the integrated absolute value of the error is used by [9] to get an indication of instability. An instability indicator is also proposed by [66] by observing the output, for the same purpose. We can also combine these and other instability detection proposals to achieve better instability detection [154]. Very fast rise time, high overshoot ratio and very long settling time can be good stability indicators also. Once instability has been detected the controller parameters are switched to a set of guaranteed stable parameters pre-stored in memory or the system is shut down by setting  $K_{p_j}(k)$ ,  $K_{i_j}(k)$  and  $K_{d_j}(k)$  to zero. This observer based stability implementation, is proposed as in figure 5.6.

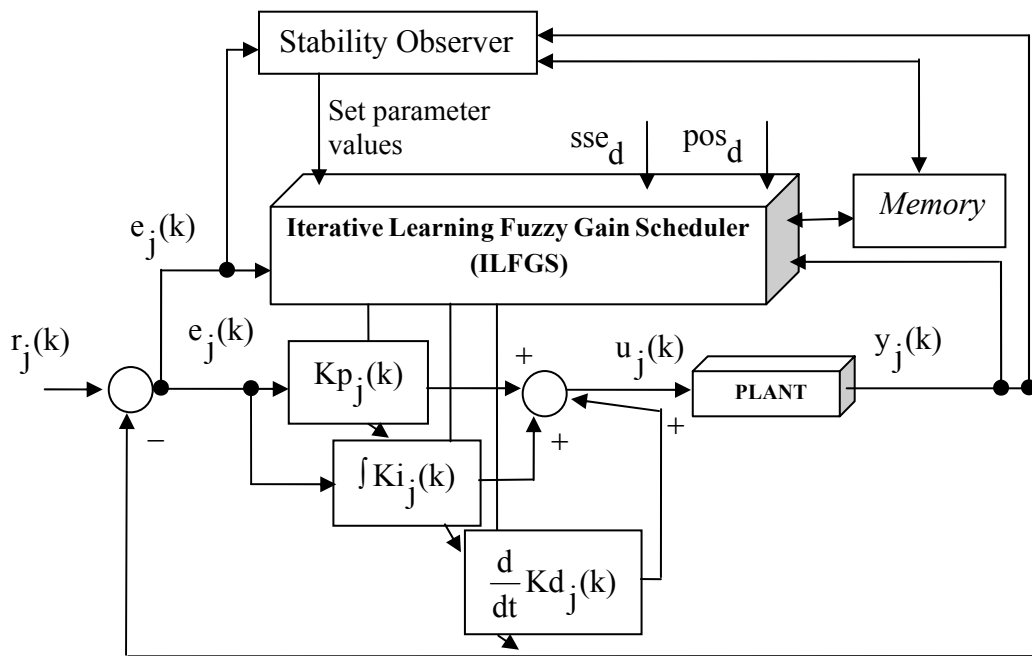


Figure 5.6: Proposed scheme with a stability observer.

The stability observer monitors the error and the current output of the system continuously. For this scheme the stability observer takes action only when three peaks of output are found with increasing values. Other stability observation criteria can also be utilized.

### 5.2.3 Simulations and results

The scheme presented in this chapter was tested through simulations. One of the simulations involved a model of DC motor. The aim is to design a speed controller which should endure less than 5% steady state error (sse) and less than 5% percentage over shoot (pos). For simulation purposes, all the ranges were normalized between -1 and 1.

#### 5.2.3.1 Motor Speed Control

The motor transfer function used is given in (4.14). As discussed in section 4.4.1, using conventional design approach alone, it is impossible to configure a controller for this system which would ensure less than 5% steady state error and less then 5% over shoot. Using the proposed scheme, explained above, the first step is to find the value of  $K_{pc}$ . The value of  $K_{pc}$  learnt was,  $K_{pc} = 11.68$ . Its learning history is presented in figure 5.7.

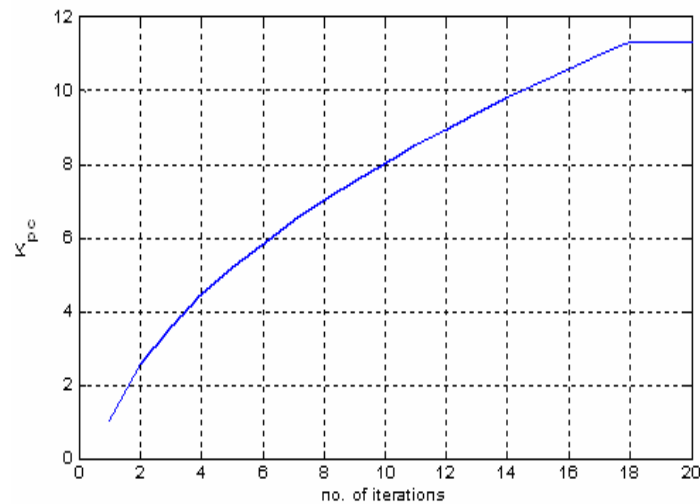


Figure 5.7: Learning value of  $K_{pc}$ .

Learnt value of  $a_j(5)$  for a steady state error of less than 5% was  $a_j(5) = 0.0846$ .

A plot of steady state error against number of iterations as  $a_j(5)$  was learnt is shown in figure 5.8 below.

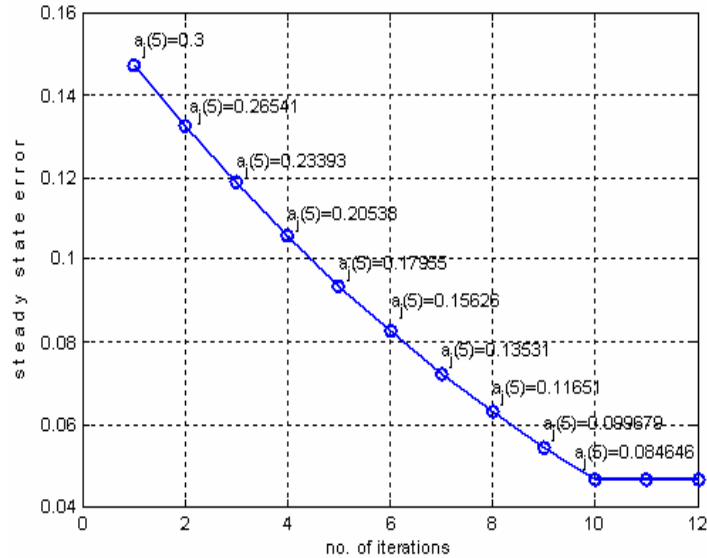


Figure 5.8: Decrease in steady state error as  $a_j(5)$  is learnt.

After first iteration, value of  $a_j(5)$  was learnt to be 0.3. At second iteration it was 0.265.

The value kept on decreasing until at 10<sup>th</sup> iteration it became 0.084. At this iteration the steady state error requirements were met.

The learnt value of  $a_j(6)$  was  $a_j(6) = 0.7626$ . After learning both endpoints, the fuzzy controller had a control surface, as exhibited in figure 5.9. The scheme was able to learn a non linear control surface, through piece-wise linear approximation.

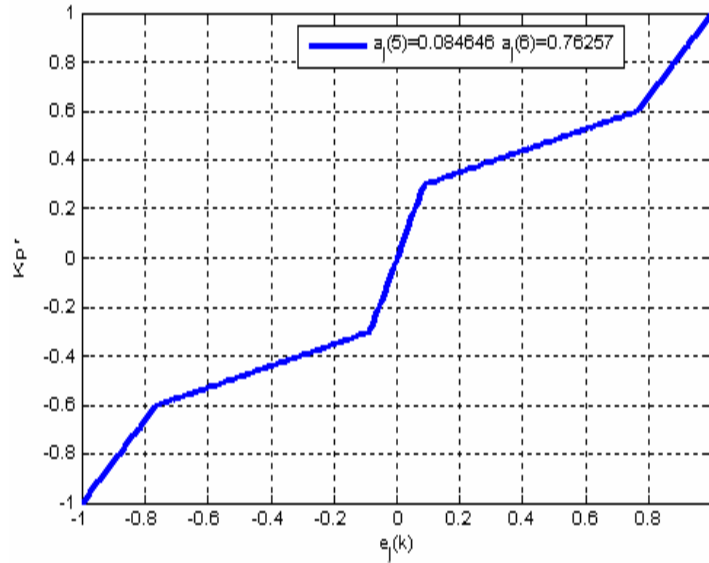


Figure 5.9: Learnt control surface.

The surface can be made smoother by considering more membership functions. The system's response for a desired speed of 1 rad/sec is presented in figure 5.10.

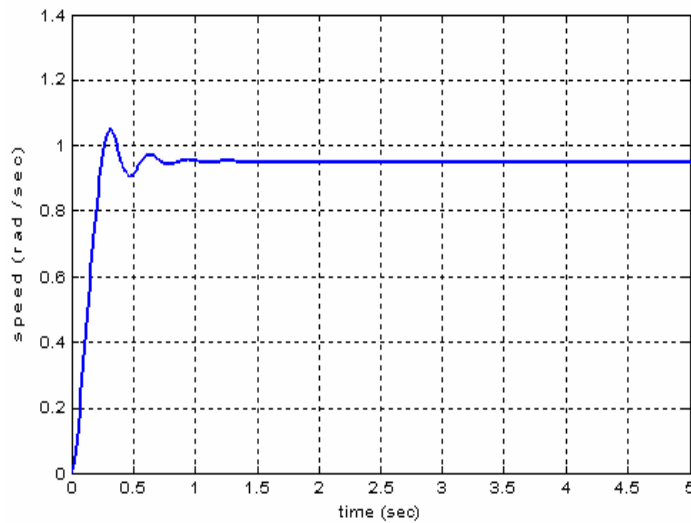


Figure 5.10: Motor speed against time.

The response shows that the system reaches 0.96 rad/sec within 1 second and the overshoot is under 5%. The behaviour of error is shown in figure 5.11.

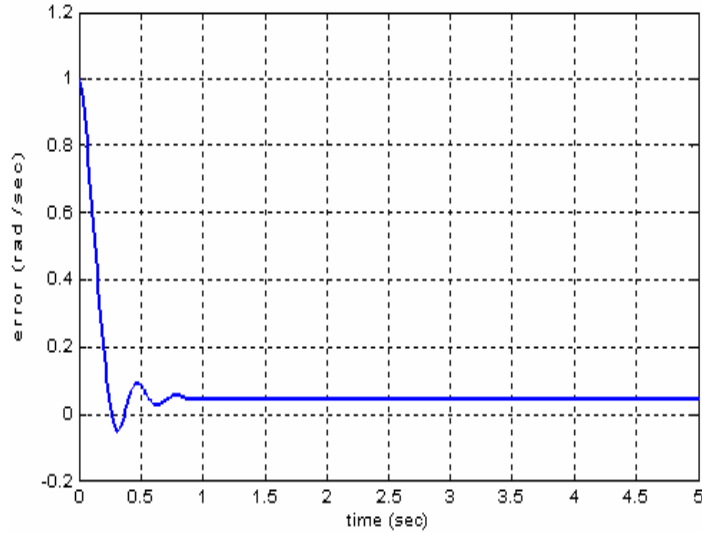


Figure 5.11: Error between the desired speed and the actual speed.

The ILFGS calculates the value of  $K_{p_j}(k)$  at each sample. This variation in  $K_{p_j}(k)$  help us achieve the desired steady state error and percentage overshoot. The values of  $K_{p_j}(k)$  calculated by the proposed scheme for a desired speed of 1 rad /sec with an over shoot of no more than 0.05 rad /sec, are given in figure 5.12.

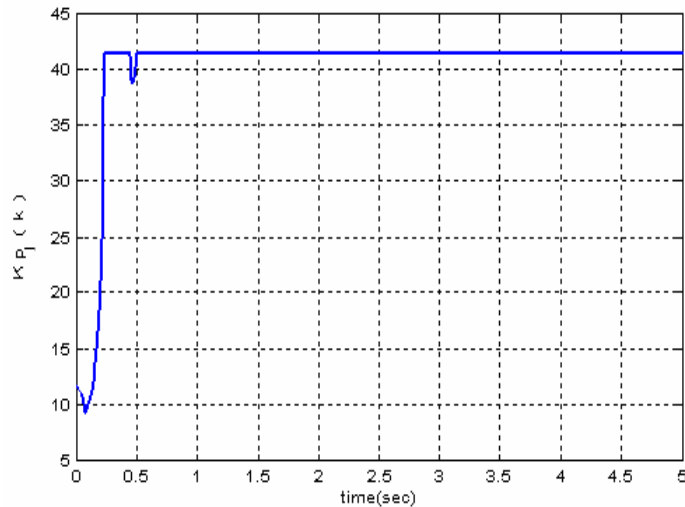


Figure 5.12: The values of  $K_{p_j}(k)$  calculated by the proposed system to achieve the desired performance.

The plot shows that quite a complicated variation of  $K_{p_j}(k)$  was calculated by the scheme to ensure design requirements. Once the required speed is achieved, there is no change in  $K_{p_j}(k)$ .

### 5.2.3.2 Zeigler-Nichols controller vs. proposed approach

PID controllers are the most widely used controllers in the industry. PID controllers can be implemented to meet various design specifications such as steady state error, percentage overshoot and rise time. Despite their wide use, tuning a PID controller can be a very tedious job. Most of the conventional tuning methods require at least some knowledge of the system we want to control. One approach, the Zeigler-Nichols tuning method, which was developed in the 1950's and has stood the test of time is still the most used tuning method. The procedure adapted in this chapter for tuning the controller using Zeigler-Nichols approach uses the table 5.4.

Control	$K_p$	$K_i$	$K_d$
<b>P</b>	$0.5 K_c$		
<b>PI</b>	$0.45 K_c$	$0.833 T_c$	
<b>PID</b>	$0.6 K_c$	$0.5 T_c$	$0.125 T_c$

Table 5.4: Zeigler-Nichols gain calculation table.

Here  $K_c$  is the critical gain and  $T_c$  is the time constant of the system.

This section presents the results obtained by using Zeigler-Nichols method. These results are then compared with the results from our proposed approach. For this comparison three different systems were selected. Their transfer functions are presented below.

1- A cruise control system given by the transfer function

$$G_1(s) = \frac{1}{10s + 50} \quad (5.9)$$

2- A road vehicle model given by the transfer function

$$G_2(s) = \frac{40}{2s^3 + 10s^2 + 82s + 10} \quad (5.10)$$

3- A motor speed control system given by the transfer function

$$G_3(s) = \frac{0.05}{0.005s^2 + 0.06s + 0.1025} \quad (5.11)$$

The parameters determined for the Zeigler-Nichols based P, PI and PID controllers, to control the selected three systems, are presented in table 5.5.

System	Zeigler-Nichols P controller	Zeigler-Nichols PI controller	Zeigler-Nichols PID controller
$G_1(s)$	$K_p = 10000$	$K_p = 9000$ $K_i = 0.1666$	$K_p = 12000$ $K_i = 0.1$ $K_d = 0.025$
$G_2(s)$	$K_p = 4.2$	$K_p = 3.784$ $K_i = 0.9163$	$K_p = 5.076$ $K_i = 0.55$ $K_d = 0.1375$
$G_3(s)$	$K_p = 122.5$	$K_p = 110.25$ $K_i = 0.1083$	$K_p = 147$ $K_i = 0.065$ $K_d = 0.0163$

Table 5.5: Parameters calculated for Zeigler-Nichols based P, PI and PID controllers.

The ILFGS was also made to learn the required parameters for the selected systems. The parameters learnt, are shown in table 5.6.

System	$K_{pc}$	$a_j(5)$	$a_j(6)$
$G_1(s)$	1.0015	0.00003	0.00006
$G_2(s)$	3.7836	0.2050	0.7023
$G_3(s)$	11.68	0.0846	0.7626

Table 5.6: Parameters calculated by ILFGS.

Comparison of the response for a cruise control system, between the 3 Ziegler-Nichols based controllers and the controller proposed in this chapter is shown in figure 5.13.

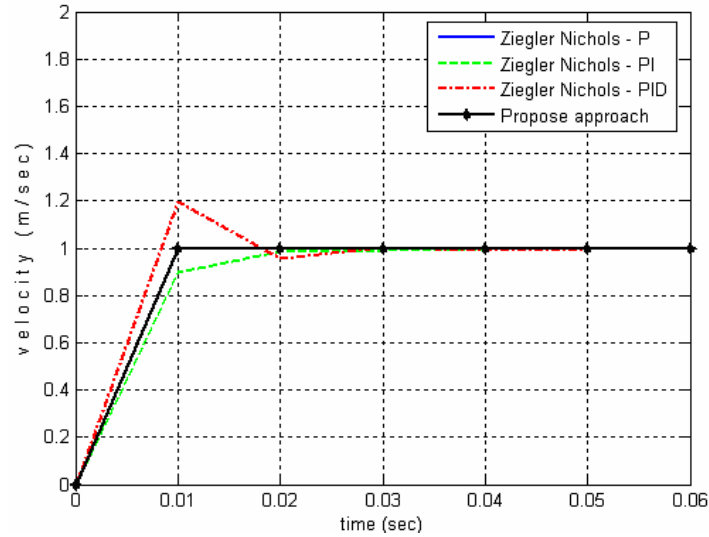


Figure 5.13: Comparison of step response for cruise control system.

This was the easiest of the three systems to control. Figure 5.13 show that the P controller and the proposed ILFGS based system, give best performances. With respect to steady state error all the controllers performed well.

For the road vehicle system the response with different controllers is exhibited in figure 5.14.

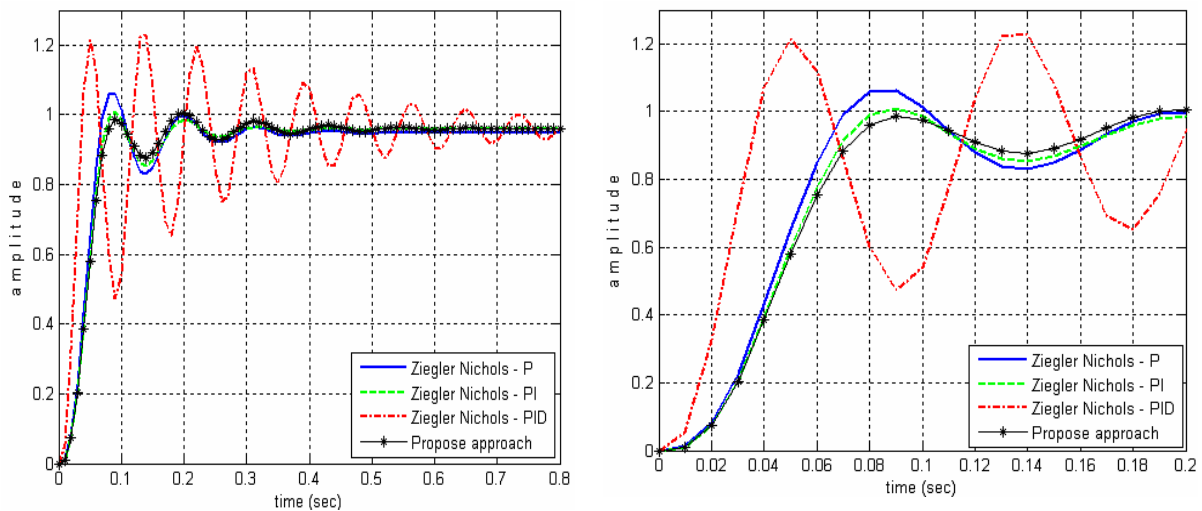


Figure 5.14: Comparison of step response of road vehicle system.



Figure on the left show that ILFGS based system performed better then other controllers. The figure on the right shows a zoomed version of the response to get a better view of the behaviour of the controllers. For further insight, comparison of the error between the PID controller and the ILFGS based controller is presented in figure 5.15.

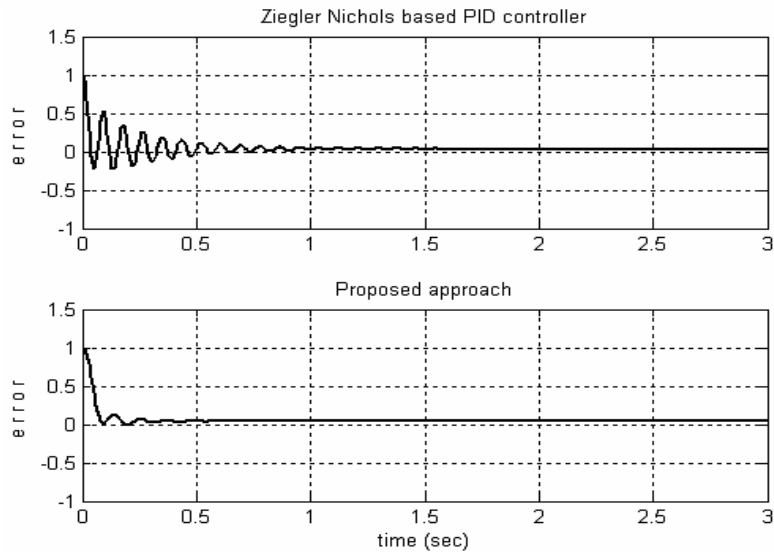


Figure 5.15: Comparison of error between PID and proposed controller.

The response given by the motor speed control system, using different controllers is shown in figure 5.16.

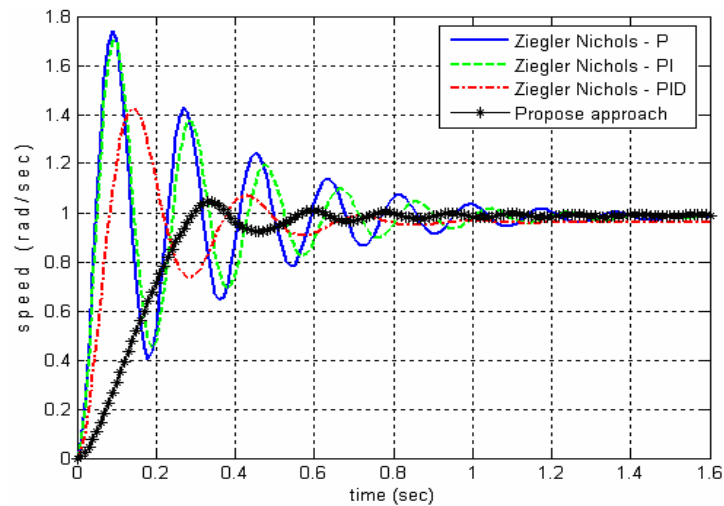


Figure 5.16: Comparison of step response for a motor speed control system.

The comparison of error for the above system is exhibited in figure 5.17.

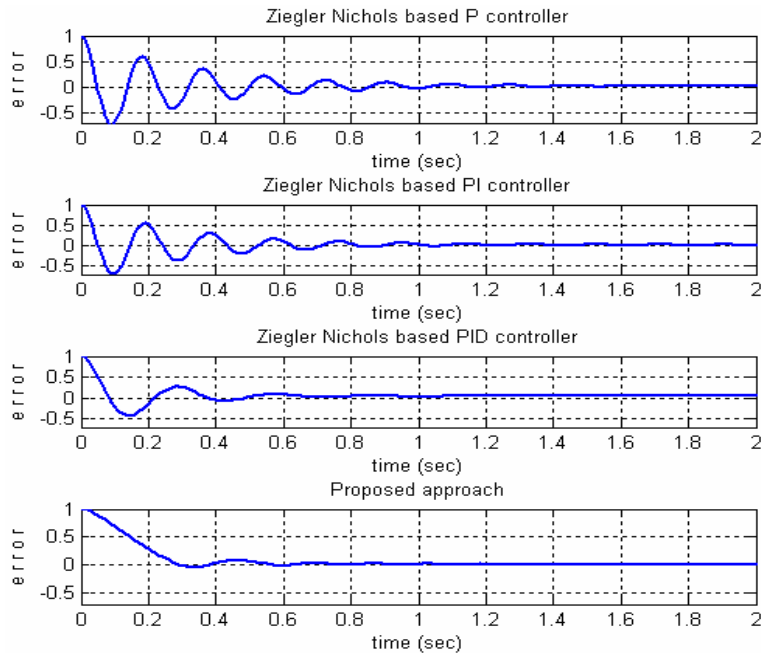


Figure 5.17: Comparison of error for motor speed control system.

For this system, as far as steady state error is concerned, PID and ILFGS based controllers performed equally well. For percentage overshoot requirements ILFGS based controller met the requirements better than any other controller.

Simulation results, presented above show that a variety of systems can be satisfactorily controlled by the ILFGS based controller. It yielded better control performance than the Ziegler-Nichols based P, PI and PID controllers. This is true even when only proportional gain scheduling was done for the proposed scheme. The control surfaces generated by the ILFGS based controller are plotted below to have a feel of the learning done by the controller. For the cruise control system the control surface is shown in figure 5.18.

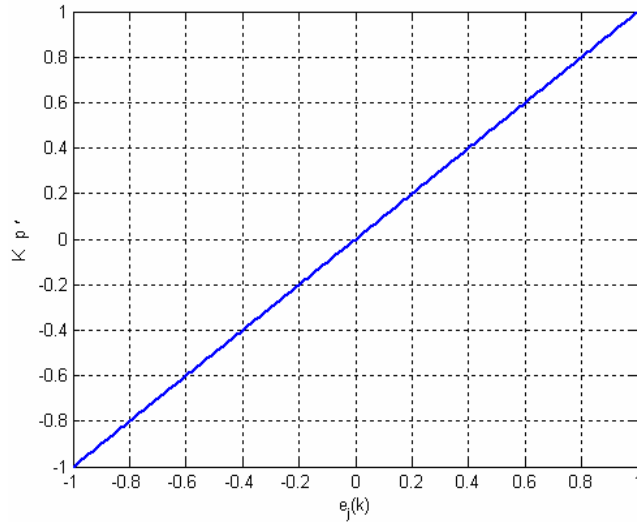


Figure 5.18: Learnt control surface for cruise control system.

The control surface is linear and similar to the control surface generated by the P controller. For the road vehicle system the control surface generated is plotted in figure 5.19.

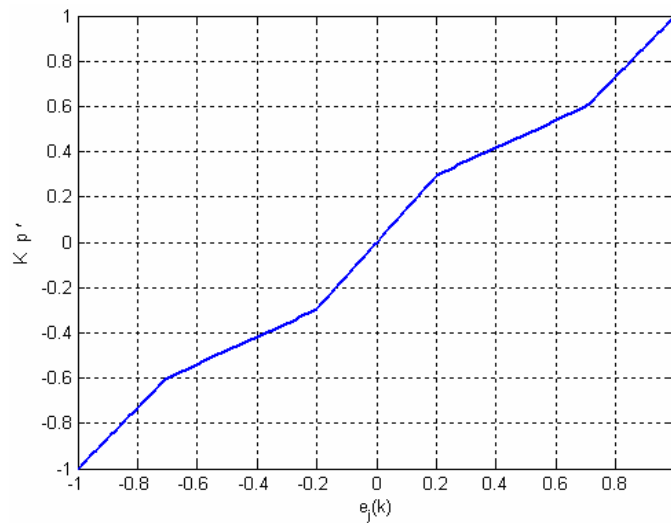


Figure 5.19: Learnt control surface for road vehicle system.

The control surface is slightly non-linear. The motor speed control system led to the generation of control surface presented in figure 5.20.

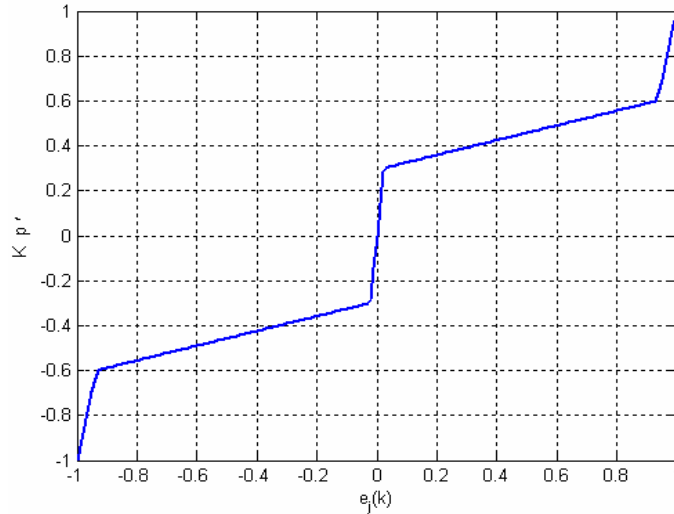


Figure 5.20: Learnt control surface for motor speed control system.

The control surface is non-linear. The abrupt change in slope, after regular intervals, can be smoothed by considering more membership functions.

### 5.2.3.3 Tracking trajectories in real time

Tracking is an important and difficult practical problem, especially when the target is changing its position. To demonstrate the effectiveness of this approach, different trajectories were tracked. One of them was a sinusoidal signal. This tracking was done after the learning of the parameters had been achieved, as discussed before. For this demonstration, road vehicle system and motor speed control system of equation (5.10) and (5.11) were used and was given a task to follow a sinusoidal amplitude and speed requirements. The speed requirements meant that the system had to move in one direction with a maximum speed of 1 rad/sec and then in the opposite direction achieving speeds of 1 rad/sec again.

The plots in figure 5.21 and 5.22 show the performance of road vehicle system and motor speed control system, while tracking a sinusoidal trajectory, using ILFGS.

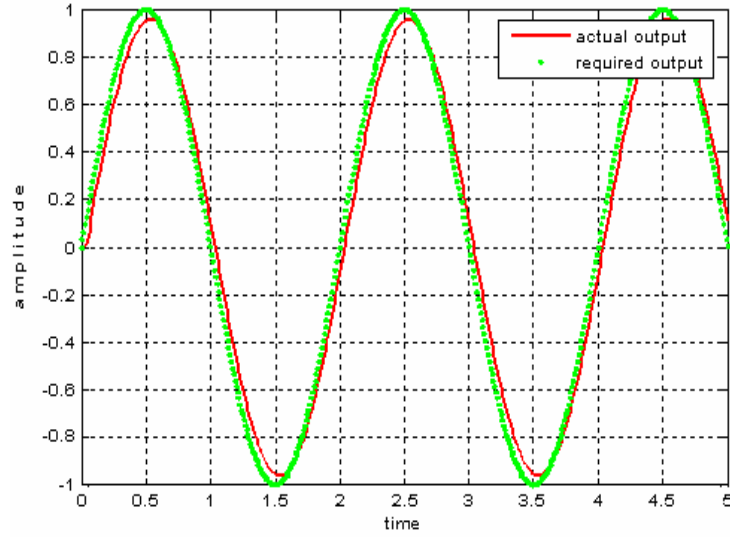


Figure 5.21: A road vehicle system tracking a sinusoidal signal.

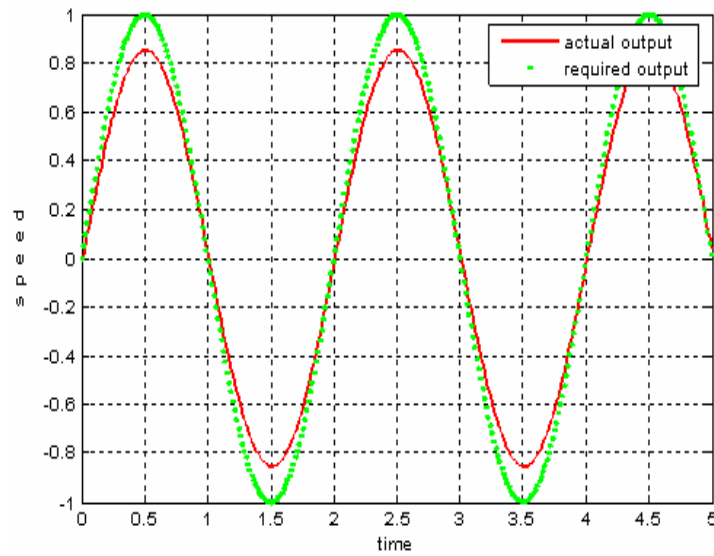


Figure 5.22: A motor speed control system tracking a sinusoidal signal.

Figures show that the system was quite effectively able to track the desired response. The plots of error vs. time for these performances are shown in figure 5.23 and 5.24.

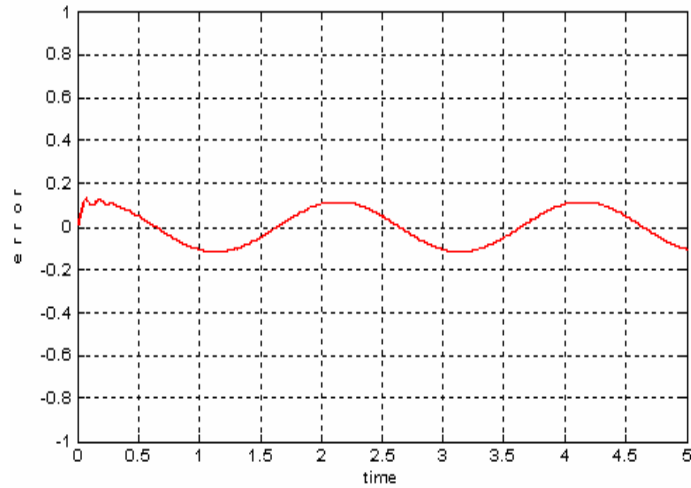


Figure 5.23: Error generated by road vehicle system.

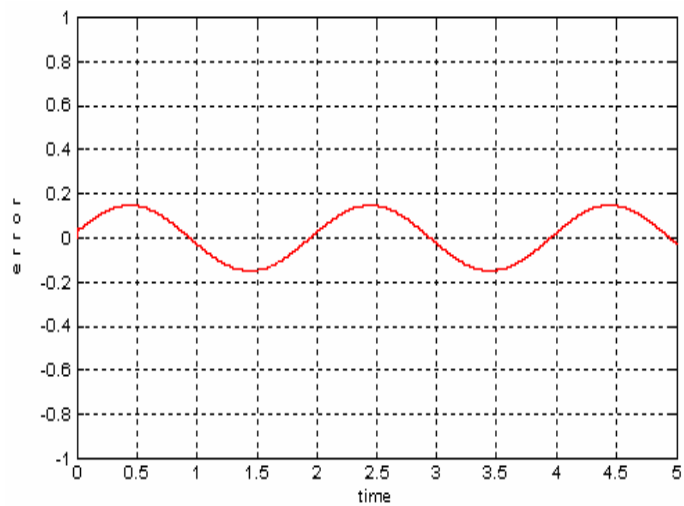


Figure 5.24: Error generated by motor control system.

The error looks larger than 5% but this is due to the fact that the response was delayed. These error plots are taken without taking the delay in consideration.

Other waveforms were also tracked successfully by the ILFGS. One interesting plot showing a vehicle moving at constant speed and then gradually decreasing its speed to zero is shown in figure 5.25.

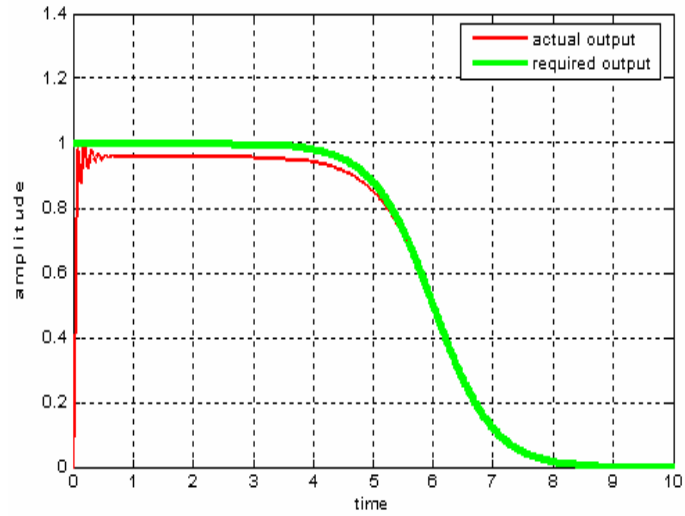


Figure 5.25: Road vehicle system following the desired speed curve.

The vehicle system is successfully able to follow the required waveform. The corresponding error plot is presented in figure 5.26.

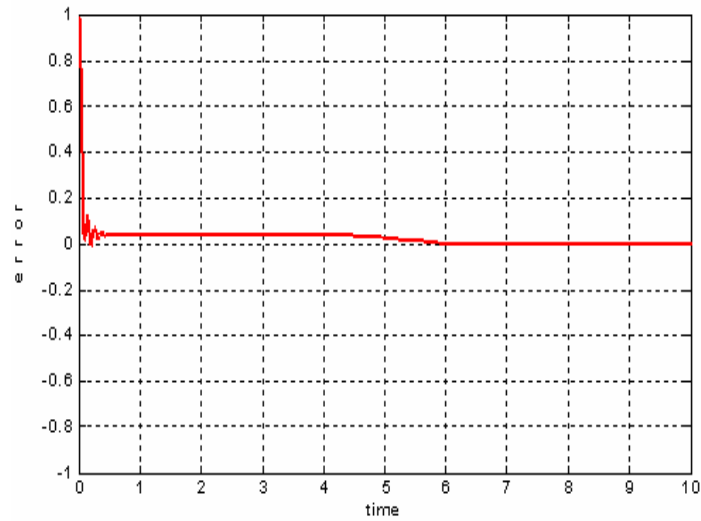


Figure 5.26: Behaviour of error.

The error decreases very sharply and eventually goes to within the desired limit. The desired limit in this case was 1% of the desired amplitude.

### 5.3 Experimental Setup and Results

A QET DC Motor Kit from Quanser was used to test the approach with a practical setup. The kit can be programmed to set proportional, integral and derivative gains. MATLAB was used to communicate with the kit through serial port. ILFGS was implemented in MATLAB. The micro controller on the kit was used to set proportional gain and return speed of the motor. The experimental setup is show in figure 5.27 below. The aim is to make the motor run at 100 rad/sec with a steady state error of less than 5% and an over shoot of less than 5%.



Figure 5. 27: Experimental setup with the QET DC Motor Kit from Quanser.

The different step size parameters, in equations (5.2), (5.4) and (5.6), were given values of  $\mu_{kp} = 0.01$ ,  $\mu_{sse} = 0.01$  and  $\mu_{pos} = 0.01$ . The error had a range of  $e_j(k) = [-200, 200]$  and gain had a range of  $K_p = [-1, 1]$ . During the learning phase, the values of  $K_{pc}$ ,  $a_j(5)$  and  $a_j(6)$  learnt were  $K_{pc} = 0.25$ ,  $a_j(5) = 15.8$  and  $a_j(6) = 76.36$ . A plot of learning history of parameters  $a_j(5)$  and  $a_j(6)$  vs. no. of iterations is indicated in figure 5.28 below.



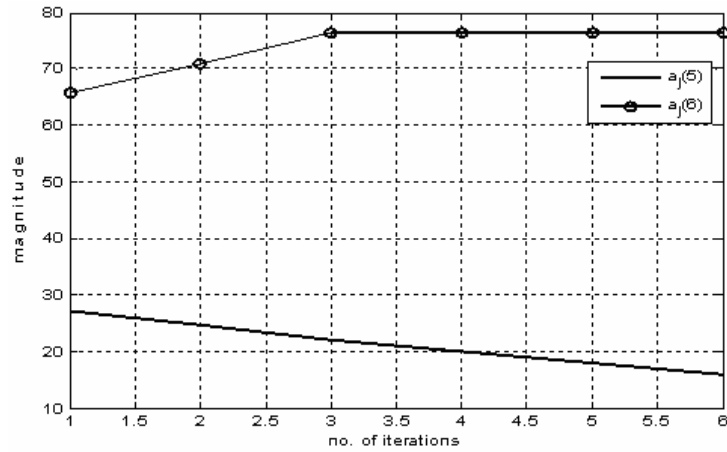


Figure 5.28: Learning values of  $a_j(5)$  and  $a_j(6)$ .

The plot shows that  $a_j(5)$  was learnt in 6 iterations and  $a_j(6)$  was learnt in 4 iterations. A 3-D plot of speed vs. time vs. number of iterations, while learning  $a_j(6)$ , during the process of eliminating percentage over shoot, is given in figure 5.29.

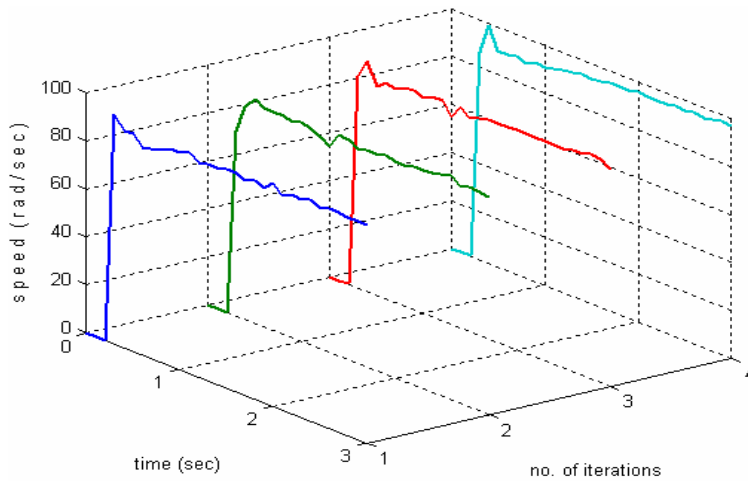


Figure 5.29: History of system output while learning  $a_j(6)$ .

The plot shows reduction in percentage over (or under) shoot as  $a_j(6)$  is learnt iteratively. The membership functions learnt after the learning process was completed are given in figure 5.30.

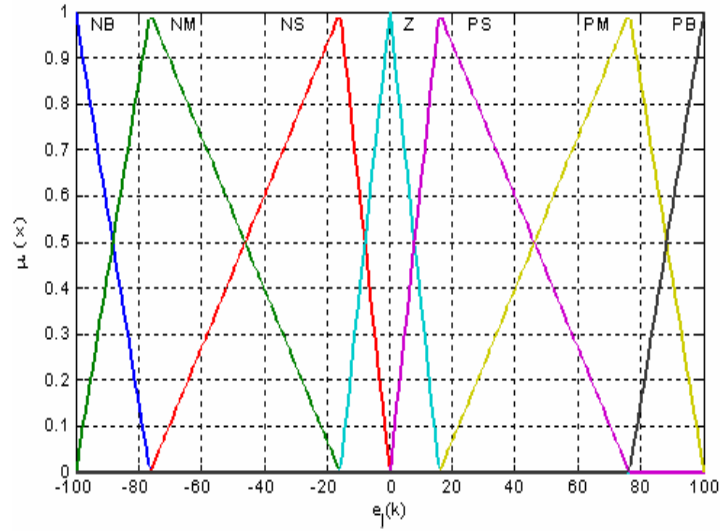


Figure 5.30: Learnt membership functions.

Once the learning was complete, the DC motor was made to run at a speed of 100 radians /sec. The motor's response is presented in figure 5.31.

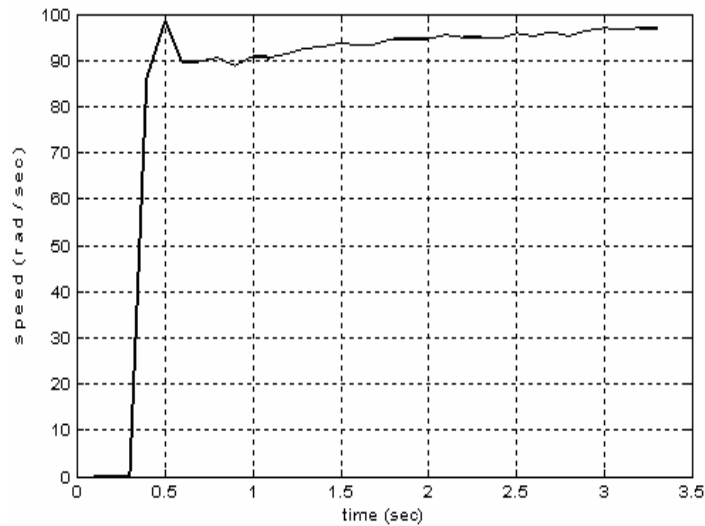


Figure 5.31: Response using the proposed scheme.

As can be seen from the figure the motor hits a speed of 96.1 rad /sec in 3 sec. The overshoot (or undershoot) is also within limits. To give a better perspective of the response, a plot of error is exhibited in figure 5.32.

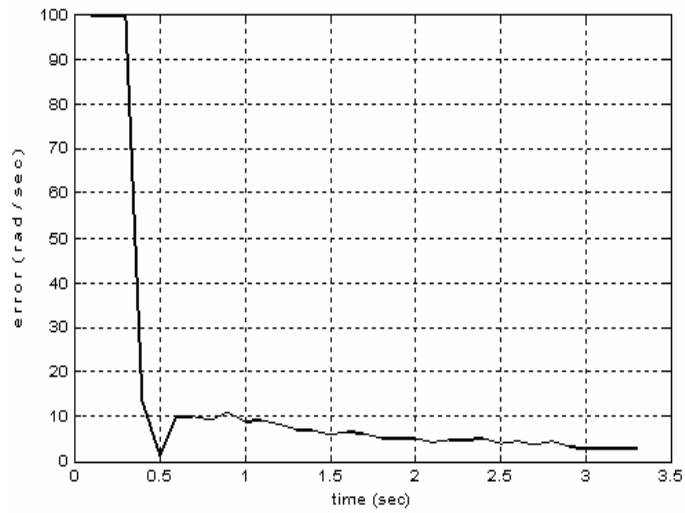


Figure 5.32: Behaviour of error.

The values of proportional gain calculated by ILFGS during this run are shown in figure 5.33.

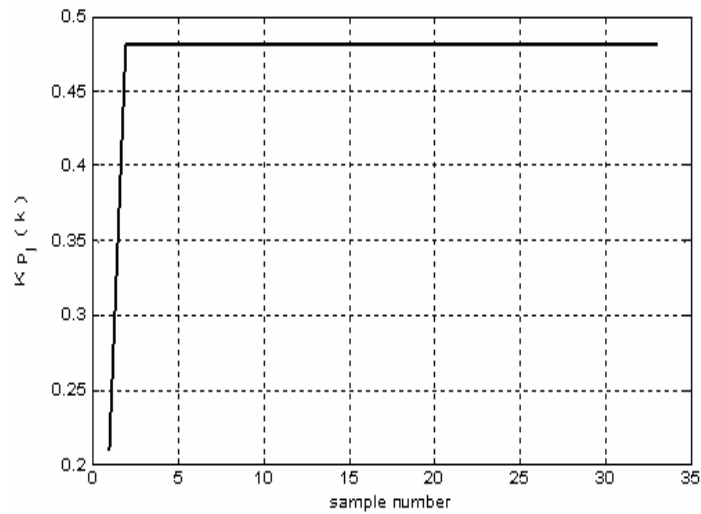


Figure 5.33: Calculated proportional gains by the ILFGS to control the QET DC Motor.

The figure shows that there are large variations in the values of gain initially, when the error is large. It then settles to a steady value of 0.481 as the required speed is achieved.

### 5.3.1 Stability using Linguistic Trajectory

Stability is one of the most important subjects for a controlled system. For fuzzy based systems it is a very complex topic and to date no general stability analysis methodology has been presented. Because of the complexity, there are even different definitions of stability. One definition is that if bounded input is applied, then there should be bounded output. This is called BIBO stability. Fuzzy logic because of its structure defines input and output boundaries for membership function values. Another definition is that when all the input and interferences have disappeared, the system should come to rest to its original state.

By probing into the relation between the relative influence of each rule within the rule base and the linguistic trajectory of the dynamic system, it can be determined whether the fuzzy system has reached stability. For this experimental setup we had one input and one output linguistic variable. It is proposed to plot all the rules fired in an extended rule base table to get the linguistic trajectory.

There were seven rules defined for the rule base. The rule base is shown in table 5.7.

$e_j(k)$	NB	NM	NS	Z	PS	PM	PB
$K_p'$	NB	NM	NS	Z	PS	PM	PB

Table 5.7: Rule base for the experimental setup.

If the linguistic trajectory converges to the equilibrium point the system is stable. The linguistic trajectory is shown in table 5.8.

$e_j(k)$	NB	NM	NS	Z	PS	PM	PB
$K_p'$	NB	NM	NS	Z	PS	<b>PM</b>	<b>PB</b>
	NB	NM	NS	<b>Z</b>	<b>PS</b>	PM	PB
	NB	NM	NS	<b>Z</b>	<b>PS</b>	PM	PB
				$\vdots$	$\vdots$		
	NB	NM	NS	<b>Z</b>	PS	PM	PB
	NB	NM	NS	<b>Z</b>	PS	PM	PB

Table 5.8: Linguistic trajectory.

The trajectory in table 5.8 shows a graphical view of the rules fired during system operation. The first row of the table shows the input membership functions while the second row shows the output membership functions. The subsequent rows also represent the output membership functions to show the firing of rules. As can be seen, the firing of rules converges to the desired rule, i.e.:-

$$\text{If } e_j(k) \text{ is } Z \text{ then } K_p' \text{ is } Z \quad (5.12)$$

Hence the system is stable and the error is decreasing continuously.

## 5.4 Summary

Fuzzy controllers and for that matter any controller needs to be adaptive in order to compensate for uncertainties, noise, variation in parameters and changes in design requirements. Conventional P, PI, PID etc. controllers need to be adaptive in order to be more useful, especially when the plant parameters are not known. It is now realized that complex real world problems require intelligent systems that combine knowledge, techniques and methodologies from various sources. These intelligent systems are supposed to possess humanlike expertise within a specific domain, adapt themselves and learn to do better in changing environment [75]. The approach presented in this chapter combines conventional controls with fuzzy logic and iterative learning to tackle these real world problems.

The scheme changes the proportional, integral and derivative gains adaptively through the Iterative Learning Fuzzy Gain Scheduler (ILFGS). The fuzzy controller in ILFGS itself has the capability to adapt because of iterative learning. The learning laws ensure that steady state error, percentage overshoot etc. requirements are met. The ILFGS adjusts the proportional gain in real time to meet design requirements.

As seen from simulation results and results from a practical Quanser DC motor based setup, the proposed scheme was able to learn non-linear control surfaces even when only the proportional gain was scheduled. Integral and derivative gains can also be scheduled if the requirements are not met. Derivative gain should be introduced if the

percentage overshoot requirements are not met and integral gain should be used if steady state error requirements are not met. Stability of the ILFGS is discussed using a supervisory level based approach and linguistic trajectories.

It is difficult to determine whether humans are fuzzy based learning machines or learning based fuzzy machines. Chapter 4 and 5 have been focusing on Learning based Fuzzy controllers.

In the next chapter we develop a Fuzzy based Learning controller.

## 6 FUZZY ITERATIVE LEARNING CONTROLLER (FILC)

As fuzzy logic tries to mimic one aspect of human behaviour i.e. perception based thinking; iterative learning follows the other aspect i.e. learning through experience. So far, the research has mainly focused on iterative learning and iterative learning helping the fuzzy system. The research results presented in this chapter are aimed at developing a mechanism to use the power of fuzzy to adjust iterative learning controller parameters. In the previous two chapters, fuzzy controller was the main controller and Iterative Learning was used as a secondary controller. Here ILC is the main controller and fuzzy acts as a helping unit. The ILC can be any controller developed in chapter 2 and 3.

### 6.1 Proposed Approach

The block diagram of the proposed approach, named Fuzzy Iterative Learning Controller (FILC) is described in figure 6.1.

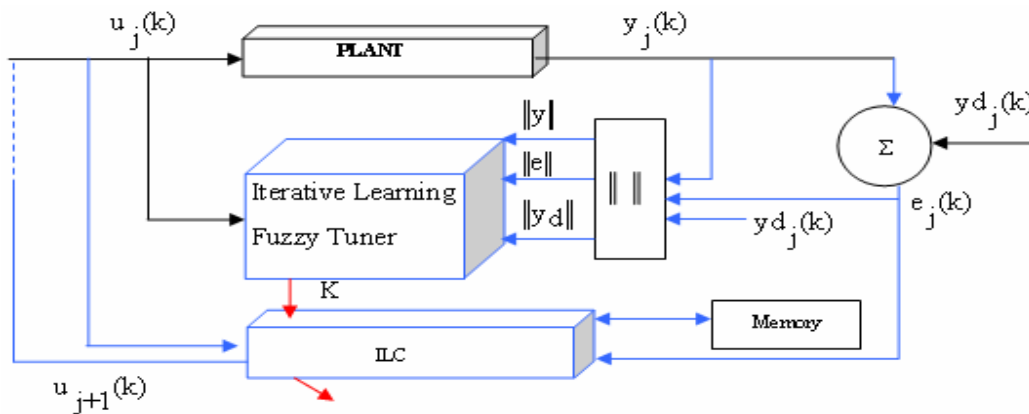


Figure 6.1: Proposed Fuzzy Iterative Learning Controller.

Input  $u_j(k)$  is applied to the plant which results in an output labelled,  $y_j(k)$ . This output and the desired output  $y_{d_j}(k)$ , results in an error  $e_j(k)$ . The system output, desired

output and the error go into an Iterative Learning Fuzzy Tuner (ILFT) block. The output of the block is the value of gain,  $K$ , that is used by the ILC to calculate the next input to the plant,  $u_{j+1}(k)$ . Other gain parameter values, like  $K_1$  from equation (2.4), can also be calculated by the ILFT block using similar procedure. Memory is used to store different parameters, to be used in subsequent iterations. The ILFT block is further explained in figure 6.2.

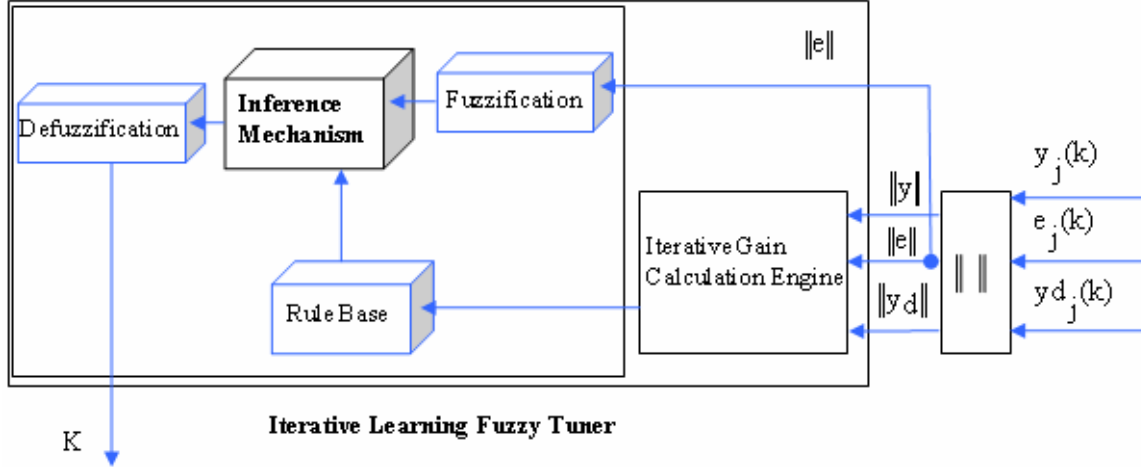


Figure 6.2: ILFT block in more detail.

Error is the input for the fuzzy system and a value of  $K$  is the output. Three inputs, norm of error, norm of the output and norm of desired output, enter the Iterative Gain Calculation Engine. This engine adjusts the values of  $K_l$ ,  $K_m$  and  $K_h$  iteratively. Where  $K_l$ ,  $K_m$  and  $K_h$  are the Low, Medium and High values of gain for the output fuzzy sets. Their values are updated using the laws given below.

$$K_{l,j+1} = K_{l,j} + \mu_1 \text{abs}(e) \alpha \quad (6.1)$$

$$K_{m,j+1} = K_{m,j} + \mu_2 \text{abs}(e) \alpha \quad (6.2)$$

$$K_{h,j+1} = K_{h,j} + \mu_3 \text{abs}(e) \alpha \quad (6.3)$$

Here  $\alpha = \text{sign}(\|y_d\| - \|y\|)$  and



$$\alpha = \begin{cases} 1 & \text{if } \|y_d\| \geq \|y\| \\ -1 & \text{if } \|y_d\| < \|y\| \end{cases}$$

This  $\alpha$  determines whether the next value of  $K_I$ ,  $K_M$  and  $K_H$  will decrease or increase. The value of gain, calculated by the Defuzzifier is used in ILC controller to find the next input to the plant,  $u_{j+1}(k)$ . This input should bring  $y_j(k)$  and  $y_d(k)$  closer together.

### 6.1.1 Simulation Results

Simulation results from car suspension system, presented in Appendix A, are described in this section. Input membership functions, defined for this system, are explained in figure 6.3.

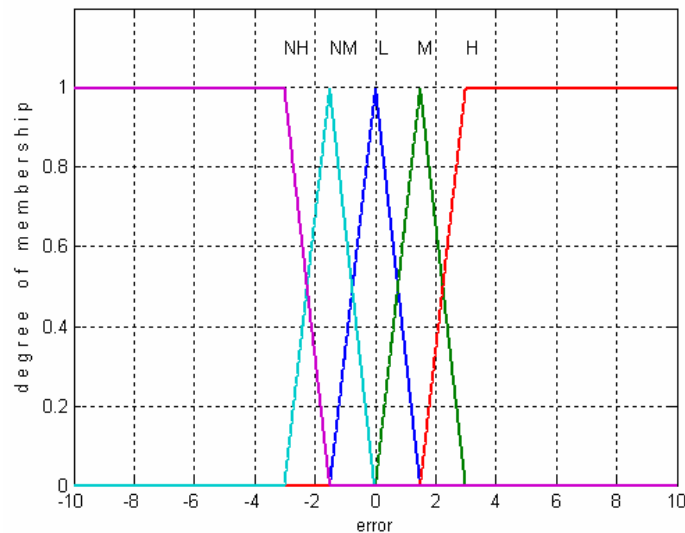


Figure 6.3: Input membership function for car suspension system.

For ease of implementation, triangular membership functions were used. Here L stands for ‘Low’, M for ‘Medium’, H for ‘High’, NM for ‘Negative Medium’ and NH for ‘Negative High’ values of error. The universe of discourse for error is  $[-10,10]$ . The proposed end point values of the membership functions are tabulated in table 6.1.

$e_j(k)$	NH	NM	L	M	H
End point values	-10, -1.5	-3, 0	-1.5, 1.5	0, 3	1.5, 10

Table 6.1: Input membership function end points.

Two sets of results are presented. One, with no initial guess for the values of  $K_l$ ,  $K_m$  and  $K_h$  and the other with some initial guess of these values. The Iterative learning controller used for both cases was MSATILC, from chapter 2.

### 6.1.1.1 Case 1: With no initial guess

For this case the starting values of the parameters in equation (6.1), (6.2) and (6.3) were,  $K_l = 0.1$ ,  $K_m = 0.1$ ,  $K_h = 0.1$ ,  $\mu_l = 0.1$ ,  $\mu_m = 0.1$  and  $\mu_h = 0.1$ . The Universe of Discourse for the output membership functions was taken as  $[-30, 30]$ . The output membership functions  $K_l$ ,  $K_m$  and  $K_h$  are shown in figure 6.4.

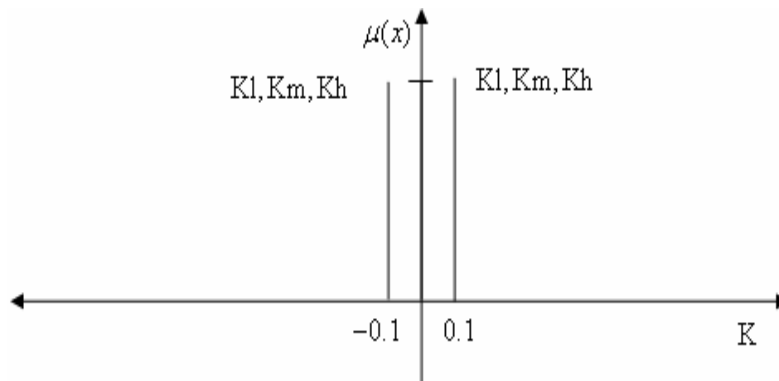


Figure 6.4: Output membership function before learning.

The ILFT block will adjust these values iteratively. With these settings, the control surface is shown in figure 6.5.

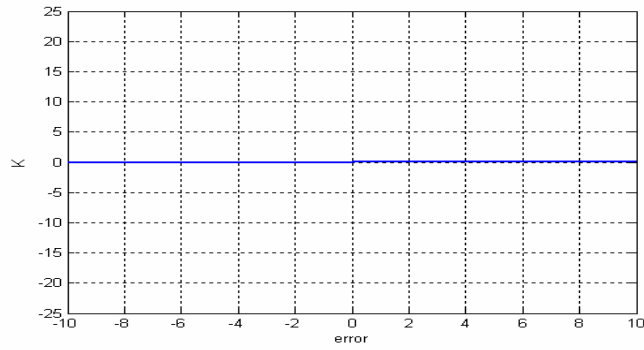


Figure 6.5: Control surface before learning.

The control surface reveals that the controller is not capable of doing any useful work, at this time. The proposed FILC should learn the appropriate control surface.

The rule base consisted of five rules. These are shown in table 6.2.

$e_j(k)$	NH	NM	L	M	H
$u$	Kh	Km	Kl	Km	Kh

Table 6.2: Proposed rule base.

During the process of learning, Kl, Km and Kh, the behaviour of error obtained is plotted in figure 6.6.

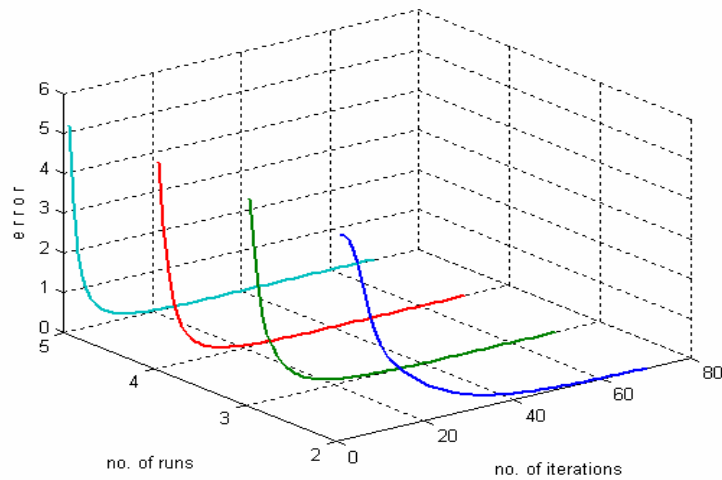


Figure 6.6: Improvement in performance as iterations increase.

There is a consistent decrease in error as the approach is used more. Two processes are working simultaneously to reduce number of iterations. One uses fuzzy logic to provide values of gain to the ILC controller. The other uses the iterative learning laws to find the appropriate input. Figure 6.6 shows that the system converges at first run but the convergence rate is slow. The number of iterations decreased with number of runs as exhibited in figure 6.7.

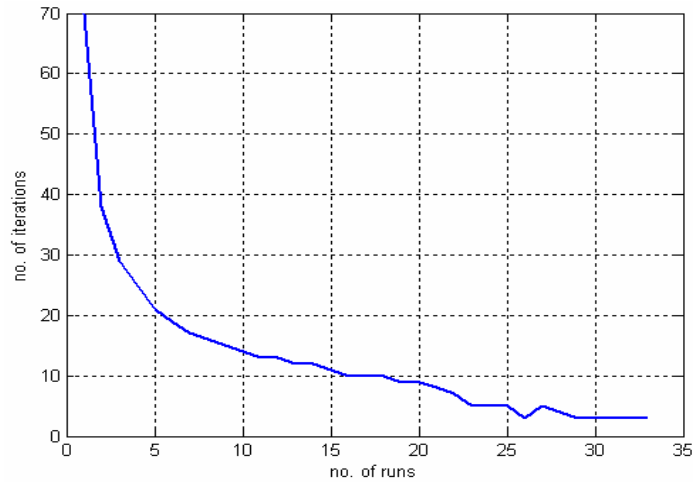


Figure 6.7: Decrease in number of iterations as task is repeated.

It is to be noted that after 30 runs the number of iterations is reduced to 3. The behaviour of error at 30<sup>th</sup> run is plotted in figure 6.8.

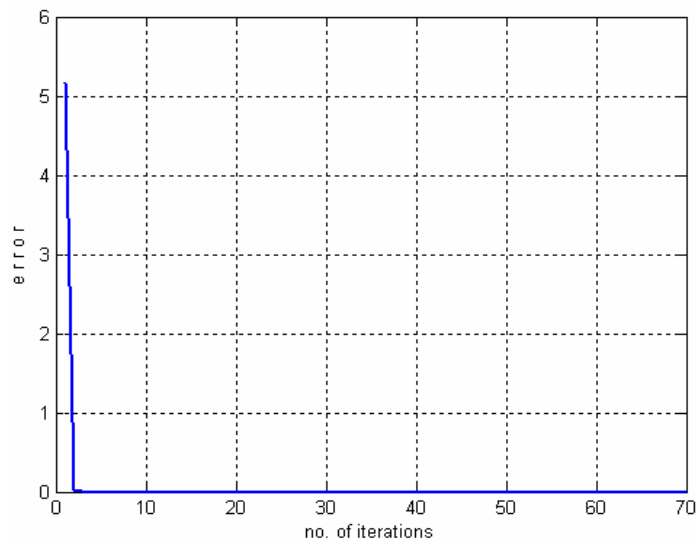


Figure 6.8: Behaviour of error at 30<sup>th</sup> run, with zero initial input.

Figure 6.8 shows a sharp decrease in error. The learning of  $K_I$ ,  $K_M$  and  $K_h$  is presented in figure 6.9.

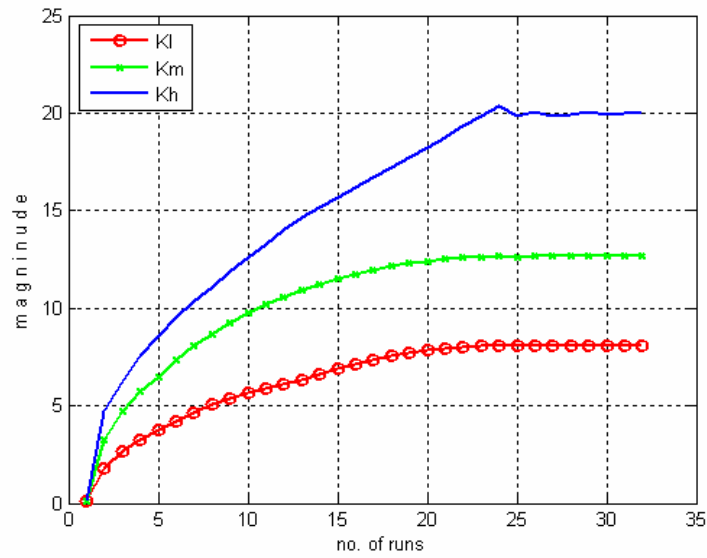


Figure 6.9: Learning behaviour of  $K_I$ ,  $K_M$  and  $K_h$ .

The learning settles at about 30<sup>th</sup> run. The control surface at this run is plotted in figure 6.10.

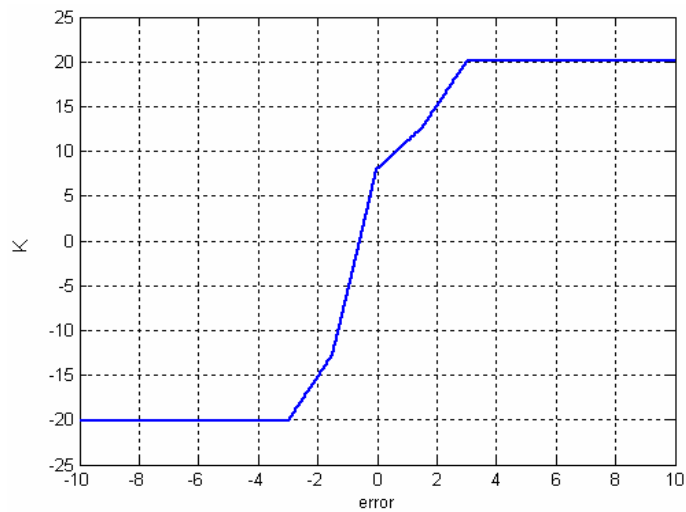


Figure 6.10: Control surface after 30 runs.

The output membership functions learnt, during this simulation, are shown in figure 6.11.

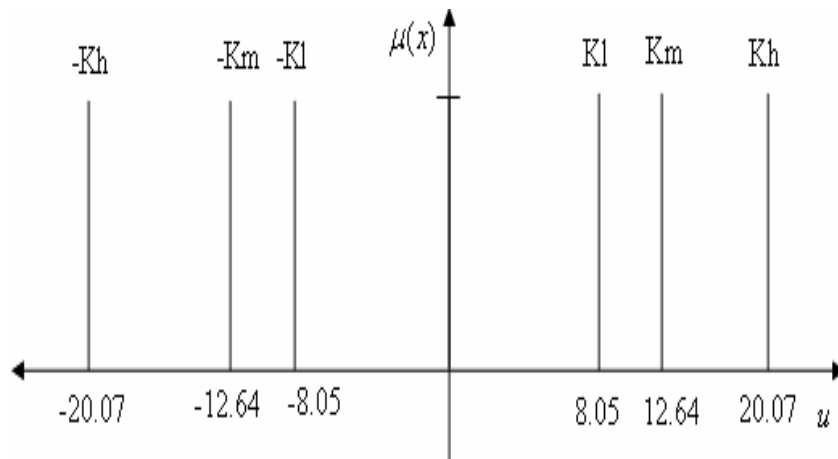


Figure 6.11: Learnt output membership functions after 30 runs.

At 30<sup>th</sup> run, the behaviour of the system is shown in figure 6.12. The desired response is shown in dotted thick lines and the actual output is in thin solid lines.

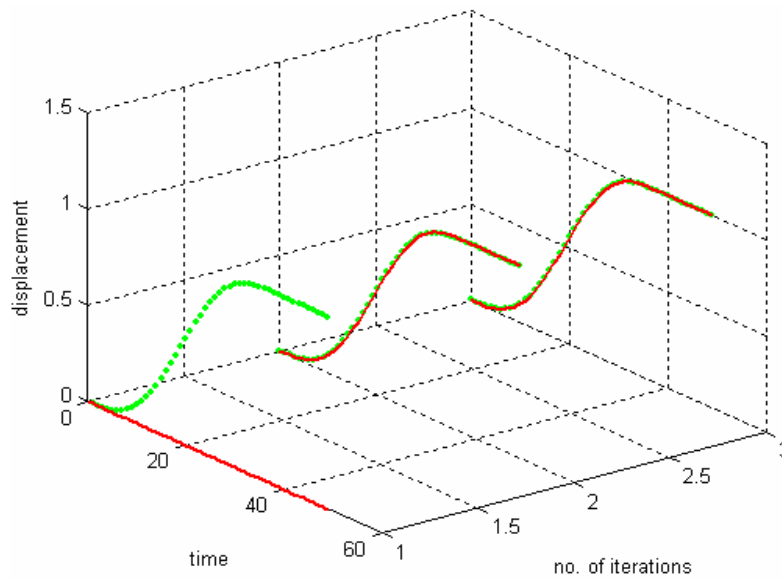


Figure 6.12: Desired output (dotted lines) being learnt at 30<sup>th</sup> run.

The output follows the desired output.

### 6.1.1.2 Case 2: With initial guess

If the range of output gain can be determined, an initial guess at the output membership functions can be made. Suppose that the maximum value of gain that can be given is  $K_{max}$ . Research results have shown that 20%, 30% and 70% of this value is a good start for initial values of  $K_l$ ,  $K_m$  and  $K_h$  respectively. Other values can also be chosen with the following restriction.

$$K_l < K_m < K_h < K_{max} \quad (6.4)$$

The proposed initial values of  $K_l$ ,  $K_m$  and  $K_h$  are given as

$$K_{li} = 0.2 * K_{max} \quad (6.5)$$

$$K_{mi} = 0.3 * K_{max} \quad (6.6)$$

$$K_{hi} = 0.7 * K_{max} \quad (6.7)$$

Using these equations, for the car suspension system the values come out to be  $K_{li} = 6$ ,  $K_{mi} = 12$  and  $K_{hi} = 21$ . Here  $K_{li}$ ,  $K_{mi}$  and  $K_{hi}$  are the guessed values for  $K_l$ ,  $K_m$  and  $K_h$ . With these values the system performance is shown in figure 6.13.

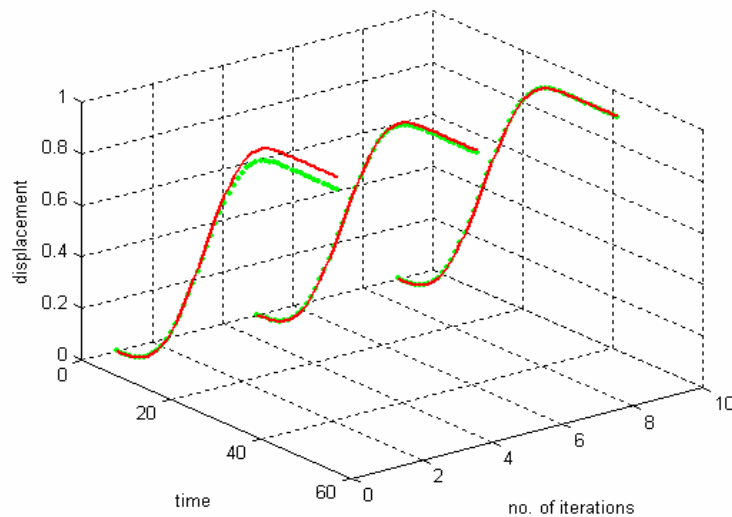


Figure 6.13: Behaviour of the system at first run.

The systems converged in 9 iterations. The system again learnt a piece-wise-linear control surface. The surface is plotted in figure 6.14.

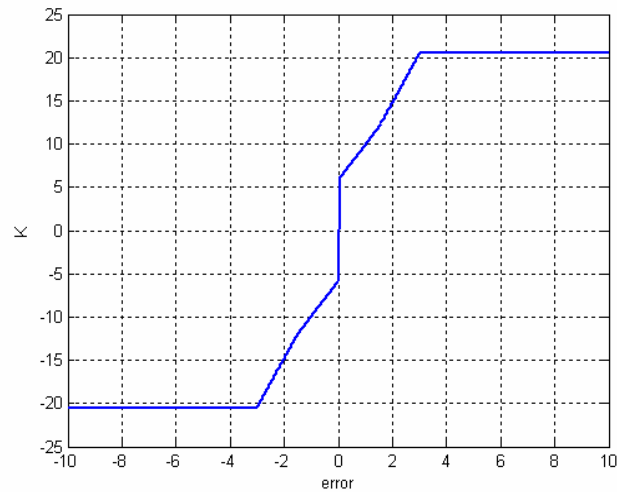


Figure 6.14: Learnt control surface.

The surface gets smoother as number of membership functions is increased. The behaviour of error, during this run is shown in figure 6.15.

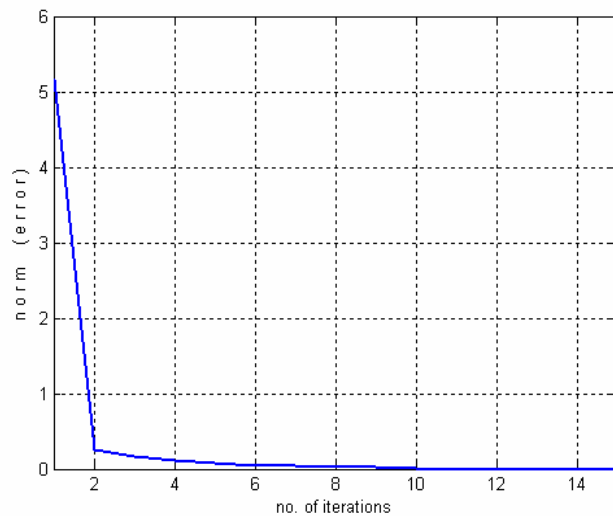


Figure 6.15: Plot of error vs. number of iterations.

The error is reduced in lesser number of iterations as compared to case 1.



## 6.2 Stability

The easiest and so far the only reliable way to prove the stability of a fuzzy based system is the linguistic trajectory method. In this method the firing of rules are plotted to get a measure of which rules are dominant. To plot this trajectory each rule was assigned a number. The rules with their assigned numbers are tabulated in table 6.3.

Rules	Rule No.
if error is NH then K is Kl	1
if error is NM then K is Km	2
if error is L then K is Kl	3
if error is M then K is Km	4
if error is H then K is Kh	5

Table 6.3: Assignment of a number, to each rule, in the rule base.

The firing of rules, for the car suspension system, with no initial guess, at 30<sup>th</sup> run is shown in table 6.4.

Sample no.	Rules				
<b>1</b>	1	2	3	4	5
<b>2</b>	1	2	3	4	5
<b>3</b>	1	2	3	4	5
⋮	⋮	⋮	⋮	⋮	⋮
<b>70</b>	1	2	3	4	5

Table 6.4: Linguistic trajectory.

The table shows all the rules in each row. Each row represents a sample. At each sample, the rules fired are shown with shaded blocks. In this case the firing trajectory has a converging behaviour.

### **6.3 Summary**

This chapter presents a fuzzy based Iterative Learning Controller (ILC) named Fuzzy Iterative Learning Controller (FILC). The gains of the ILC are calculated by the Iterative Learning Fuzzy Tuner (ILFT) module of the FILC. The ILFT adjusts its rule base iteratively to optimize ILC's gains. The ILFT block was able to learn membership functions without any prior knowledge about the system dynamics.

Results from a car suspension system are also presented. These results show excellent performance. To reduce learning time, a procedure is also presented for initial output membership function formulation. Stability of the controller is tested using linguistic trajectory analysis.

## 7 CONCLUSIONS AND RECOMMENDATIONS

The best known and most used controllers in industrial control processes are proportional-integral (PI) and proportional-integral-derivative (PID) controllers. Designing and implementing these controllers have difficulties associated with them, namely:-

- (a) They require a detailed knowledge of the model of the plant or process to be controlled. Such a model rarely exists.
- (b) Plants, controllers, environments and their constraints may vary with time. These variations can cause unexpected changes in the performance indices.
- (c) They are designed to operate at a specific set point, and hence lack flexibility.
- (d) The real world devices, systems and processes are nonlinear. Finding the models of today's complex devices, systems and processes is very difficult if not impossible. Therefore, researchers generally try to develop equivalent linearised model. This linear model is too restrictive and does not represent the actual dynamics of the system.
- (e) Multi loop and multi variable systems are interdependent and have very complex constraints and dependencies. Conventional controller performances are affected by these constraints and dependencies.
- (f) Even after theoretical design, extensive tuning is required for getting optimal performance.

In view of the above highlighted constraints, it is imperative that new methodologies be researched and workable solutions be evolved. We need a control philosophy, which is:-

- (a) More general in its scope of operation.
- (b) Not dependent on detailed system knowledge.
- (c) Capable of handling real world imprecisions and imperfections inherent in any engineering application.

- (d) Capable of mimicking the human expertise.
- (e) Capable of learning from experience.
- (f) Easy to design and easy to alter.
- (g) More robust and can cover a wider range of operating conditions.
- (h) Cost effective.

Development of Iterative Learning philosophy and Fuzzy Logic has ushered in a new era of controller design. Iterative Learning Controllers mimic the human learning process and are cheap to develop. Fuzzy controllers mimic the human perception based approach and do not require system model. Also, Fuzzy controllers can handle imperfections and imprecisions. To give more robustness to ILC we need to incorporate adaptivity as well. To achieve all these capabilities, we need to develop a new hybrid approach for designing intelligent controllers.

## **7.1 Conclusion**

In this thesis, new algorithms for Iterative Learning Control, adaptive Iterative Learning Control and hybrids of Iterative Learning and Fuzzy Logic Control have been derived and their convergence properties analysed. All the controllers are made to track different reference signals. Because of the repetitive nature of the algorithms, information learnt from previous executions of the tasks is used to improve the tracking performance. This results in learning algorithms which find the input that result in perfect tracking.

The author started with research in Iterative Learning and consequently Iterative Learning Control (ILC). Iterative Learning Control was found to have short comings like slow convergence, non adaptivity, model dependency and complex mathematical structure. After the basic introduction of ILC and Fuzzy in chapter 1, chapter 2 starts with the development of a frame work for the controllers to be developed later. Using this frame work “One Sample At a Time Iterative Learning Controller (OSATILC)”, “Multiple Samples At a Time Iterative Learning Controller (MSATILC)” and “Modified Multiple Samples At a Time Iterative Learning Controller (MMSATILC)” were developed. Many simulations and an experimental setup using M-850 Hexapod from

Physik Instrumente (PI) were used to confirm their performances. The hexapod tracks a laser in real time but the learning is done off line. The conditions for convergence were also formulated. The performance of the three controllers is tabulated in Table 7.1.

Approach \ System	Classical ILC (iterations)	OSAT ILC (iterations)	MSAT ILC (iterations)	MMSAT ILC (iterations)
SS	146	2086	63	1
CCS	DNC	686007	11629	8
CSS	1335	5083	886	2
NLS	DNC	877	2141	228
INVPL	DNC	1763	2316	402

Table 7.1: Comparative performance of controllers presented in chapter 2.

The table above shows the number of iterations to converge for five systems. These systems are described in Appendix A. The symbol DNC means “Did Not Converge”.

More robust adaptive ILCs are developed in chapter 3. These adaptive ILCs have the capability to readjust and learn just as humans do. A comprehensive mathematical base is developed to prove the stability and convergence of these schemes. Using innovative cost functions and introduction of the concept of adaptive step size in ILC, provide a lot of freedom to fine tune the presented schemes. These algorithms were tested using simulations and a practical setup, made from DC motor kit by Quanser Consulting Inc. The schemes have the capability to learn and adapt in real time. The performance of 4 adaptive ILCs, presented in chapter 3, is shown in table 7.2.

Controller \ System	SS	CCS	CSS	NLS
Classical ILC (iterations)	146	DNC	1335	DNC
Approach-3 (iterations)	6	2328	83	6
Approach-3 at 10 <sup>th</sup> run (iterations)	2	768	17	3
Approach-4 (iterations)	44	426	238	30
Approach-4 at 10 <sup>th</sup> run (iterations)	17	73	25	8
Approach-5 (iterations)	18	254	54	13
Approach-5 at 10 <sup>th</sup> run (iterations)	4	15	13	3

Table 7.2: Comparative performance of 4 main ILCs presented in chapter 3.

Here, Approach-3 (section 3.4) uses system identification; Approach-4 (section 3.5.1) uses innovative cost functions with gradient descent, and Approach-5 (section 3.6) uses “Iterative Learning Gain”, to reduce iterations. The table shows the numbers of iterations decreasing as the Approaches are used repeatedly.

To incorporate model independence, perception and linguistic based capabilities, Fuzzy Logic was added in the controllers. The remaining chapters present controllers with both Iterative Learning and Fuzzy Logic working together. Any fuzzy based design is not without difficulty. Fuzzy designers not only have to deal with uncertainties in linguistic terms and design of membership functions but also uncertainties about the input, control output, change in operating conditions and noisy data etc. The main uncertainty is in the selection of membership functions. Chapter 4, apart from presenting a methodology to combine ILC and Fuzzy Logic, also resolves this uncertainty. Moreover, this uncertainty is linked with steady state error and percentage overshoot. These performance requirement parameters are normally given to control system designers. The hybrid methodology in chapter 4 is called “Iterative Learning Fuzzy Tuner (ILFT)”. A number of simulation results are presented. A novel stability analysis methodology is also developed, using piece wise linear approach. To see the effectiveness of this controller, a Two Degree Of Freedom Tracking Device (S-101) was constructed. This device has a camera mounted on it to recognize a moving target. The hybrid controller tracks the target in real time. The results of this experiment are presented in chapter 4. One such result where the S-101 is tracking a moving target is shown in figure 7.1.

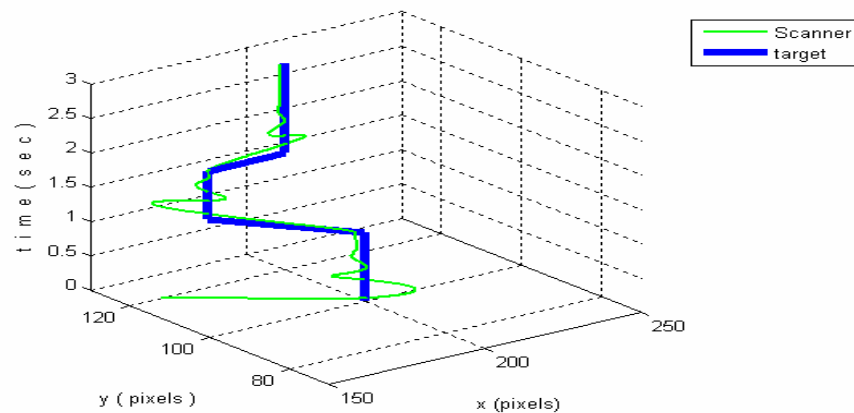


Figure 7.1: Tracking a moving target.

Chapter 5 discusses the use of combining Iterative Learning and Fuzzy Logic for scheduling the gains of the P, PI and PID controllers. The result is a controller named “Iterative Learning Fuzzy Gain Scheduler (ILFGS)”. This novel approach also produced excellent results and performed much better than the conventional controllers. The controller was also made to track desired-speed trajectories using the DC motor kit by Quanser Consulting Inc. The output of the motor trying to achieve a speed of 100 rad./sec., with less than 2% steady state error and less than 5% over shoot, is presented in figure 7.2.

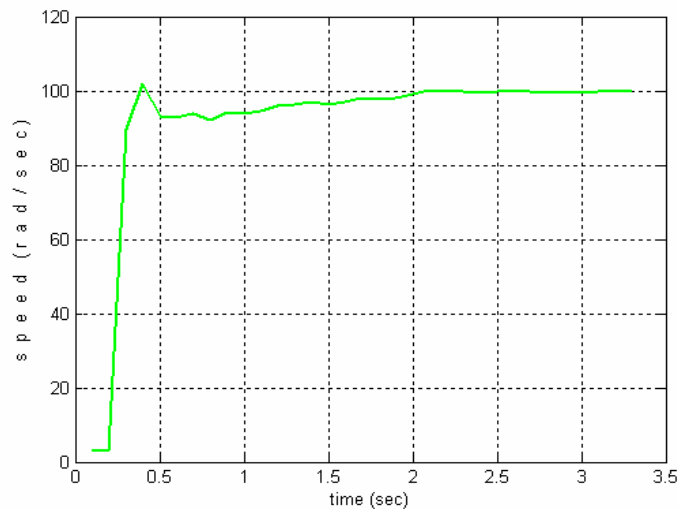


Figure 7.2: Motor speed against a desired speed of 100 rad. /sec.

Chapter 6 aims at making Fuzzy help ILC to perform its task. Fuzzy Logic, because of its non-linear behaviour, was able to adapt ILC gains. This adaptation was defined using ordinary language statements (rules). The hybrid controllers in chapter 4, 5 and 6 indirectly learn the rule base because of adaptivity in membership functions. The controller presented in chapter 6 is named “Fuzzy Iterative Learning Controller (FILC)”. A three dimensional plot of the output of a car suspension system given is Appendix A, trying to track a desired trajectory, is shown in figure 7.3. In this figure the dotted lines show the desired response while the solid lines show the system output.

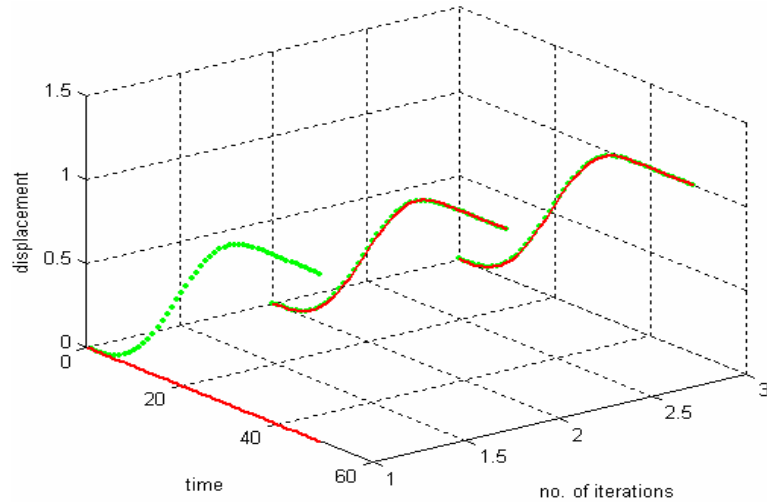


Figure 7. 3: Car Suspension System achieving the desired trajectory.

Combination of Adaptive ILC and Fuzzy Logic produced robust, learning controllers that can adapt with changing conditions. They do not require detailed knowledge about the plant, and hence, can control complex systems.

## 7.2 Recommendations

The research accomplished and described in this thesis has mainly focused on merging Fuzzy Logic and Iterative Learning Control philosophies to realise an intelligent controller. This experience has brought to light a number of avenues still remaining to be investigated. In this context, the following recommendations are presented:-

- (a) The 2-D framework presented in section 2.2 can be further exploited by incorporating weighted values of past inputs. This will result in change in control law of equation 2.4. Furthermore, the effect of higher order terms of this control law should be further studied. We recommend making the gains of these higher order terms decrease with reducing error to avoid instability. The rate of decrease of these gains is another area open for research.
- (b) Knowledge learnt, to achieve a desired response, should be used to track other similar desired responses. For this, a link between desired responses should be



established. This link should have a “degree of similarity indicator”. This indicator should be used to decide how much learnt knowledge to use for the new desired response. Also, how to use this info needs to be worked out.

- (c) The hybrid controllers presented in chapter 4,5 and 6 are aimed to cater for percentage overshoot and steady state error performance indices. Peak time is another important design parameter that can be studied.
- (d) Methodologies need to be formulated for Multiple-Input and Multiple-Output (MIMO) systems.
- (e) Zooming camera on the devices shown in section 2.10 and 4.5.2 can reduce the steady state error further. The camera can zoom in on target as the error decreases. This will increase the accuracy of the system.
- (f) Target Simulation Board (TSB) presented in section 4.5.2.3 consists of 36 LEDs and a remote console to generate target trajectories. The density of the LEDs should be increased so that highly non-linear and smooth trajectories can also be generated. Moreover, this board should be linked with another Personal Computer (PC) to generate stored trajectories.
- (g) The research has mainly concentrated on triangular Membership Functions due to the ease of implementation for microcontroller based solutions. For more complex systems, the impact of Gaussian Membership Functions on the response should be studied.

A very significant lesson gleaned from this research is, *“Combining proven methodologies and functionalities holds the promise to create new knowledge for designing intelligent systems.”*

## REFERENCES

- [1] A. D. Luca, G. Paesano and G. Ulivi, "A frequency domain approach to learning control: implementation for a robot manipulator", IEEE transactions on industrial electronics, vol. 39, no. 1, pp. 1-10, 1992.
- [2] A. Rafael, C. Jorge, C. Oscar, G. Antonio and H. Francisco , "A genetic rule weighting and selection process for fuzzy control of heating, ventilating and air conditioning system", Elsevier Engineering Applications of Artificial Intelligence, vol. 18, pp. 279-296, 2005.
- [3] A. Irtaza, H. Ali, F. Salman, M. Khalid, M. Bilal and Tanveer, "Development of a 2 Degree of Freedom Tracking System Part II: Controller Design and Implementation", IEEE International Conference on Emerging Technologies, Islamabad, Pakistan, 2005.
- [4] A. Tanweer, Y. K. Muhammad, B. Masood-ul-Haq and M. Khalid, "Development of a 2 Degree of freedom tracking system: Design and fabrication of platform", IEEE International Conference on Emerging Technologies, Islamabad, Pakistan, 2005.
- [5] A. Tayebi, "Adaptive Iterative Learning Control for Robot Manipulators", Automatica 40, pp. 1195-1203, 2004.
- [6] A. Zilouchian, "An iterative learning control technique for a dual arm robotic system", Proceedings of the IEEE international conference on robotics and automation", San Diego, CA, vol. 4, pp. 1528-1533, 1994.
- [7] B.C. Kuo and F. Golnarahgi, "Automatic control system", 8<sup>th</sup> edition, NY, USA, John Wiley & Sons, Inc., 2002.
- [8] B. S. Zhang and J. R. Leigh, "Predictive time sequence iterative learning control with application to a fermentation process", Proceedings of IEEE international Conference on control and applications, Vancouver, Canada, vol. 2, 1993.
- [9] C. G. Nesler, "Adaptive control of thermal processes in buildings", IEEE Control System Magazine, vol. 6, no. 4, pp. 9-13, 1986.

- [10] C.H. Choi and T.J. Jang, "Iterative learning control in feedback systems based on an objective function", *Asian Journal of control*, vol. 2, no. 2, pp. 101-110, June 2000.
- [11] C. J. Li, H. S. M. Beigi and S. Li, "Nonlinear piezo-actuator control by self tuning regulator", *Journal of dynamic systems, measurement and control*, vol. 115, pp. 720-723, 1993.
- [12] C. Mi, H. Lin and Y. Zhang, "Iterative Learning control of antilock braking of electric and hybrid vehicles", *IEEE transactions on vehicular technology*, vol.54, no.2,pp. 486-494, 2005.
- [13] C. Smith and M. Tomizuka, "Shock rejection for repetitive control using disturbance observer", *Proceedings of the 35<sup>th</sup> IEEE conference on decision and control*, Kobe, Japan, 1996.
- [14] D. A. Linkens and J. Nie, "Fuzzified RBF network-based learning control: structure and self-construction", *IEEE international conference on Neural Networks*, vol. 2, pp. 1016-1021, 1993.
- [15] D. H. Owens, N. Amann and E. Rogers, "Systems structure in iterative learning control", *Proceedings of the international conference on System structure and control*, Nantes, France, pp. 500-505, 1995.
- [16] E. Cam and I. Kocaarslan, "A fuzzy gain scheduling PI controller application for an interconnected electrical power system", *Electrical power systems research*, Elsevier, vol. 73, pp. 267-274, 2005.
- [17] E. Cox, "The fuzzy systems handbook: A practical guide to building, using and maintaining fuzzy systems", Academic Press, second edition, 1999.
- [18] E. H. Mamdani, "Fuzzy control - a misconception of theory and application", *IEEE Expert*, vol. 9, no. 4, pp. 27-28, Aug, 1994.
- [19] E. H. Mamdani, "Twenty years of fuzzy control: Experiences gained and lessons learnt", *Proceedings of second international IEEE international conference on fuzzy systems (FUZZ-IEEE)*, pp. 339-344, 1993.
- [20] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller", *Int. J. Man Mach. Studies*, vol. 7, no. 1, pp. 1-13, 1975.

- [21] E. K. Jerzy, and B. Z. Marek, "Iterative learning control synthesis based on 2-D system theory", IEEE Transactions on Automatic control, vol.38, no.1, pp. 121-124, 1993.
- [22] F. C. H. Rhee, "Uncertain fuzzy clustering: Insights and recommendations", IEEE Computational Intelligence Magazine, vol.2, no.1, pp. 44-56, 2007.
- [23] F. M. Boland and D. H. Owens, "Linear multi-pass processes-a two dimensional interpretation", Proceedings of the IEE, vol. 127, no. 5, pp. 189-193, 1980.
- [24] F. Yong, Y. C. Soh and G. G. Feng, "Convergence analysis of iterative learning control with uncertain initial conditions", Proceedings of the 4<sup>th</sup> world congress on intelligent control and automation, Shanghai, China, pp. 960-963, 2002.
- [25] G. J. Klir and M. J. Wierman, Uncertainty-based information, Physica-Verlag, Heidelberg, Germany, 1998.
- [26] G. J. Klir and T. A. Folger, "Fuzzy sets, uncertainty and information", Prentice Hall, Englewood Cliffs, NJ, 1988.
- [27] G. Li, H. Li and J. X. Xu, "The study and design of the longitudinal learning control system of the automobile", Proceedings of the 2<sup>nd</sup> Asian control conference, Seoul, Korea, 1997.
- [28] G. Zeng and M. Jamshidi, "Learning control system analysis and design based on 2-D theory", Journal of Intelligent and Robotic Systems 3, Kluwer Academic Publishers, Netherland, pp. 17-26, 1990.
- [29] H. B. B. Abad, A. Y. Varjani and T. Asghar, "Using fuzzy controller in induction motor speed control with constant flux", Enformatika-Transactions on engineering, computing and technology, vol. 5, pp. 307-310, 2005.
- [30] H. Dou, Z. Zhou, M. Sun and Y. Chen, "Robust high order P-type iterative learning control for a class of uncertain nonlinear systems", Proceedings of the IEEE international conference on systems, man and cybernatics, Beijing, China, vol. 2, 1996.

- [31] H. Hagnas, "Type-2 FLCs: A new generation of fuzzy controllers", IEEE Computational Intelligence Magazine, vol.2, no.1, pp. 30-43, 2007.
- [32] H. Osaka, C. J. Lin, T. Shimogawa and H. Qiu, "Control of mechatronic systems by learning actuator reference trajectories described by B-spline curves", Proceedings of the IEEE international conference on systems, Man and Cybernetics, Vancouver, BC, Canada, vol. 5, pp. 4679-4684, 1995.
- [33] H. S. Ahn, S. H. Lee and D. H. Kim, "Frequency domain design of iterative learning controllers for feedback systems", IEEE international symposium on industrial electronics, Athens, Greece, vol. 1, pp. 352-357, 1995.
- [34] <http://egweb.mines.edu/faculty/kmoore/>
- [35] <http://en.wikipedia.org/wiki/Aristotle>
- [36] <http://en.wikipedia.org/wiki/Buddha>
- [37] [http://en.wikipedia.org/wiki/Georg\\_Cantor](http://en.wikipedia.org/wiki/Georg_Cantor)
- [38] [http://en.wikipedia.org/wiki/Mitsubishi\\_Lancer\\_Evolution](http://en.wikipedia.org/wiki/Mitsubishi_Lancer_Evolution)
- [39] [http://en.wikipedia.org/wiki/Sendai\\_City\\_Subway\\_Line](http://en.wikipedia.org/wiki/Sendai_City_Subway_Line)  
<http://osamuabe.id.infoseek.co.jp/subway/maincity/sendai/sendai.htm>  
<http://www.youtube.com/watch?v=N4V3vGXMONA&mode=related&search=>
- [40] [http://en.wikipedia.org/wiki/Spectral\\_radius](http://en.wikipedia.org/wiki/Spectral_radius)
- [41] [http://engineering.utsa.edu/EE/faculty\\_staff/jamshidi.html](http://engineering.utsa.edu/EE/faculty_staff/jamshidi.html)
- [42] <http://factae.elfak.ni.ac.yu/facta9801/inmemoriam.html>  
Yakov Zalmanovich Tsytkin
- [43] <http://mathematica.ludibunda.ch/fuzzy-logic7.html>
- [44] <http://mechatronics.ece.usu.edu/yqchen/>  
<http://www.ece.usu.edu/csois/people/yqchen/picture/index.html>  
Dr. YangQuan Chen
- [45] <http://sipi.usc.edu/~mendel/>  
Dr. J. M. Mendel
- [46] <http://web.abo.fi/~rfuller/fuzs.html>

- [47] <http://wing.comp.nus.edu.sg/pris/FuzzyLogic/HistoricalPerspectiveDetailed.html>
- [48] <http://www.aptronix.com/>
- [49] <http://www.computerworld.com/>  
<http://www.computerworld.com/news/2004/story/0,11280,95282,00.html>
- [50] <http://www.cs.berkeley.edu/~zadeh/>  
<http://www.eecs.berkeley.edu/Faculty/Homepages/zadeh.html>
- [51] <http://www.cs.berkeley.edu/~wkahan/>  
[http://jurist.law.pitt.edu/views/blogs/tillers/2005\\_07\\_10\\_archive.htm](http://jurist.law.pitt.edu/views/blogs/tillers/2005_07_10_archive.htm)
- [52] <http://www.ece.ust.hk/~eewang/>  
 Dr Li-Xin Wang
- [53] <http://www.ecs.soton.ac.uk/people/etar>
- [54] <http://www.hi.cs.meiji.ac.jp/~takagi/index.en.html>  
 Professor Dr. Tomohiro Takagi
- [55] [http://www.ieee.org/web/aboutus/history\\_center/biography/kalman.html](http://www.ieee.org/web/aboutus/history_center/biography/kalman.html)  
[http://jurist.law.pitt.edu/views/blogs/tillers/2005\\_07\\_10\\_archive.htm](http://jurist.law.pitt.edu/views/blogs/tillers/2005_07_10_archive.htm)
- [56] <http://www.ortech-engr.com/fuzzy/togai.html>
- [57] [http://www.quanser.com/english/html/products/template\\_switch.asp?lang\\_code=english&pcat\\_code=exp-mec&prod\\_code=S24-QET&tmpl=](http://www.quanser.com/english/html/products/template_switch.asp?lang_code=english&pcat_code=exp-mec&prod_code=S24-QET&tmpl=)
- [58] [http://www.ritsumei.ac.jp/~arimoto/bio\\_e.html](http://www.ritsumei.ac.jp/~arimoto/bio_e.html)  
 Professor Dr. Suguru Arimoto
- [59] <http://www.type2fuzzylogic.org/>
- [60] <http://www.shef.ac.uk/acse/staff/dho>
- [61] <http://www.xbitlabs.com/articles/storage/display/seagate-u6.html>
- [62] H. X. Li and H. B. Gatland, "Conventional fuzzy control and its enhancement", IEEE Transactions on systems, man and cybernetics-Part B: Cybernetics, vol.26, no.5, pp. 791-797, 1996.
- [63] H. Ying, Y. Yongquan and Z. Tao, "A new real-time self-adaptive rule modification algorithm based on error convergence in fuzzy control", IEEE international conference on Industrial Technology, ICIT, pp. 789-794, 2005.

- [64] J. E. Kurek, "Stability of nonlinear parameter varying digital 2-D systems", IEEE transactions of automatic control, vol. 40, no. 8, pp. 1428-1432, 1986.
- [65] J. E. Kurek and M. B. Zaremba, "Iterative learning control synthesis based on 2-D system theory", IEEE transactions on automatic control, vol. 38, no. 1, pp. 121-125, 1993.
- [66] J. Gertler and H.S. Chang, "An instability indicator for expert control", IEEE Control System Magazine, vol. 6, no. 4, pp. 14-17, 1986.
- [67] J. H. Moon, M. N. Lee, M. J. Chung, S. Y. Jung and D. H. Shin, "Track following control for optical disk drives using an iterative learning scheme", IEEE Transactions on Consumer Electronics, vol. 42, no. 2, pp. 192-198, 1996.
- [68] J. H. Moon, T. Y. Doh and M. J. Chung, "An iterative learning control scheme for manipulators", Proceedings of the IEEEERSJ international conference on intelligent robots and systems, Grenoble, France, vol. 2, pp. 759-765, 1997.
- [69] J. Hu and M. Tomizuka, "Adaptive asymptotic tracking of repetitive signals – a frequency domain approach", IEEE transactions on Automatic Control, vol. 38, no. 19, pp. 1572-1579, 1993.
- [70] J. Lee, "On methods for improving performance of PI type fuzzy logic controllers", IEEE transactions on fuzzy systems, vol. 1, no. 4, pp. 298-301, 1993.
- [71] J. M. Mendel, "Type-2 fuzzy sets and systems: an overview", IEEE Computational Intelligence Magazine, vol.2, no.1, pp. 20-29, 2007.
- [72] J. M. Mendel, "Fuzzy sets for words: a new beginning", Proceedings of the IEEE international conference on fuzzy systems", St. Louis, MO., pp. 37-42, 2003.
- [73] J. M. Mendel, "UNCERTAIN rule-based fuzzy logic systems: Introduction and new directions", NJ, Prentice Hall, 2001.
- [74] J. Nie and D. Linkens, "Fuzzy-neural control : Principles, algorithms and applications", Prentice Hall, India, 1995.

- [75] J. S. R. Jang, C.T. Sun and E. Mizutani, "Neuro fuzzy and soft computing", Prentice Hall, USA, 1997.
- [76] J. W. Perng, B.F. Wu, H. I. Chin and T. T. Lee, "Gain phase margin analysis of dynamic fuzzy control systems", IEEE transactions on systems, man and cybernetics-part B, vol. 34, no.5, pp. 2133-2139, 2004.
- [77] J. X. Xu., "Direct learning of control input profiles with different time scales", Proceedings of the 35<sup>th</sup> IEEE conference on decision and control, Kobe, Japan, 1996.
- [78] J. X. Xu, Y. Dote, X. Wang and C. Shun, "On the instability of iterative learning control due to sampling delay", Proceedings of the 1995 IEEE IECON 31<sup>st</sup> international conference on industrial electronics, control and instrumentation, Orlando, FL, vol. 1, pp. 150-155, 1995.
- [79] J. X. Xu and Y. Song, "Direct learning control scheme with an application to a robotic manipulator", Proceedings of the 2<sup>nd</sup> Asian Control Conference, Seoul, Korea, 1997.
- [80] J.Y. Choi and J. S. Lee, "Adaptive Iterative Learning Control of Uncertain Robotic Systems", IEE Proc.-Control Theory Appl., vol. 147, no. 2, pp. 217-223, 2004.
- [81] K. Furuta and M. Yamakita, "The design of a learning control system for multivariable systems", Proceedings of IEEE international symposium on intelligent control, Philadelphia, Pennsylvania, pp. 371-376, 1987.
- [82] K. L. Anderson, G.L. Blankenship and L.G. Lebow, "A rule based adaptive PID controller", Proceedings of the 27<sup>th</sup> IEEE conference on Decision and Control, Austin, USA, vol. 1, pp. 564-569, 1988.
- [83] K. L. Moore, "Iterative learning control for deterministic systems", Springer-Verlag, London, 1993.
- [84] K. L. Moore, M. Dahleh and S. P. Bhattacharyya, "Iterative learning control: A survey and new results", Journal of Robotic Systems vol. 9, no. 5, pp. 563-594, 1992.
- [85] K. L. Moore and Y. Q. Chen, "A separative high order framework for monotonic convergent iterative learning controller design", Proceedings of



- the American control conference, Denver, Colorado, pp. 3644-3649, 2003.
- [86] K. P. Venugopal, R. Sudhakar and A. S. Pandya, "On line learning control of autonomous underwater vehicles using feedforward neural networks", IEEE Journal of Oceanic Engineering, vol. 17, pp. 308-319, 1992.
- [87] K. Teng-Kai, F. Li-Che, J. Jong-Hann, C. Peri-Ying and C. Yu-Ming, "Zoom-Based head tracker in complex environment", Proceedings of the 2002 IEEE International Conference on Control Applications, Glasgow, Scotland, UK, 2002.
- [88] K. Teng-Kai, H. Chen-Ming, F. Li-Chen and C. Pei-Ying, "A robust visual servo based headtracker with auto-zooming in cluttered environment", Proceedings of the American Control Conference, Denver, Colorado, 2003.
- [89] L. A. Zadeh, "From computing with numbers to computing with words-From manipulation of measurements to manipulation of perceptions", Int. J. Appl. Math. Comput. Sci., vol. 12, no. 3, pp.307-324, 2002.
- [90] L. A. Zadeh, "From computing with numbers to computing with words-From manipulation of measurements to manipulation of perceptions", IEE transactions on circuits and systems 1: Fundamental Theory and application", vol. 4, pp.105-119, 1999.
- [91] L.A. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic", Fuzzy Sets and Systems vol. 90, no. 2, pp. 111-127, 1997.
- [92] L. A. Zadeh, "A fuzzy algorithmic approach to the definition of complex or imprecise concepts", Int. J. man-machine studies, vol. 8, pp.249-291, 1976.
- [93] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes", IEEE transactions on systems, man and cybernatics", vol. 3, no. 1, pp.28-44, 1973.
- [94] L. A. Zadeh, "Fuzzy Sets", Information and Control, vol. 8, pp.338-353, 1965.
- [95] L. G. Sison and E. K. P. Chong, "No-reset iterative learning control", Proceedings of the 35<sup>th</sup> conference on decision and control, Kobe, Japan,

- pp. 3062-3063, 1996.
- [96] L. Ljung, "System Identification: Theory for the user", 2<sup>nd</sup> Edition, PTR Prentice Hall, Upper Saddle River, N.J., 1999.
- [97] L. M. Hideg, "Stability and convergence issues in iterative learning control – II", Proceedings of the 1996 IEEE international symposium on intelligent control, Dearborn, MI, pp. 480-485, 1996.
- [98] L. Hideg and R. Judd, "Frequency domain analysis of learning systems", Proceedings of the 27<sup>th</sup> conference on decision and control, Austin, Texas, pp. 586-591, 1988.
- [99] L. M. Hideg, "Time delays in iterative learning control schemes", Proceedings of the 1995 IEEE international symposium on intelligent control, Monterey, CA, pp. 5-20, 1995.
- [100] L. P. Zhang and F. Yang, "Fuzzy iterative learning control design for output tracking of discrete time fuzzy systems", Proceedings of the third international conference on machine learning and cybernetics, Shanghai, 2(2004), pp.678-682, 2004.
- [101] L. Reznik, "Fuzzy Controllers", Newnes, A division of reed educational and professional publishing ltd., Oxford, 1997.
- [102] L. Stotts, B. H. Kleiner, "New developments in fuzzy logic computers", Industrial Management & Data Systems, vol.95, no.5, pp. 13-17, 1995.
- [103] L. X. Wang, "A course in fuzzy systems and control", Prentice Hall PTR, Upper Saddle River, NJ, 1997.
- [104] M.A. Rodrigo, A. Seco, J. Ferrer and J.L. Penya-roja, "Non-linear control of an activated sludge aeration process: use of fuzzy techniques for tuning PID controllers", ISA transactions, vol. 38, no. 3, pp. 231-241, 1999.
- [105] M. Norrlof, "An Adaptive Iterative Learning Control Algorithm With Experiments on an Industrial Robot", IEEE Transactions On Robotics and Automation ,vol. 18, no.2, pp. 245-251, 2002.
- [106] M. Norrlof and S. Gunnarsson, "Experimental Comparison of some classical iterative learning control algorithms", IEEE Transactions On Robotics and Automation ,vol. 18, no.4, pp. 636-641, 2002.

- [107] M. Pandit and K. H. Buchheit, "Optimizing iterative learning control of cyclic production processes with application to extruders", IEEE transactions on control systems technology, vol. 7, no. 3, pp. 382-390, 1999.
- [108] M. Pandit and S. Baque, "Learning control of cyclic production processes", Proceedings of the 1997 6<sup>th</sup> ETFA international conference on emergin technologies and factory automation, Los Angeles, CA, USA, pp. 64-70, 1997.
- [109] M. Phan and R. W. Longman, "A mathematical theory of learning control for linear discrete multivariable systems", Proceedings of the AIAA/AAS Astrodynamics Conference, Minneapolis, Minnesota, pp. 740-746, 1988.
- [110] M. Yamakita, M. Ueno and T. Sadahiro, "Trajectory tracking control by an adaptive iterative learning control with artificial neural network", Proceedings of the American Control Conference, Arlington, VA, pp. 1253-1255, 2001.
- [111] M. Sun, B. Huang, X. Zhang and Y. Chen, "Robust convergence of D-type learning controller", Proceedings of the 2<sup>nd</sup> Chinese world congress on intelligent control and intelligent automation, Xian, China, 1997.
- [112] N. Amann and D. H. Owens, "Non-minimal phase plants in iterative learning control", Second international conference on intelligent systems engineering, Hamburg, Germany, pp. 107-112, 1994.
- [113] N. Amann, D. H. Owens and E. Rogers, "Iterative learning control using optimal feedback and feed forward actions", International journal of control, vol. 65, no. 2, pp. 277-293, 1996.
- [114] N. Amann, D. H. Owens and E. Rogers, "Robustness of norm-optimal iterative learning control", Proceedings of international conference on control, Exeter, UK, vol. 2, pp. 1119-1124, 1996.
- [115] N. Amann, D. H. Owens and E. Rogers, "Iterative learning control for discrete time systems with exponential rate of convergence", IEE proceedings on Control Theory and Applications, vol. 143, vol. 2, pp. 217-224, 1996.

- [116] N. Amann, D. H. Owens and E. Rogers, "New results interactive learning control", International conference on control, Coventry, UK, vol. 1, pp. 640-645, 1994.
- [117] P. Lucibello, "On the role of high gain feedback in P-type learning control of robots", Proceedings of the 32<sup>nd</sup> IEEE conference on decision and control, San Antonio, Texas, USA, pp. 2149-2152, 1993.
- [118] P. Y. Tsai, H. C. Huang, Y. J. Chen and R. C. Hwang, "The model reference control by auto-tuning PID like fuzzy controller", Proceedings of the 2004 IEEE international conference on control applications, Taipei, Taiwan, pp. 406-411, September 2-4, 2004.
- [119] R. John and S. Coupland, "Type 2 fuzzy logic: A historical view", IEEE Computational Intelligence Magazine, vol.2, no.1, pp. 57-62, 2007.
- [120] R. Longman, M. Q. Plan and J. Juang, "An overview of a sequence of research developments in learning and repetitive control", Proceedings of the first international conference on motion and vibration control, Yokohama, Japan, September 1992.
- [121] R. M. Milasi, M. R. Jamali and C. Lucas, "Intelligent Washing Machine: A Bio inspired and Multi-objective Approach", International Journal of Control, Automation and Systems, vol.5, no.4, pp. 436-443, 2007.
- [122] R. P. Judd, R. P. Van Til and L. Hideg, "Equivalent Lyapunov and frequency domain stability conditions for iterative learning control systems", Proceedings of the 8<sup>th</sup> IEEE international symposium on intelligent control, pp. 487-493, 1993.
- [123] R. P. Roesser, "A discrete state space model for linear image processing", IEEE transactions on automatic control, vol.20, no.1, pp. 1-10, 1975.
- [124] R. W. Longman, "Design methodologies in learning and repetitive control", Proceedings of the 2<sup>nd</sup> Asian control conference, Seoul, Korea, 1997.
- [125] S. Arimoto, S. Kawamura and F. Miyazaki, "Bettering operation of robots by learning", Journal of robotic systems, vol. 1, no. 2, pp. 123-140, 1984.
- [126] S. Ashraf, R.M.Parkin and E. Muhammad, "Iterative learning-based laser

- beam tracker”, *Industrial Robot: An international journal*, vol. 34, no.4, 326-331, 2007.
- [127] S. Gunnarsson and M. Norrlof, “On the design of ILC algorithms using optimization”, *Automatica*, vol. 37, no. 12, pp. 2011-2016, 2001.
- [128] S. Kawamura and N. Fukao, “A time scale interpolation for input torque patterns obtained through learning control on constrained robot motions”, *Proceedings of the 1995 IEEE international conference on robotics and automation*, pp. 2156-2161, 1995.
- [129] S. Kawamura, F. Miyazaki and S. Arimoto, “Realization of robot motion based on a learning method”, *IEEE transactions on Systems, Man and Cybernetics*, vol. 18, no. 1 , pp. 126-134, 1988.
- [130] S. Kawamura, F. Miyazaki and S. Arimoto, “Intelligent control of robot motion based on learning method”, *Proceedings of IEEE international symposium on intelligent control*, Philadelphia, Pennsylvania, pp. 365-370, January 1987.
- [131] S. K. Tso and Y. X. Ma, “Cartesian based learning control for robots in discrete time formulation”, *IEEE transactions on Systems, Man and Cybernetics*, 1992.
- [132] S. Tang, C.C. Hang and J.S. Chai, “Gain scheduling from conventional to newro-fuzzy”, *Automatica*, vol. 33, no. 3, pp. 411-419, 1997.
- [133] S. Xu, J. Lam, Z. Lin, K. Galkowski, W. Paszke, B. Sulikowski, E. Rogers and H. Owens, “Positive real control of two-dimensional systems: Roesser models and linear repetitive processes”, *International journal of control*, vol. 37, no.11, 1047-1058, 2003.
- [134] T. C. Hsia, “System Identification”, Lexington Books, D.C. health and Company, Lexington, Massachusetts, Toronto, 1997.
- [135] T. R. Stefani, C.J. Savant, B. Shahian and G.H. Hostetter, “Design of feedback control systems”, Saunders College publishing, USA, 1994.
- [136] T. P. Blanchett, G.C. Kember and R. Dubay, “PID gain scheduling using fuzzy logic”, *ISA Transaction*, Elsevier, vol. 39, pp. 317-325, 2000.
- [137] T. J. Procyk and E.H. Mamdani, “A linguistic self organizing process

- controller”, *Automatica*, vol. 15, no. 1, pp. 15-30, 1979.
- [138] T. Sogo and N. Adachi, “Convergence rates and robustness of iterative learning control”, *Proceedings of the 35<sup>th</sup> IEEE conference on decision and control*, vol. 3, Kobe, Japan, pp. 3050-3055, 1996.
- [139] V. Hatzikos, J. Hatonen and D.H. Owens, “Genetic algorithms in norm-optimal linear and non-linear iterative learning control”, *International Journal of Control*, vol. 77, no. 2, pp. 188-197, 2004.
- [140] W. C. Kim and K. S. Lee, “Design of quadric criterion based iterative learning control by principal component analysis”, *Proceedings of the 2<sup>nd</sup> Asian control conference*, Seoul, Korea, 1997.
- [141] W. Jouse and J. Williams, “PWR heat up control by means of a self teaching neural network”, *Proceedings of international conference on control and instrumentation in nuclear installations*, Glasgow, 1990.
- [142] W. S. Chang and I. H. Suh, “Analysis and design of dual-repetitive controllers”, *Proceedings of the 35<sup>th</sup> IEEE conference on decision and control*, Kobe, Japan, 1996.
- [143] W. S. Tommy and F. Yong, “An iterative learning control method for continuous time systems based on 2-D theory”, *IEEE transactions on circuits and systems I: Fundamental theory and applications*, vol. 45, no. 4, pp. 683-689, 1998.
- [144] Y. Chen and C.C. Wong, “Implementation of the Takagi Sugeno model based fuzzy control using an adaptive gain controller”, *IEE Proceedings on Control Theory Applications*, vol. 147, no. 5, pp. 509-514, 2000.
- [145] Y. Chen and C. Wen, “Lecture notes in control and information sciences 248”, Springer-Verlag, 1999.
- [146] Y. J. Liang and D. P. Looze, “Performance and robustness issues in iterative learning control”, *Proceedings of 32<sup>nd</sup> IEEE conference on decision and control*, San Antonio, TX, vol. 3, pp. 1990-1995, 1993.
- [147] Y. Yongquan, H. Ying and Z. Bi, “The Dynamic Fuzzy Method to Tune the Weight Factors of Neural Fuzzy PID Controller”, *IEEE Proceedings of The International Joint Conference on Neural Network(IJCNN)*, Budapest,

Hungary, vol.3, pp.2379-2402, 2004.

- [148] Z. Bien and J. X. Xu, "Iterative learning control analysis, design, integration and applications", Kluwer academic publishers, USA, 1998.
- [149] Z. Bien and K. M. Huh, "Higher order iterative control algorithm", IEE proceedings part D, control theory and applications, vol. 136, pp. 105-112, 1989.
- [150] Z. Feng, Z. Zhang and D. Pi, "Open-closed-loop PD-type iterative learning controller for nonlinear systems and its convergence", Proceedings of the 5<sup>th</sup> world congress on intelligent control and automation, Hangzhou, China, pp. 1241-1245, 2004.
- [151] Z. Geng and M. Jamshidi, "Learning control system analysis and design based on 2-D system theory", Journal of Intelligent and Robotic Systems 3:17-26, 1990.
- [152] Z. Geng, M. Jamshidi, R. Carroll and R. Kisner, "A learning control scheme with gain estimator", Proceedings of the 1991 IEEE international symposium on intelligent control, 13-15 August 1991.
- [153] Z. Xinggun, Z. Keding, W. Shenglin, W. Mao and H. Hengzhang, "Iterative learning control for nonlinear systems based on neural networks", IEEE international conference on intelligent processing systems, ICIPS, vol. 1, pp. 517-520, 1997.
- [154] Z.Y. Zhao, M. Tomizuka and S.I. Isaka, "Fuzzy gain scheduling of PID controllers", IEEE transactions on systems, man and cybernetics, vol. 23, no. 5, pp. 1392-1398, 1993.
- [155] Z. Zhong, "An application of H-infinity control and iterative learning control to a scanner driving system", Proceedings of the 3<sup>rd</sup> international conference on computer integrated manufacturing, Singapore, vol. 2, pp. 981-988, 1995.

## Appendix A

For simulation purposes different systems with different characteristics were considered. These are presented in this Appendix.

### **A.1 A Simple System (SS)**

This system consists of a single pole at  $s = -2$ . This is a stable system. The transfer function of the system is given by

$$G_1(s) = \frac{1}{s+2}$$

At a sampling rate of 10 samples per sec. i.e. sampling time of 0.1 ( $T_s=0.1$ ) the discrete transfer function of the Simple System is

$$G_1(z) = \frac{0.09063}{z-0.8187}$$

### **A.2 Cruise Control System (CCS)**

Assuming that there is no inertia in the wheels and that friction is the only thing opposing the motion of the car, the cruise control system can be reduced to a simple mass and damper system shown in figure A 1.



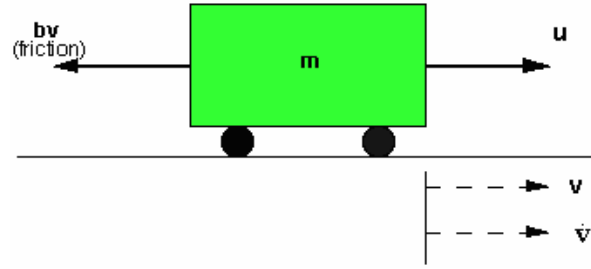


Figure A 1: A cruise control system diagram.

In the figure  $u$  is the input from the engine,  $v$  is the velocity of the body,  $m$  is the mass of the body and  $b$  is the damping. The design values are

$$m = 1000 \text{ Kg}$$

$$b = 50 \text{ N sec/m}$$

$$u = 500 \text{ N}$$

The system transfer function is given by

$$G_2(s) = \frac{1}{ms + b}$$

Using the design values the transfer function becomes

$$G_2(s) = \frac{1}{1000s + 50}$$

The discrete transfer function for a sampling frequency of 10 samples per sec. is

$$G_2(z) = \frac{0.00009754}{z - 0.995}$$

### A.3 Car Suspension System (CSS)

A car suspension system can be modelled with a mass, spring and damper system of the form shown in figure A 2.

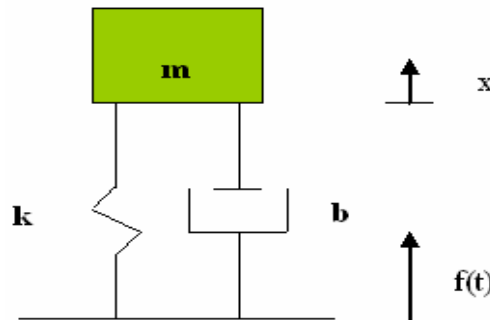


Figure A 2: Block diagram of the car suspension system.

The force produced by the spring is proportional to the translation of the spring. The spring produces a force  $kx$  in the direction of the force as the mass ( $m$ ) is displaced by an amount,  $x$ . As the mass is moved with a positive velocity ( $\frac{dx}{dt}$ ), the damper produces a force,  $b\frac{dx}{dt}$ .

The system equation can be written as

$$F(s) = ms^2X(s) + bsX(s) + kX(s)$$

The transfer function is

$$G_3(s) = \frac{X(s)}{F(s)} = \frac{1}{ms^2 + bs + k}$$

Taking the following design values

$$m = 1 \text{ Kg}, b = 10 \text{ N.s / m}, k = 20 \text{ N/m and } F(s) = 1$$

The transfer function is given by the equation

$$G_3(s) = \frac{1}{s^2 + 10s + 20}$$

The discrete transfer function for a sampling frequency of 10 samples per sec. is

$$G_3(z) = \frac{0.003622z + 0.002596}{z^2 - 1.244z + 0.3679}$$

#### A.4 Non-Linear System (NLS)

A second order non-linear system is given by the following dynamic equation

$$\ddot{y} + 0.1\dot{y} + 0.375y = 0.375u$$

#### A.5 Motor Speed Control System (MSCS)

A DC motor directly provides rotary motion. The electrical circuit of the armature and the free body diagram of the rotor is shown in the figure A 3.

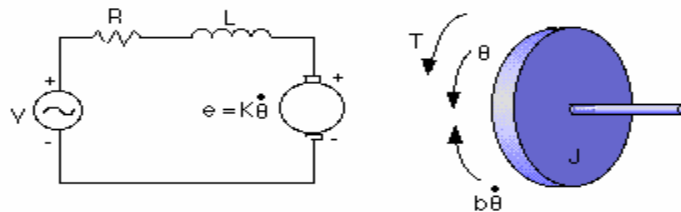


Figure A 3: Free body diagram of the rotor.

Assuming a rigid rotor shaft the following design parameters are assumed

J (moment of inertia of the rotor) = 0.01 Kg.m<sup>2</sup>/s<sup>2</sup>

b (damping ratio of the mechanical system) = 0.1 Nms

$K$  (electromotive force constant) = 0.01 Nm/Amp

$R$  (electrical resistance) = 1 ohm

$L$  (electric inductance) = 0.5 H

$V$  (source voltage)

$\theta$ , Theta (position of the shaft)

The transfer function is given by the equation.

$$G_4(s) = \frac{K}{(Js + b)(Ls + R) + K^2}$$

### A.6 Inverted Pendulum (INVPL)

The inverted pendulum is a nonlinear system. The open loop plant is highly unstable. The goal is to maintain the desired vertically oriented position at all times. The diagram of the whole system is shown below.

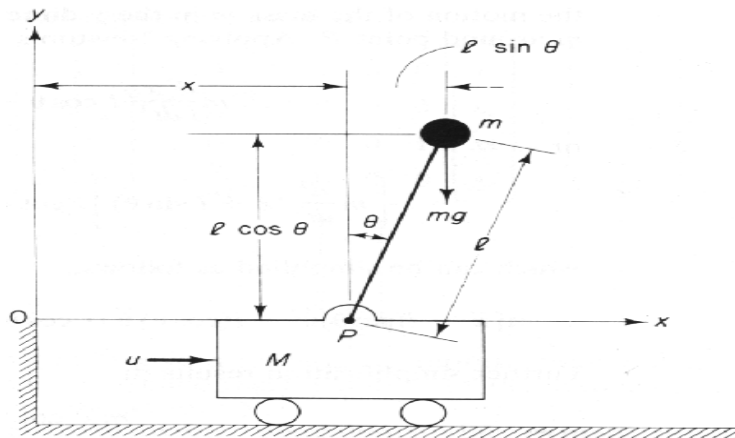


Figure A 4: Inverted pendulum on a cart.

Where

$M$  = Mass of cart, (kg)

$m$  = Mass of inverted pendulum, (kg)

$u$  = External x-directed force, (N)

$g$  = Force of gravity, (m/sec<sup>2</sup>)

$x$  = Cart position, (m)

$\theta$  = Tilt Angle, (radians)

$\ell$  = Lever arm length, (m)

The final state space equations for the inverted pendulum implemented are

$$\frac{d}{dt} \mathbf{z} = \frac{d}{dt} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \frac{d}{dt} \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} z_2 \\ \frac{u \cos z_1 - (M + m)g \sin z_1 + m\ell (\cos z_1 \sin z_1) z_2^2}{m\ell \cos^2 z_1 - (M + m)\ell} \\ z_4 \\ \frac{u + m\ell (\sin z_1) z_2^2 - mg \cos z_1 \sin z_1}{M + m - m \cos^2 z_1} \end{bmatrix} \quad (\text{A } 1)$$

If both the pendulum angle  $\theta(t)$  and the cart position  $x(t)$  are of interest, we have

$$\mathbf{y} = \begin{bmatrix} \theta \\ x \end{bmatrix} = \mathbf{Cz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} \quad (\text{A } 2)$$

Equations (A 1) and (A 2) give a complete state space representation of the nonlinear inverted pendulum. This is the system that was used for simulations.




## A.7 Desired Trajectory

The desired trajectory, used in the thesis is defined by a seventh order polynomial.

$$y_d(t) = -20\left(\frac{t}{40}\right)^7 + 70\left(\frac{t}{40}\right)^6 - 84\left(\frac{t}{40}\right)^5 + 35\left(\frac{t}{40}\right)^4 \quad 0 < t \leq 40$$

$$y_d(t) = 1 \quad t > 40$$

## Appendix B

 Z. Tsypkin [42]	1971	An extraordinary Russian scientist who wrote a book on “Foundations of the theory of learning systems”.
 D. H. Owens [60]	Late 1970’s	Along with E. Rogers gave the Idea of multi pass systems.
Uchiyama	1978	Now believed by some to be the first researcher to give the concept of Learning. But because his contribution was in Japanese most English literature still does not recognise his work.
 S. Arimoto [58]	1984	Generally believed to be the pioneer of Iterative Learning Control. The term learning control is contributed to him. In his words “The learning control concept stands for the repeatability of operating a given object system and the possibility of improving the control input on the basis of previous actual operating data”.
S. Arimoto [58]	1984	S. Arimoto presented the first application of ILC in robotics [125].
J. X.Xu	1990	The concept of Direct learning control was introduced.
Heinzinger	1992	“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”
Luca	1992	“Learning control is a technique in which the input signal required to achieve a given behaviour as output of a











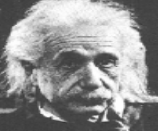





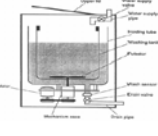

		dynamical system is built iteratively from successive experiments”
 K. L. Moore [34]	1993	“Learning control is an iterative approach to the problem of improving transient behaviour for processes that are repetitive in nature”
Jang	1995	“The main strategy of the Iterative Learning Control is to improve the quality of control iteratively by using information obtained from previous trials, and finally to obtain the control input that causes the desired output”
D. H. Owens [60]	1990’s	Studied the two dimensional nature of ILC and gave $H_\infty$ approach to ILC.
	1990’s	Lots of work done in ILC
Velthuis	2000	Learning feed forward control was introduced.
 Y. Q. Chen [44]	2001	With the collaboration of Seagate implemented ILC in its U6 hard drives.
Goldsmith	2002	“The goal of Iterative Learning Control is to improve the accuracy of a system that repeatedly follows a reference trajectory”
	2004	ILC implemented in Antilock braking of Toyota Prius.

Table B 1: Brief history of ILC.

## Appendix C

 Buddha [36]	(500)BC	Buddhism was founded on the base that world was filled with contradictions, and every thing contained some of its opposite. This means that things can be A and not A at the same time.
 Plato	(427-347)BC	“No chair is perfect; it is only a chair to a certain degree.”
 Aristotle [35]	(384-322)BC	Developed binary logic. It means that thing has to be A or not A, it can't be both.
 Georg Cantor [37]	(1845-1918)	Gave Set Theory at the end of 19 <sup>th</sup> century which is now called the Crisp Set theory after the introduction of fuzzy sets.
 Sanders Peirce	(1839-1970)	“All that exists is continuous and such continuums govern knowledge.”
 Bertrand Russel	(1872-1970)	“Both vagueness and precision are features of language, not reality. Vagueness clearly is a matter of degree.”
 J. Lukasiewicz	(1878-1955)	Jan Lukasiewicz proposed a formal method of vagueness, where 1 stood for TRUE, 0 stood for FALSE and ½ stood for possible.



 Albert Einstein	(1879-1955)	“So far as the laws of mathematics refer to reality, they are not certain. And so far as they are certain, they do not refer to reality”
 L. A. Zadeh [50]	1965 1973	“The closer one looks at a real world problem, the fuzzier becomes its solution” Wrote a seminal paper on concept of fuzzy sets. Wrote a paper about fuzzy algorithms and showed how to apply fuzzy.
M. Sugeno	1974	Gave the concept of fuzzy measure.
 E. Mamdani	1974	Ebrahim Mamdani started applying fuzzy control to steam engine control [20].
L. A. Zadeh		Gave the concept of Type-2 fuzzy sets.
Sweden	1980	Control of Cement Kiln plant through fuzzy control.
 Japan	1986	The Sendai Fuzzy logic subway [39] system first proposed in 1978, was developed by Hitachi Ltd.
 Yamaichi Securities	1988	Yamaichi Securities (Japan) developed and implemented Fuzzy based stock trading expert system.
	1989 1989	First fuzzy logic air conditioner was developed. First fuzzy auto focus camera developed by Canon.
	1990	First fuzzy logic washing machine was developed by Matsushita [121].
	1993-1994 1994	Too many commercial applications. Japanese companies sold \$34 billion worth of


		consumer products based on fuzzy logic.
	1992-2002	Research on neuro fuzzy techniques gained momentum.
Mendel [45] and Karnik	1999	Started developing tools for type-2 based fuzzy logic set theory.
L. A. Zadeh	2000	Redirects researchers towards computing with words.
	2001	Implementation of automatic gear shift using fuzzy logic by Mitsubishi [38].

Table C 1: Brief history of fuzzy.

# Appendix D

Form PhD-4  
DOCTORAL  
PROGRAMME  
OF STUDY  
(Must be type  
written)

National University of Sciences & Technology, Rawalpindi

DOCTORAL THESIS WORK

We hereby recommend that the dissertation prepared under our supervision

by Mr. Suhail Ashraf Regn No 2002-NUST-PhD-20

Entitled: INVESTIGATIONS INTO ITERATIVE LEARNING IN FUZZY CONTROL SYSTEMS

be accepted as fulfilling in part of Doctor of Philosophy Degree.

## THESIS EVALUATION COMMITTEE

GEC Member 1: \_\_\_\_\_ Signature : \_\_\_\_\_

GEC Member 2: \_\_\_\_\_ Signature : \_\_\_\_\_

GEC Member (External) 3: \_\_\_\_\_ Signature : \_\_\_\_\_

Supervisor: Dr. Ejaz Muhammad Signature : \_\_\_\_\_

Co-Supervisor (if appointed): Nil Signature : \_\_\_\_\_

External Evaluator 1: Dr. Amin Al-Habaibeh Signature :   
(Foreign Expert) (UK) 1017108

External Evaluator 2: Dr. Reza Derakhshani Signature : \_\_\_\_\_  
(USA)

External Evaluator 3: Dr. Saeed Ur Rehman Signature : \_\_\_\_\_

## APPROVED

Dated: \_\_\_\_\_

\_\_\_\_\_  
Head of the Department

## COUNTERSIGNED

Dated: \_\_\_\_\_

\_\_\_\_\_  
Dean/Commandant/Principal/DG

### Distribution:

- 1 x copy each to Registrar, D(R&D), D(E&A) at HQ NUST and HoD, Supervisor, Co-Supervisor (if appointed), in student's dossier and each member of GEC.