

UNIVERSITY OF CALGARY

Experimental Investigation of Noncausal Iterative Learning Control

by

Ming Xia

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF MECHANICAL AND MANUFACTURING ENGINEERING

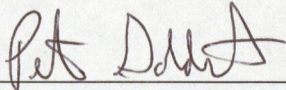
CALGARY, ALBERTA

August, 2004

© Ming Xia 2004

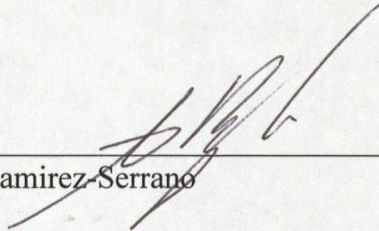
UNIVERSITY OF CALGARY
FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Experimental Investigation of Noncausal Iterative Learning Control" submitted by Ming Xia in partial fulfilment of the requirements of the degree of Master of Science.



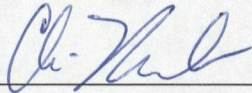
Dr. Peter B. Goldsmith, Supervisor

Department of Mechanical and Manufacturing Engineering



Dr. Alejandro Ramirez-Serrano

Department of Mechanical and Manufacturing Engineering



Dr. Chris Macnab

Department of Electrical and Computer Engineering

August 16, 2004
Date

Abstract

In many industrial robot applications, the robot is programmed to do the same task over and over. Iterative Learning Control (ILC) uses tracking errors from previous trials to correct the control input, thereby reducing tracking errors caused by plant uncertainty. Though many ILC algorithms in the literature process the previous error using causal operators, it was recently proved that the performance of causal ILC is fundamentally limited to that of conventional feedback control (without iterations). It was also proved that noncausal ILC improves on both feedback control and causal ILC. In this thesis, we validate these theoretical results through simulations and experiments. We show that, unlike causal ILC, noncausal ILC can converge to zero error even if the plant has a relative degree greater than one. Practical implementation issues, such as unsteady initial conditions and the truncation of signals in the time domain, are also addressed.

Acknowledgements

I would like to thank Dr. Goldsmith for his guidance, patience, financial support and understanding throughout the course of my study. I would also like to thank my family for their understanding, support and encouragement.

I appreciate the help from Mr. Zheng Wang for his knowledgeable suggestions and opinions. I also appreciate Mr. Chris Regier who helped me to proofread this thesis and provided me many good suggestions.

Dedication

To My Family

Table of Contents

Approval Page	ii
Abstract.....	iii
Acknowledgements	iv
Dedication	v
Table of Contents	vi
List of Tables	viii
List of Figures and Illustrations	ix
List of Symbols	xii
Chapter 1	1
Introduction.....	1
1.1 Problem description	1
1.2 Principle of Iterative Learning Control	2
1.3 Literature Review.....	3
1.3.1 ILC algorithms	3
1.3.2 Algorithms for some specific systems	6
1.3.3 ILC in time domain and frequency domain	7
1.3.4 Convergence and Robust Analysis of ILC.....	7
1.3.5 Comparison with some similar control paradigms	8
1.3.6 Equivalent Feedback Control and Noncausal ILC.....	9
1.4 Contributions.....	9
1.5 Thesis Organization	10
Chapter 2	11
Experiment Setup.....	11
2.1 The Composition of Experiment Platform.....	11
2.2 UPM.....	12
2.3 Data Acquisition Board.....	12
2.4 DC Motor	13
2.5 WinCon application software	15
2.5.1 WinCon Server.....	15
2.5.2 WinCon Client	16
2.6 Simulink Model for the Control of a DC Motor	16
2.7 Graphical User Interface for the ILC experiment	17
Chapter 3	19
Causal ILC and Equivalent Feedback Control.....	19
3.1 Fixed Point.....	19

3.2	Convergence Condition	21
3.3	Conditions on P for Convergence to $e_{\infty} = 0$	24
3.4	Causal Signals Convolution.....	25
3.5	Equivalent Feedback Control.....	26
3.6	Equivalent Feedback Control in the Case of $F = 1$	28
3.7	Simulations and Experiments	28
3.7.1	Causal ILC for a Nonminimum Phase Process.....	28
3.7.2	Zero Ultimate Error.....	30
3.7.3	Nonzero Ultimate Error	32
3.8	Summary of Causal ILC Results.....	37
Chapter 4	39
Noncausal ILC.....	39
4.1	Convolution of Noncausal Signals.....	39
4.2	A Noncausal Symmetric Low-Pass Filter.....	40
4.3	Noncausal ILC on a NMP Plant.....	43
4.4	Noncausal ILC on a DC Motor.....	44
4.5	ILC on a Higher Order System—Ball and Beam	48
4.5.1	Mathematical Model of Ball and Beam System	48
4.5.2	Stabilization of the Plant.....	50
4.5.3	Closed-loop Transfer Function	54
4.5.4	Setting Initial Point of the Ball in Iteration.....	54
4.5.5	Causal ILC on Ball and Beam System.....	56
4.5.6	Equivalent Feedback Control on Ball and Beam System	59
4.5.7	Noncausal ILC on the Ball and Beam System.....	61
4.5.8	Noncausal ILC Reduces Phase Delay	63
4.6	Conclusions About Noncausal ILC	65
Chapter 5	68
Robustness of Noncausal ILC for LTI Systems	68
5.1	Robust Performance Condition of Noncausal ILC	68
5.2	Case Study of the Robustness of Noncausal ILC	71
5.2.1	Case 1: Uncertain Feedback Gain.....	72
5.2.2	Case 2: Uncertain Parameter in Plant Model.....	78
5.2.3	Case 3: Multiplication of Plant by Low-pass Filter	83
5.3	Summary	86
Chapter 6	88
Summary and Conclusions	88
6.1	Summary.....	88
6.2	Conclusions.....	89
6.3	Recommendations for Future Work.....	91
Bibliography	92
Appendix A.....	98

List of Tables

Table 3.1: Verification Results of Lemma 2 and Equivalent Feedback Control Theory .	37
Table 4.1: Simulation and Experiment Results Comparison	65
Table 5.1: Verification Results of Lemma 3 and Lemma 4.....	86

List of Figures and Illustrations

Figure 2.1: Composition of Experiment Platform [40].....	11
Figure 2.2: Experiment Working Diagram	12
Figure 2.3: Data Acquisition Terminal Board Diagram [40].....	13
Figure 2.4: DC Motor Working Principle [40]	14
Figure 2.5: The Composition of the DC Motor [40].....	14
Figure 2.6: WinCon Client Diagram [43]	16
Figure 2.7: Simulink Control Model of a DC Motor	17
Figure 2.8: Graphic User Interface	18
Figure 3.1: Divergent Situation [39].....	25
Figure 3.2: Three Kinds of Signals.....	25
Figure 3.3: The Convolution Operation of Causal Signals.....	26
Figure 3.4: NMP Process with Causal Input.....	29
Figure 3.5: Causal ILC with Zero Ultimate Error.....	30
Figure 3.6: Equivalent Feedback Control with Zero Ultimate Error	31
Figure 3.7: Causal ILC, on Simulated Motor	33
Figure 3.8: Causal ILC, on Real Motor	33
Figure 3.9: Causal ILC with $F \neq 1$, on Simulated Motor.....	34
Figure 3.10: Causal ILC with $F \neq 1$, on Real Motor	35
Figure 3.12: Equivalent Feedback Control with $F \neq 1$, on Real Motor.....	36
Figure 4.1: The Convolution Operation of Noncausal Signals.....	40
Figure 4.2: Filter L in the Time Domain.....	42

Figure 4.3: Filter L in the Frequency Domain	42
Figure 4.4: A NMP Process with Noncausal Input.....	44
Figure 4.5: Noncausal ILC on Simulated Motor	47
Figure 4.6: Noncausal ILC on Real Motor	47
Figure 4.7: Ball and Beam System [40].....	48
Figure 4.8: Configuration of Ball and Beam System [41].....	49
Figure 4.9: Ball and Beam Simulink Control Scheme [43].....	51
Figure 4.10: Setting Initial Point of Ball at the 20 th trial	55
Figure 4.11: Influence of unsteady Initial Point to ILC System Performance at the 20 th trial	56
Figure 4.12: Causal ILC on Simulated Ball and Beam System with $F = 1$	57
Figure 4.13: Causal ILC on Real Ball and Beam System with $F = 1$	57
Figure 4.14: Causal ILC on Simulated on Ball and Beam System with $F \neq 1$	58
Figure 4.15: Causal ILC on Real Ball and Beam System with $F \neq 1$	59
Figure 4.16: Equivalent Feedback Control on Ball and Beam	60
Figure 4.17: Equivalent Feedback Control on Simulated Ball and Beam System	60
Figure 4.18: Equivalent Feedback Control on Real Ball and Beam System	61
Figure 4.19: Noncausal ILC on Simulated Ball and Beam System.....	62
Figure 4.20: Noncausal ILC on Real Ball and Beam System.....	62
Figure 4.21: Tracking a High Frequency Signal with Feedback Control.....	63
Figure 4.22: Bode Plot of Process Model	64
Figure 4.23: Tracking a Higher Frequency Signal with Noncausal ILC on Simulated Ball and Beam	64

Figure 4.24: Tracking a Higher Frequency Signal with Noncausal ILC on Real Ball and Beam	65
Figure 5.1: Plant.....	71
Figure 5.2: Bode Plot of ΔW_2 with $F = 1$ and $k = 0.3$	74
Figure 5.3: Simulation of Case 1 with $F = 1$ and $k = 0.3$	74
Figure 5.4: Bode Plot of ΔW_2 with $F = 1$ and $k = 10$	75
Figure 5.5: Simulation of Case 1 with $F = 1$ and $k = 10$	75
Figure 5.6: Bode Plot of $(\Delta W_2)^{-1}$ and L with $F = L$ and $k = 3$	77
Figure 5.7: Simulation of Case 1 with $F = L$ and $k = 3$	77
Figure 5.8: Bode Plot of ΔW_2 with $F = 1$ and $\alpha = 20$	79
Figure 5.9: Simulation of Case 2 with $F = 1$ and $\alpha = 20$	80
Figure 5.10: Bode Plot of ΔW_2 with $F = 1$ and $\alpha = 0.5$	81
Figure 5.11: Simulation of Case 2 with $F = 1$ and $\alpha = 0.5$	81
Figure 5.12: Bode Plot of $(\Delta W_2)^{-1}$ and L with $F = L$ and $\alpha = 0.5$	82
Figure 5.13: Simulation of Case 2 with $F = L$ and $\alpha = 0.5$	83
Figure 5.14: Bode Plot of ΔW_2 with $F = 1$ and $\omega_n = 5$	85
Figure 5.15: Simulation of Case 3 with $F = 1$ and $\omega_n = 5$	85

List of Symbols

A, B, C	State space system matrices
A_{cl}	Closed-loop state space system matrices
A_T	Toeplitz matrix
C, D, E, F, L	ILC operators
G	An open-loop system
H	Convergence operator
I	Unit matrix
J	Inertia of ball
J_l	Load of DC motor
J_m	Shaft inertia of DC motor
K	Control gain
K_p, K_d, K_{bp}, K_{bd}	Control gain
M, R	A transfer function
P	A plant
P_0	Nominal model of a plant
P_1	A low pass-filter
Q	A polynomial of ω
R	Real number set
S	Sensitivity
T_m	Torque
U	Input in frequency domain
U^*	Optimal input
V_m	Input voltage
W_c	A matrix
W_2	Uncertainty weight

Y	Output in frequency domain
Φ, Ψ, Γ	Matrices
a	A vector
d	Diameter of ball
e	Error
e_0	Initial error
e_i	Error at i-th trial
e_{i-1}	Error at (i-1)-th trial
e_∞	Ultimate error
f_L	A function
g	Acceleration of gravity
$h(t)$	Unit step signal
i	Trial number
j	Imaginary variable
k	A scalar
k_0	A scalar
m	Mass of ball
r	A vector
s	S domain variable
t	Time
t_0	Initial time
t_f	Final time
u	ILC input
u_0	Initial ILC input
u_i	ILC input at i-th trial
u_{i-1}	ILC input at (i-1)-th trial
u_∞	Ultimate ILC input
v	Total input

w	Feedback control input
x	Displacement of ball on the beam
x_d	Desired displacement of ball on the beam
x_i	State space vector at i-th trial
y	Output
y_d	Desired output
y_i	Output at i-th trial
y_{i-1}	Output at (i-1)-th trial
y_∞	Ultimate output
z_0	Representing $e_0 - e_\infty$
z_i	Representing $e_i - e_\infty$
ω	Angular frequency
ω_0	Cut-off frequency
τ	Time constant, or input time
α	A scalar or the angle of beam
φ	Rotating angle of ball
θ	The angle of DC motor's gear
θ_d	Desired angle of DC motor's gear
Δ	Unknown transfer function

Chapter 1

Introduction

1.1 Problem description

In control system design, system performance is always one of the most important factors to be considered. The performance design problem is defined as to force the output response of a dynamical system to follow a desired trajectory as close as possible. Although many control techniques have been developed so far, at times they are still not good enough for certain systems to achieve desired performance requirements. This may be due to the presence of unmodelled dynamics or parametric uncertainties that are exhibited during actual system operation or due to unsuitable design techniques for a particular class of systems.

Iterative Learning Control (ILC) was proposed in 1978 to solve these kinds of problems for some specific systems [3]. Often in industry there are certain processes, machines, equipment, or systems that execute the same trajectory motion or operation over and over. ILC aims to take advantage of this repetitive nature of processes to improve the performance accuracy of the system by learning from its previous performances. Since most industrial robot manipulators are required to perform the same movement many times and the nonlinearity of robot manipulators makes it difficult to achieve sufficiently small tracking error using conventional control methods, ILC is widely studied in robotics research.

1.2 Principle of Iterative Learning Control

The basic idea of ILC is to take advantage of the repetitive nature of the process to improve tracking accuracy. The approach is illustrated in Figure 1.1.

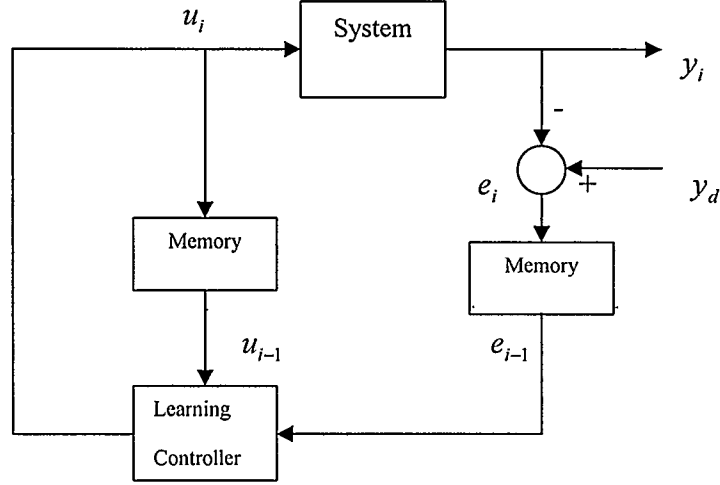


Figure 1.1: Iterative Learning Control Scheme

The subscript i indicates the trial or repetition number. The scheme operates as follows: during the i -th trial, an input $u_i(t)$ is applied to the system, producing the output $y_i(t)$, and error $e_i(t) = y_d(t) - y_i(t)$, where y_d is a desired trajectory. Then $u_i(t)$ and $e_i(t)$ are saved in memory and processed off-line by the ILC algorithm to compute an updated input signal $u_i(t)$. The updated input signal $u_i(t)$ will be applied to the system in the next trial and should produce a smaller error than the previous input.

This can be stated formally as follows. Suppose we are given a stable system

$$y_i = Pu_i, \quad (1.1)$$

where P is a linear time-invariant operator, and y_i is reset to zero at the beginning of each trial. It is assumed that only an approximate model of P is available. Then, the problem of ILC is to find operators F and D in the control update algorithm

$$u_i = Fu_{i-1} + De_{i-1} \quad (1.2)$$

such that e_i and u_i converge to fixed points e_∞ and u_∞ , respectively, with e_∞ as small as possible (in a suitable norm). The convergence of the ILC algorithm should not depend on the desired response $y_d(t)$. If P is not already open loop stable, then it must be stabilized using feedback [2].

Although each trial occurs on a finite interval $[t_0, t_f]$, we will analyze the system for the case when the final time t_f approaches infinity. This will allow convergence analysis in the frequency domain [1].

1.3 Literature Review

Iterative Learning Control was originally proposed by Uchiyama [3] in 1978. Later the research work on ILC by a group led by Sugurn Arimoto [4] drew much attention in control and robotics fields, particularly through the middle to late 1980's. Since then, ILC has been a very active research area and thousands of research papers on ILC have been published. In order to get a brief overview of ILC research achievement, we may investigate the following aspects of the literature.

1.3.1 ILC algorithms

According to the way the error is processed, we may classify the algorithms as first order, second order and higher order ILC.

1.3.1.1 First order ILC

If, in the ILC upgrade law, only the previous trial's error is used, the algorithm is first order. At the beginning stage of ILC research, just one previous error is applied in the algorithm. The typical one is the scheme proposed by Arimoto et al [4]. Suppose the dynamic system is given as

$$\dot{x} = Ax_i + Bu_i, \quad (1.3)$$

$$e_i = Cx_i, \quad (1.4)$$

and $\dot{e} = CAx_i + CBu_i.$ (1.5)

Arimoto et al applied the derivative of the previous trial's error in the algorithm (D-type) as follows:

$$u_i = u_{i-1} + \Gamma \dot{e}_{i-1}, \quad (1.6)$$

where $e_{i-1} = y_d - y_{i-1}$, u_{i-1} is previous input, u_i is updated input, and Γ is a given $r \times r$ matrix. If a LTI (Linear Time-Invariant) system is defined in the interval $[t_0, t_f]$ and some initial requirements are satisfied, a sufficient convergence condition can be expressed as:

$$CB > 0, \quad (1.7)$$

and $\|I - CB\Gamma\|_i < 1.$ (1.8)

Later P-type and PID-type ILC algorithms were further proposed respectively by S.S.Saab [5] and Arimoto [6]. For P-type the input update law is:

$$u_{i+1} = u_i + Ae_i, \quad (1.9)$$

while a PID-type law is designed as:

$$u_{i+1} = u_i + \Phi e_i + \Gamma \dot{e}_i + \Psi \int e_i dt . \quad (1.10)$$

Although there are some differences between the above algorithms in processing the previous error of the system, the common point is that just one previous step error is considered.

1.3.1.2 Second Order ILC

By introducing “Current Cycle Feedback” control into ILC update law, the algorithm becomes second order. The idea of combining ILC with conventional feedback control appeared in [7]. The general algorithm ([2], [8]) can be written as:

$$u_i = Fu_{i-1} + Ce_i + De_{i-1}, \quad (1.11)$$

where e_i is current cycle error and e_{i-1} is previous error, u_{i-1} is previous input, F, C and D are proper operators.

An obvious advantage of (1.11) is that it allows stabilization of the plant and it is more general than the first order algorithm.

1.3.1.3 Higher order ILC

Higher order ILC was first considered in [10], [11]. The general algorithm includes current cycle feedback, and errors from several previous steps:

$$u_i = Fu_{i-1} + Ce_i + De_{i-1} + Ee_{i-2} + \dots . \quad (1.12)$$

This update law can provide more information about the past performance of the system, and will be an important area for further research.

1.3.2 Algorithms for some specific systems

1.3.2.1 ILC for LTI systems

Consider the LTI ILC algorithm (1.11), and errors:

$$e_i = y_d - Pu_i, \quad (1.13)$$

$$e_{i-1} = y_d - Pu_{i-1}. \quad (1.14)$$

Substituting (1.11) into (1.13) gives

$$e_i = y_d - P(Fu_{i-1} + Ce_i + De_{i-1}). \quad (1.15)$$

Multiplying (1.14) by F and subtracting the result from (1.15) gives

$$(1 + PC)e_i - (F - DP)e_{i-1} = (1 - F)y_d. \quad (1.16)$$

The fixed point of the error e_∞ is obtained by setting $e_i = e_{i-1} = e_\infty$ in (1.16), which yields

$$[1 - F + (C + D)P]e_\infty = (1 - F)y_d. \quad (1.17)$$

Rearranging (1.17), we have

$$e_\infty = Ry_d, \quad (1.18)$$

where
$$R = \frac{1 - F}{1 - F + (C + D)P}. \quad (1.19)$$

A very important property of this algorithm is that:

- If $F = 1$, the fixed point of the error is zero.
- If $F \neq 1$, the fixed point of the error is nonzero, and the nonzero error can be expressed as (1.18) [12].

1.3.2.2 Nonlinear operator and nonlinear systems in ILC

A discussion on the use of Artificial Neural Networks (ANN) in ILC is given in [1]. This method can be viewed as a kind of nonlinear black-box identification approach. In this situation not only the control signal but also the ILC algorithm changes over the iterations. A specific class of nonlinear systems have been considered by Choi [13] and a more thorough discussion of this kind of ILC approach can be found in [14].

1.3.2.3 ILC for discrete-time systems

Since all practical ILC implementations will result in a discrete-time algorithm, it is necessary to investigate the applications of ILC to discrete-time systems. The discrete-time linear case is discussed in [1], time-varying systems are considered in [15], nonlinear systems are considered in [16], and non-minimum phase systems have been discussed in [17].

1.3.3 ILC in time domain and frequency domain

Both time domain and frequency domain are good tools for ILC research. Since the time domain provides the convenience for practical implementation, most ILC algorithms are discussed in the time domain. But the frequency domain can provide more design freedom and useful physical insights than in the time domain. There are many publications related to frequency domain based ILC. The typical papers are [18] and [19].

1.3.4 Convergence and Robust Analysis of ILC

Convergence and robustness are some important factors that an ILC algorithm designer has to consider. Although ILC researchers normally provide convergence

analysis for their algorithms, it is still a topic worthy of discussing separately. This work has done by [20], [21] for linear systems and [22], [23] for nonlinear systems.

Robustness analysis was discussed in [24], [25] for linear systems and in [26], [27] for nonlinear systems.

1.3.5 Comparison with some similar control paradigms

ILC is similar to some other control paradigms. To clearly show the difference between ILC and other control techniques, some comparisons are necessary.

1.3.5.1 Feedback control

Compared to feedback control, ILC is essentially a feed forward control that processes previous errors off-line to get an updated input. ILC may include current cycle feedback to make the system stable or provide a minimum performance.

1.3.5.2 Optimal control

Most optimal controls are based on a model of a system, while ILC does not require the accurate information of the system. Both methods are looking for an optimal input U^* . The difference is that optimal control gets the optimal input by processing the current cycle error in real time. ILC makes use of not only current cycle error but also past system behaviour. ILC gets U^* by off-time processing.

1.3.5.3 Repetitive Control

ILC and Repetitive Control (RC) are similar control methods, but differences exist.

- ILC works on a finite time interval, while RC works continuously.

- ILC starts over at the same initial condition, while RC does not require the same initial condition.

The common point is that both of them work repetitively.

1.3.6 Equivalent Feedback Control and Noncausal ILC

The latest concern on ILC is the topic of noncausal ILC. All of the algorithms discussed above are normally implemented using causal operators. It has recently been found that if an ILC algorithm is causal, an equivalent feedback control exists [8], [28]. This implies that causal ILC does not improve on conventional feedback control because the latter does not need iteration to achieve the same fixed point. The purpose of ILC is to reduce the performance error that is difficult to overcome using feedback and other control methods. The ideal goal is that the fixed point is equal to zero. But for causal ILC, this goal can only be achieved for the simplest processes: systems of minimum phase and with relative degree less than or equal to 1.

Since causal ILC is limited to feedback control and the simplest processes, noncausal ILC has to be explored to break the limitations of causal ILC.

A noncausal ILC algorithm is proposed in [29]. The theoretical proof shows that noncausal ILC can solve the limitation problem of causal ILC and thus can improve feedback control. But so far the research is in the theoretical analysis stage and some essential experiments on real systems are needed to validate the theoretical result.

1.4 Contributions

The purpose of this thesis is to experimentally verify the theoretical hypotheses proposed in [8, 29]. The contributions of this thesis are:

- 1) Experimental validation of the equivalent feedback control theorem for causal ILC.
- 2) Experimental examination of noncausal ILC's improvement over causal ILC.
- 3) Experimental validation of robust ILC design for LTI systems.
- 4) An experimental platform setup and Simulink model design.
- 5) Demonstration of the benefit of noncausal operators using Matlab simulations.
- 6) Successful application of noncausal ILC on a higher order system (Ball and Beam system).
- 7) Successfully using feedback control for setting ILC initial conditions in the Ball and Beam experiment.

1.5 Thesis Organization

In Chapter 2, the experiment platform is introduced. Chapter 3 discusses causal ILC and equivalent feedback control. Simulation results of causal ILC and equivalent feedback control are compared and real system results are also obtained to verify the theory. In Chapter 4, noncausal ILC is discussed and experiments on a DC motor and Ball and Beam module are implemented. In Chapter 5, a robust ILC design for LTI systems is introduced and simulations are done to verify the robustness of the algorithm. Chapter 6 concludes the thesis.

Chapter 2

Experiment Setup

2.1 The Composition of Experiment Platform

The experiment platform is composed of a computer with Matlab and WinCon software installed, UPM (Universal Power-supply Module), data acquisition board and a physical plant such as a DC motor shown in Figure 2.1 or a Ball Beam system. A Graphical User Interface (GUI) is created with Matlab to communicate between the Simulink model and an M-file. The purpose of this experiment is to control the motor's shaft position or the position of the ball on a beam by applying ILC algorithms.

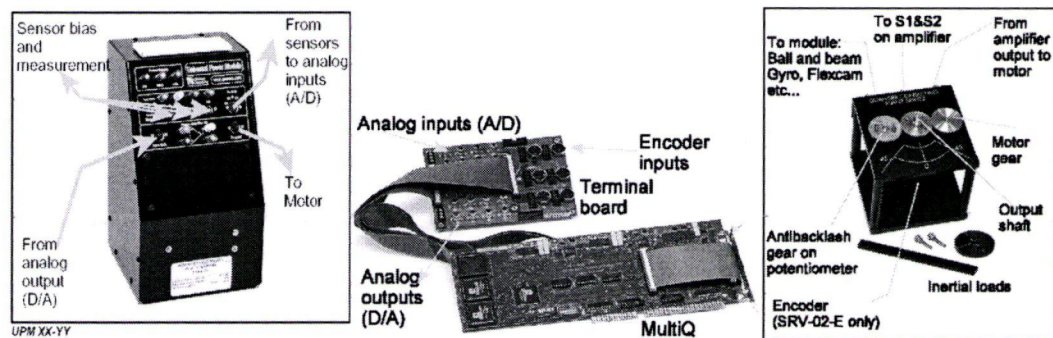


Figure 2.1: Composition of Experiment Platform [40]

The control diagram of the system is described in Figure 2.2. A Simulink control model is built in the computer. With the WinCon application software and the hardware module a real time control can be implemented. When the control model begins to operate, the digital control signal is produced and sent to MultiQ-3 board. Then the D/A converter on the board converts the digital signal to a voltage signal, which is supplied by

UPM to drive the motor. The output of the plant is measured by a sensor and fed back to the controller in the computer via the MutiQ-3 board so that the error is reduced and plant is working as desired.

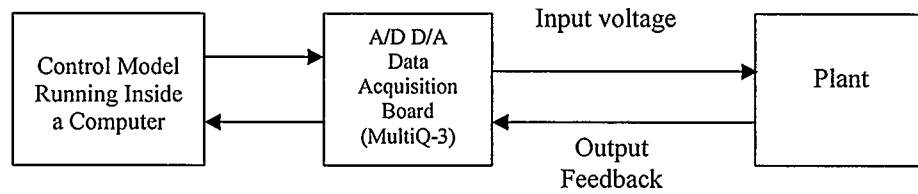


Figure 2.2: Experiment Working Diagram

2.2 UPM

The power module (UPM2405) consists of a regulated dual output DC power supply set at ± 12 Volts (Vs) and a built-in linear power operational amplifier. The UPM is used to drive the DC motor. The maximum current available is 1 Ampere without the amplifier cable. The functional ports of UPM are located on the panel (See Figure 2.1). The D/A and A/D ports are connected to MultiQ-3 board. The sensor's ports and voltage output port are connected to DC motor. In the Ball Beam experiment, a potentiometer sensor on the Ball Beam module is connected to a sensor's port as well. [40]

2.3 Data Acquisition Board

The MultiQ-3 is a general purpose data acquisition and control board, which has 8 single analog inputs, 8 analog outputs, and 16 bits of digital input, 16 bits of digital output, 3 programmable timers and up to 8 encoder inputs decoded in quadrature. Interrupts can be generated by any of the three clocks, one digital input line or the end of conversion from the A/D.

The system is accessed through a PC bus and is addressable via 16 consecutive

memory mapped locations, which are selected through a DIP switch located on the board. A diagram of the MultiQ-3 is shown in Figure 2.3. In this experiment, “Analog Inputs” are connected to the “A/D” port of the UPM, “Analog Outputs” go to “D/A” port of the UPM, and one of “Encoder Inputs” connect to the DC motor. [40]

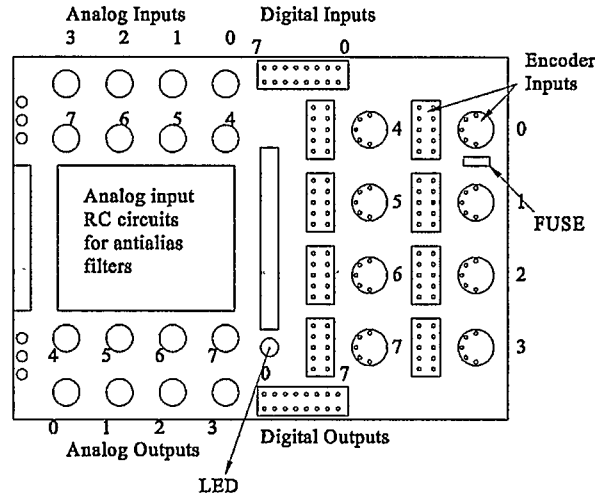


Figure 2.3: Data Acquisition Terminal Board Diagram [40]

2.4 DC Motor

SRV-02-E type motor shown in Figure 2.1 consists of a DC servomotor with built-in gearbox drives, a potentiometer, an optical encoder, and an independent output shaft. Its block diagram and configuration are shown in Figure 2.4 and Figure 2.5, respectively.

The input voltage V_{in} produces an electrical current. This electrical current generates a torque T_m , which turns the motor shaft J_m and load J_l via a gear train K_g . The angular position of the output gear can be measured with a potentiometer or an optical encoder sensor built into the motor module. This motor can be configured with a

high gear ratio. The internal gearbox gear ratio is 14:1 and the external gear ratio is set to 5:1 in this experiment, so the total gear ratio is 70:1. [40]

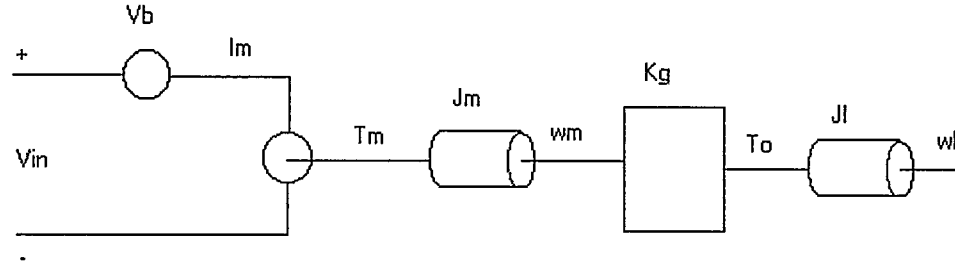


Figure 2.4: DC Motor Working Principle [40]

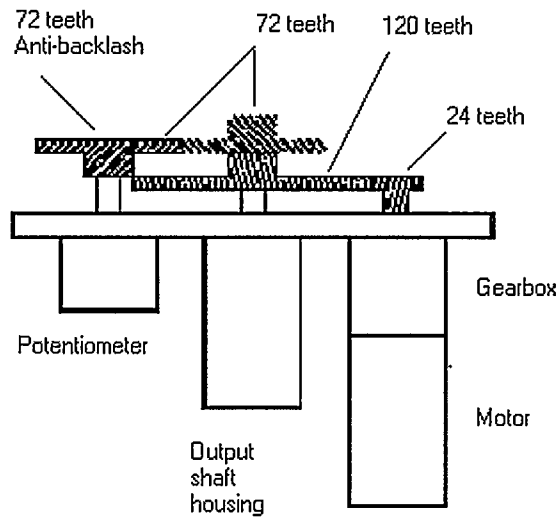


Figure 2.5: The Composition of the DC Motor [40]

The theoretical DC motor's mathematical model is $\frac{\theta(s)}{V_{in}(s)} = \frac{K}{s(\tau s + 1)}$, where K and

τ are determined by the motor's parameters and affected by the gear ratio. By

experiment K was identified to be 1.5, and τ was found to be 0.03, so the transfer function of the motor is

$$\frac{\theta(s)}{V_{in}(s)} = \frac{1.5}{s(0.03s + 1)} . \quad (2.1)$$

2.5 WinCon application software

WinCon is a real time Windows 95/98/Me/NT/2000 application that runs Simulink-generated code using Real-Time Workshop on a PC. It consists of WinCon Client and WinCon Server.

2.5.1 WinCon Server

WinCon Server performs the following functions:

- ❑ Converting a Simulink diagram to PC executable code by using real-time workshop;
- ❑ Compiling and linking the codes by using Visual C++;
- ❑ Downloading the code to run on a WinCon client;
- ❑ Performing start and stop functions for a client;
- ❑ Maintaining communications with a client;
- ❑ Maintaining communications with Simulink to catch real-time changes in parameters of the block diagram;
- ❑ Plotting the data in real time;
- ❑ Saving the data to disk after stopping the running of a client.

2.5.2 WinCon Client

WinCon client runs the code generated from the Simulink diagram. It works with WinCon Server and has the following functions:

- ❑ Receiving the code from a server;
- ❑ Running the code in real time;
- ❑ Maintaining communications with the server.

The most common configuration is a single PC equipped with the required software and hardware. In this configuration shown in Figure 2.6, the PC runs both the server and the client and can be used to perform real-time control, tuning and monitoring in the same location. [43]

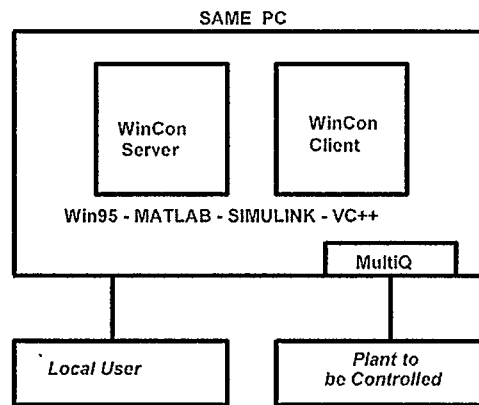


Figure 2.6: WinCon Client Diagram [43]

2.6 Simulink Model for the Control of a DC Motor

Figure 2.7 shows the ILC control scheme for a DC motor. A PID Feedback controller is applied to stabilize the system. The “ILC input Sequence” block produces a complementary sequence calculated by an ILC algorithm from the previous input and error data. The ILC control input is fed forward to adjust the performance of the plant.

The Scopes labelled “Error” and “ILC Input” display the error and ILC input signals respectively, which are loaded to the workspace after every iteration and then processed in the “ILC input Sequence” block by the ILC algorithm. [44]

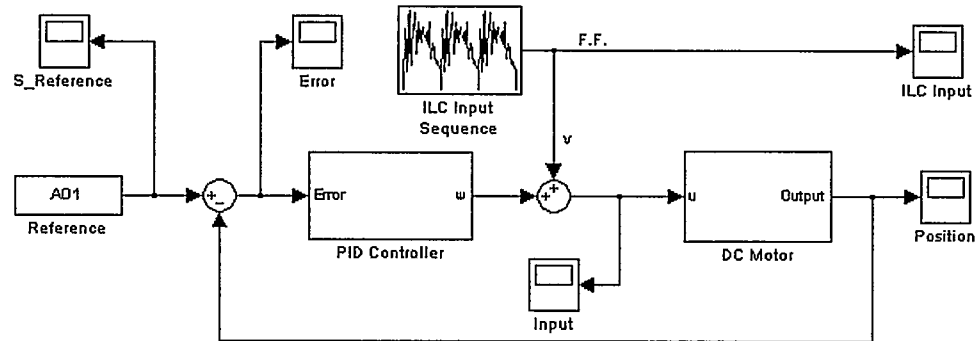


Figure 2.7: Simulink Control Model of a DC Motor

2.7 Graphical User Interface for the ILC experiment

A graphical user interface (GUI) helps to combine Simulink control model with an M-file. Some graphics objects such as windows, icons, buttons menus, and text can be created within the GUI frame. Selecting or activating an object in the interface frame usually causes an action or change to occur. The most common activation method is to use a mouse, [45].

There are several reasons to use a GUI in this experiment. In Figure 2.7, after each iteration, the previous error and input data need to be saved in the workspace and then to be processed in the “ILC Input Sequence” with the ILC algorithm. This operation requires an M-file to work with the Simulink model interactively. The GUI provides the interaction between the Simulink model and the corresponding M-file. Since this work is repetitive, the GUI makes the experiment much more convenient. With a GUI designed

for this experiment (See Figure 2.8), we can easily change some related parameters and settings of the system without opening the Simulink model.

In the GUI shown in Figure 2.8, pressing the “Initialise” button will open the Simulink model, load an executive code into the WinCon server, and open some necessary sinks. “START” is a toggle button to start or stop an operation. The section below the “Initialise” button allows one to set iteration interval and iteration times. The popup menu is used to select a real system or a mathematical transfer function of the system in order to compare the system’s performances. The “ILC Operators” section allows setting or changing ILC operators such as F, C, and D in transfer function form during the experiment to analyze the ILC influence when varying the values of these operators. The “Controller Parameters” section is for setting feedback controller’s parameters to stabilize the system. The corresponding M-file is attached in Appendix A.

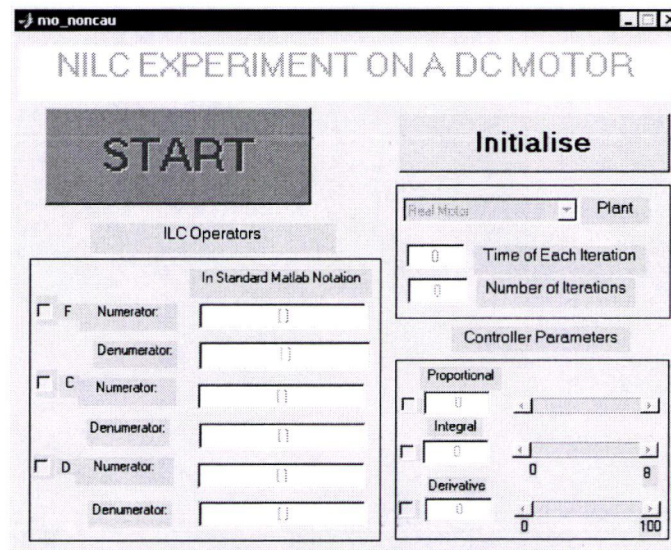


Figure 2.8: Graphic User Interface

Chapter 3

Causal ILC and Equivalent Feedback Control

This chapter introduces some general concepts of ILC such as causal signals, noncausal signals, fixed point and convergence condition, which are related to our analysis. Since most existing ILC algorithms are implemented by using causal operators, a causal signals convolution procedure is illustrated to show how causal operators work in an ILC algorithm. During the discussion, we will prove that causal ILC has some fundamental drawbacks. Equivalent Feedback Control theory proposed in [8] claims that there exists an equivalent feedback control for any causal ILC algorithm. Since this theory has not verified by practical application, some simulation and experiment are designed to investigate it further.

3.1 Fixed Point

A fixed point is an equilibrium point during iterations in an ILC system. Consider a general ILC system:

$$y_i = Pu_i, \quad (3.1)$$

where P is a stable plant and i is trial number. If an achievable reference trajectory is given as

$$y_d = Pu_d, \quad (3.2)$$

then the error in trial i is

$$e_i = y_d - y_i. \quad (3.3)$$

From (1.11), the general ILC algorithm is

$$u_i = Fu_{i-1} + Ce_i + De_{i-1}, \quad (3.4)$$

where F , C , and D are bounded causal operators.

Substituting (3.1) into (3.3) gives, for iterations i and $i-1$, respectively,

$$e_i = y_d - Pu_i, \quad (3.5)$$

$$e_{i-1} = y_d - Pu_{i-1}. \quad (3.6)$$

Substituting (3.5) and (3.6) into (3.4) gives

$$u_i = Fu_{i-1} + C(y_d - Pu_i) + D(y_d - Pu_{i-1}). \quad (3.7)$$

If $u_i = u_\infty$ and $u_{i-1} = u_\infty$, where u_∞ is bounded, the signal u_∞ is called a fixed point of control signal [1].

Set

$$u_i = u_{i-1} = u_\infty. \quad (3.8)$$

From (3.5) and (3.6), we have

$$e_i = e_{i-1} = e_\infty. \quad (3.9)$$

Substituting (3.8) and (3.9) into (3.4) gives

$$(1 - F)u_\infty = (C + D)e_\infty, \quad (3.10)$$

Rearranging (3.10) gives

$$u_\infty = Ke_\infty, \quad (3.11)$$

where

$$K = (1 - F)^{-1}(C + D). \quad (3.12)$$

Note that $F \neq 1$ in (3.12). Since

$$e_\infty = y_d - Pu_\infty, \quad (3.13)$$

Substituting (3.11) into (3.13) gives

$$e_{\infty} = (1 + PK)^{-1} y_d. \quad (3.14)$$

Then

$$u_{\infty} = K(1 + PK)^{-1} y_d, \quad (3.15)$$

and

$$y_{\infty} = PK(1 + PK)^{-1} y_d, \quad (3.16)$$

where y_{∞} is the output corresponding to the fixed point of control signal u_{∞} .

Definition: the signal $u_i = u_{i-1} = u_{\infty}$ is called a fixed point of control signal, y_{∞} or e_{∞} is called a fixed point of an ILC system [1].

Remark 1: Since u_{∞} is a bounded signal, if P is stable, y_{∞} is a bounded signal as well.

Remark 2: If a convergence condition is met, y_i will converge to y_{∞} as i increases.

3.2 Convergence Condition

Convergence analysis is performed most easily in the frequency domain. In the following, $e(j\omega)$ refers to the Fourier transform of the signal $e(t)$ (i.e. the Laplace transform evaluated at $s = j\omega$).

Definition: An ILC system is convergent if, for all bounded $e_0(j\omega)$, $\lim_{i \rightarrow \infty} e_i(j\omega) = e_{\infty}(j\omega)$ at all frequencies ω .

Substituting (3.4) into (3.5) gives

$$e_i = y_d - P(Fu_{i-1} + Ce_i + De_{i-1}). \quad (3.17)$$

Multiplying (3.6) by F and subtracting the result from (3.17) gives

$$(1 + PC)e_i - (F - DP)e_{i-1} = (1 - F)y_d. \quad (3.18)$$

Substituting (3.9) into (3.18) gives

$$[1 - F + (C + D)P]e_\infty = (1 - F)y_d. \quad (3.19)$$

Subtracting (3.19) from (3.18) gives

$$(1 + PC)(e_i - e_\infty) = (F - DP)(e_{i-1} - e_\infty), \quad (3.20)$$

which may be rearranged to give

$$e_i - e_\infty = H(e_{i-1} - e_\infty), \quad (3.21)$$

where

$$H = S(F - DP), \quad (3.22)$$

and

$$S = (1 + PC)^{-1}. \quad (3.23)$$

Finally, (3.21) implies

$$e_i - e_\infty = H^i(e_0 - e_\infty). \quad (3.24)$$

Lemma 1: *A necessary and sufficient condition for e_i to converge to e_∞ is $|H(j\omega)| < 1$ for all $\omega \in R$. [31]*

Proof:

Sufficient Condition:

Let $z_i = e_i - e_\infty$, so $z_0 = e_0 - e_\infty$. Then (3.24) becomes

$$z_i = H^i z_0. \quad (3.25)$$

Since

$$|H(j\omega)^i| = |H(j\omega)|^i, \quad (3.26)$$

$$|z_i(j\omega)| = |H^i(j\omega)z_0(j\omega)| = |H(j\omega)|^i |z_0(j\omega)|. \quad (3.27)$$

If $|H(j\omega)| < 1$, then

$$\lim_{i \rightarrow \infty} |H(j\omega)|^i = 0. \quad (3.28)$$

Thus we have

$$\lim_{i \rightarrow \infty} |z_i(j\omega)| = 0, \quad (3.29)$$

and

$$\lim_{i \rightarrow \infty} e_i = e_\infty. \quad (3.30)$$

Necessary Condition:

If the ILC system is convergent, then we have

$$\lim_{i \rightarrow \infty} e_i = e_\infty$$

and

$$\lim_{i \rightarrow \infty} |z_i(j\omega)| = 0. \quad (3.31)$$

According to (3.25),

$$\lim_{i \rightarrow \infty} |z_i(j\omega)| = \overline{\lim_{i \rightarrow \infty} |H(j\omega)|^i |z_0(j\omega)|} = 0. \quad (3.32)$$

Considering (3.26) and (3.31) gives

$$\lim_{i \rightarrow \infty} |z_i(j\omega)| = \lim_{i \rightarrow \infty} |H(j\omega)|^i |z_0(j\omega)| = 0. \quad (3.33)$$

Since z_0 is in general nonzero, (3.33) implies

$$\lim_{i \rightarrow \infty} |H(j\omega)|^i = 0. \quad (3.34)$$

So

$$|H(j\omega)| < 1.$$

Remark: According to (3.10), when choosing $F = 1$, we get $e_\infty = 0$.

3.3 Conditions on P for Convergence to $e_\infty = 0$

In order to achieve zero ultimate error ($e_\infty = 0$), we need to choose $F = 1$ in (3.10).

In this special convergence case, the properties of process P play an important role. If all zeros of a system are in the left half-plane, the system is called Minimum Phase (MP). Otherwise, the system is called Nonminimum Phase (NMP) [38]. Relative degree is the difference between the order of the denominator and the order of the numerator of a transfer function.

Lemma 2: *If a causal ILC system converges to $e_\infty = 0$, then the process P must be M.P. and its relative degree must be less than or equal to 1.* [39]

Proof: In order to achieve $e_\infty = 0$, set $F = 1$. According to (3.22),

$$H = S(1 - DP) . \quad (3.35)$$

If this system is convergent, we have $|H(j\omega)| < 1$. That is

$$|S(j\omega)(1 - D(j\omega)P(j\omega))| < 1 . \quad (3.36)$$

Since $S = 1 - SPC$, rearranging (3.36) gives

$$|1 - (C(j\omega) + D(j\omega))S(j\omega)P(j\omega)| < 1 , \quad (3.37)$$

or

$$|1 - M(j\omega)| < 1 , \quad (3.38)$$

where $M = (C + D)SP$.

If P is NMP and /or has a relative degree greater than 1, then so is M . This implies

$\text{Re}(M(j\omega)) < 0$ at some ω in complex plane, and thus

$$|1 - M(j\omega)| > 1 , \quad (3.39)$$

which conflicts with the convergence assumption. Figure 3.1 illustrates the situation.

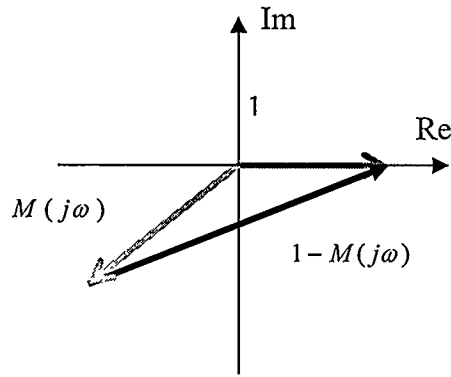


Figure 3.1: Divergent Situation [39]

3.4 Causal Signals Convolution

Definition 1: A signal u is causal if $u(t) = 0, \forall t < 0$. A signal u is anti-causal if $u(-t)$ is causal. A noncausal signal is not necessarily causal. [29] Figure 3.2 illustrates the three signals.

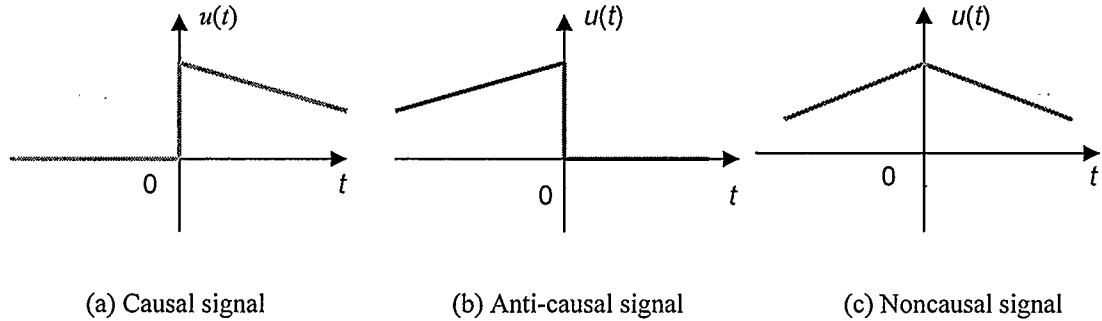


Figure 3.2: Three Kinds of Signals

Definition 2: “A system (or an operator) P that maps an input signal u to an output signal y is causal if $y(t)$ does not depend on $u(\tau)$ for all $\tau > t$. A noncausal system is not

necessarily causal and an anti-causal system is not causal. If an LTI system is causal, then every causal input produces a causal output”, [29].

For a causal LTI system, $y = Pu$, the output y is obtained according to the following formula

$$y(t) = \int_{-\infty}^{+\infty} p(t-\tau)u(\tau)d\tau, \quad (3.40)$$

where p is the impulse response of P . The convolution operation can be illustrated as shown in Figure 3.3. Since P is a causal operator, if input signal u is also a causal signal, the output y at $\tau = t$ will be the value of shaded area. [46]

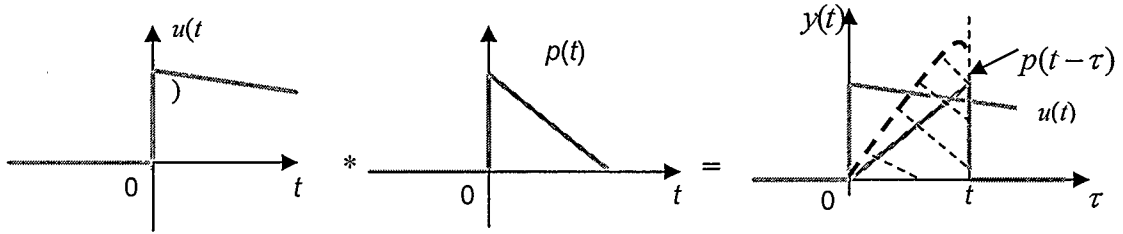


Figure 3.3: The Convolution Operation of Causal Signals

3.5 Equivalent Feedback Control

If ILC is implemented using causal operators (i.e. causal ILC), we can show that it is equivalent to feedback control (which must use causal operators). Consider a conventional feedback control system

$$e = y_d - Pu, \quad (3.41)$$

where the control is

$$u = Ke. \quad (3.42)$$

Comparing (3.42) and ILC fixed point described as (3.11), an important relationship between causal ILC and feedback control is revealed.

Theorem 1: *Suppose $(1 - F)^{-1}$ is defined, and let $K = (1 - F)^{-1}(C + D)$ as the same as (3.12), then the feedback control (3.40) applied to (3.39) gives $e = e_\infty$, [12].*

Proof: Substituting (3.40) into (3.39) gives the closed –loop system

$$e = (1 + PK)^{-1} y_d . \quad (3.43)$$

Applying (3.41) to (3.40), we have

$$u = K(1 + PK)^{-1} y_d . \quad (3.44)$$

Comparing (3.41) with (3.14), and (3.42) with (3.15) respectively, we have

$$e = e_\infty ,$$

and $u = u_\infty .$

Remark 1: The K in the equivalent feedback control depends only on the ILC operators F , C , and D , so no additional process information is required to obtain the equivalent feedback control, [12].

Remark 2: Since Theorem 1 does not exclude the case $C=0$, the equivalent feedback control K exists whether or not the ILC includes current cycle feedback C . [12]

Remark 3: The equivalent feedback achieves the fixed point of ILC even if the ILC does not converge to the fixed point, [12].

Remark 4: Since the equivalent feedback control achieves the fixed point on the infinite time interval, it also achieves it on any finite interval (i.e. a trial of finite duration), [12].

3.6 Equivalent Feedback Control in the Case of $F = 1$

In the singular case where $(1 - F)^{-1}$ does not exist, Theorem 1 is not applicable. Theorem 2 explains the relationship between the ILC and feedback control in this special case.

Theorem 2: *If causal ILC converges to zero error, then the feedback control $u = [C + k(C + D)]e$, $k \geq 0$, is internally stabilizing and gives $\lim_{k \rightarrow \infty} e = 0$, [8].*

The proof is done in [8], where the following remarks are noted:

Remark 1: Here $K = C + k(C + D)$, and K depends only on the ILC operators C and D , not on the Process P (although the size of the gain factor k required for a given $\|e\|$ may depend on P), [8].

Remark 2: Since Theorem 2 does not exclude the case $C = 0$, the equivalent feedback K exists whether or not the ILC includes current feedback C , [8].

Remark 3: The achievability assumption, $y_d = Pu_d$ for bounded u_d , allows the system to track an unbounded y_d when P is unstable, [8].

3.7 Simulations and Experiments

In order to validate the above theoretical conclusions equating causal ILC and feedback control, simulations and experiments on various processes are presented.

3.7.1 Causal ILC for a Nonminimum Phase Process

Lemma 2 shows theoretically that if P is NMP, perfect tracking of a reference y_d is not possible with a causal ILC algorithm. Consider for example the NMP system

$$Y(s) = P(s)U(s), \quad (3.45)$$

where
$$P(s) = \frac{s-1}{(s+1)^2}. \quad (3.46)$$

Suppose the reference to be tracked is

$$Y_d(s) = \frac{1}{(s+1)^2}. \quad (3.47)$$

To achieve this target, the input should be

$$U(s) = P^{-1}(s)Y_d(s) = \frac{1}{s-1}. \quad (3.48)$$

If we take the two-sided inverse Laplace transform, as defined in [30], of (3.48), we get two solutions: a causal input $u(t) = e^t h(t)$ and an anti-causal input $u(t) = -e^t h(-t)$.

When we apply the causal input to the system, the simulation result is as Figure 3.4.

Because the causal input produces an unbounded output, it cannot be accepted.

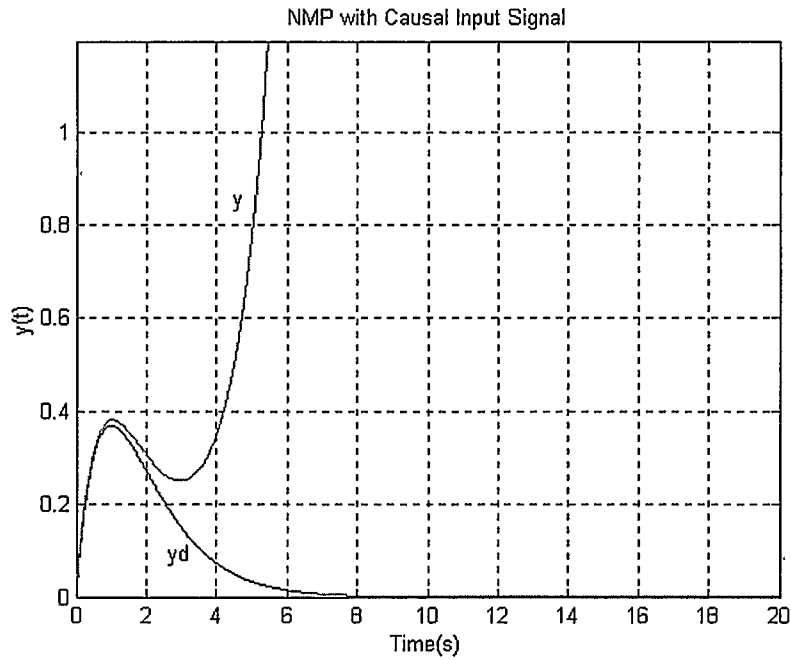


Figure 3.4: NMP Process with Causal Input

3.7.2 Zero Ultimate Error

3.7.2.1 Causal ILC Algorithm

Consider an open-loop model: $P(s) = \frac{1}{s+1}$. Since this plant is minimum phase and has relative degree of one, according to Lemma 2, zero ultimate error is possible using causal ILC. Set $C = 0$ and $D = 1$ in (3.4). To achieve zero ultimate error, we need to choose $F = 1$. If the ILC system is required to track

$$y_d = 0.5 + 0.5 * \sin(t - \pi / 2) \quad (3.49)$$

over the interval $t \in [0,3]$, simulation shows the system performance is as indicated in Figure 3.5 where the initial condition is $u_0 = 0$. Zero ultimate error is achieved for this process and the converse of Lemma 2 is validated.

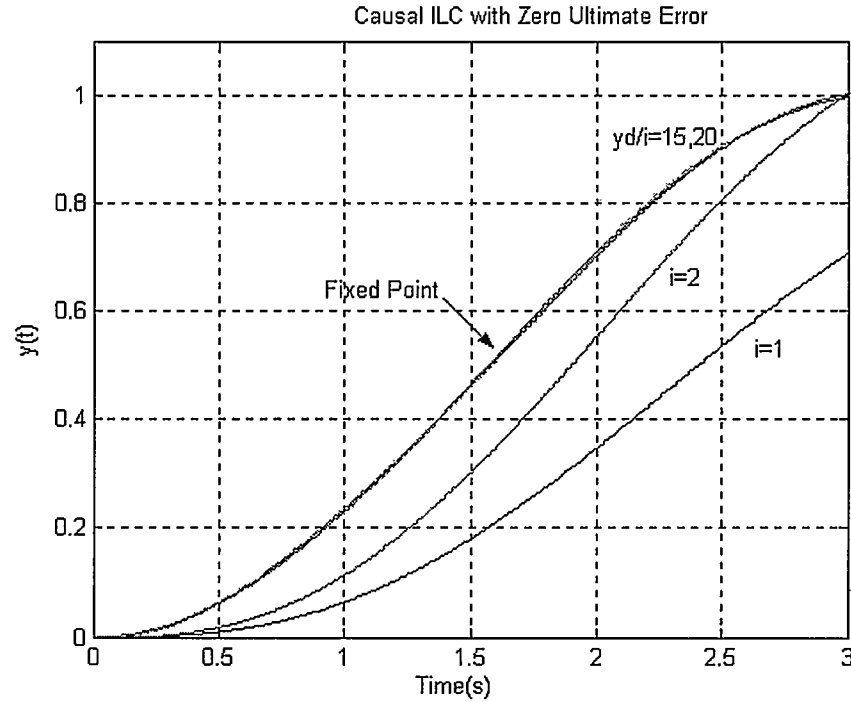


Figure 3.5: Causal ILC with Zero Ultimate Error

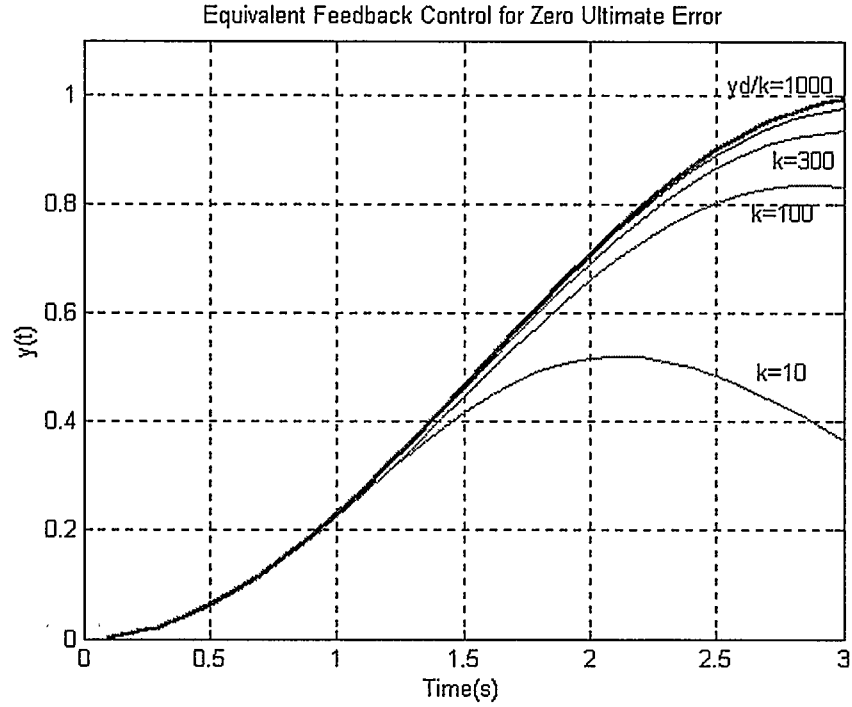


Figure 3.6: Equivalent Feedback Control with Zero Ultimate Error

3.7.2.2 Equivalent Feedback Control

Applying Theorem 2 to this example gives the result shown in Figure 3.6. The equivalent feedback control is $u = [C + k(C + D)]e$, where C and D are the same as those in the causal ILC algorithm. When we increase k , the output y is closer to y_d . Increasing k to $k = 1000$, the output y coincides with y_d . Obviously, equivalent feedback control does not need iteration and can achieve zero ultimate error at only one trial as long as the value of k is large enough in the controller formula.

3.7.3 Nonzero Ultimate Error

Consider an open loop model of a DC motor (2.1), which is $P(s) = \frac{1.5}{s(0.03s + 1)}$, with a relative degree greater than 1. The goal of this experiment is to control the motor's shaft position with an ILC algorithm (3.4) and to track a desired signal represented by y_d .

3.7.3.1 Causal ILC when $F = 1$

Suppose we choose $F = 1$ because we hope to get a zero ultimate tracking error for this process, then (3.4) becomes $u_i = u_{i-1} + Ce_i + De_{i-1}$, where we choose $C = 1$ to stabilize the system and $D = LP^{-1}$. $L = \frac{1}{(0.5s + 1)^2}$ is a filter to make D proper.

According to Lemma 2, a zero ultimate error may not be achieved since the relative degree of the process is greater than 1.

A simulation and an experiment on the motor are implemented to validate Lemma 2 at this situation. Figure 3.7 shows the simulation result, while Figure 3.8 shows the result of the experiment. From both results, we see that the system is divergent, validating Lemma 2 for the case of the plant relative degree exceeding one. The rate of divergence differs for the simulation and experiment because of the unmodeled parameters and uncertainties that exist in the real plant.

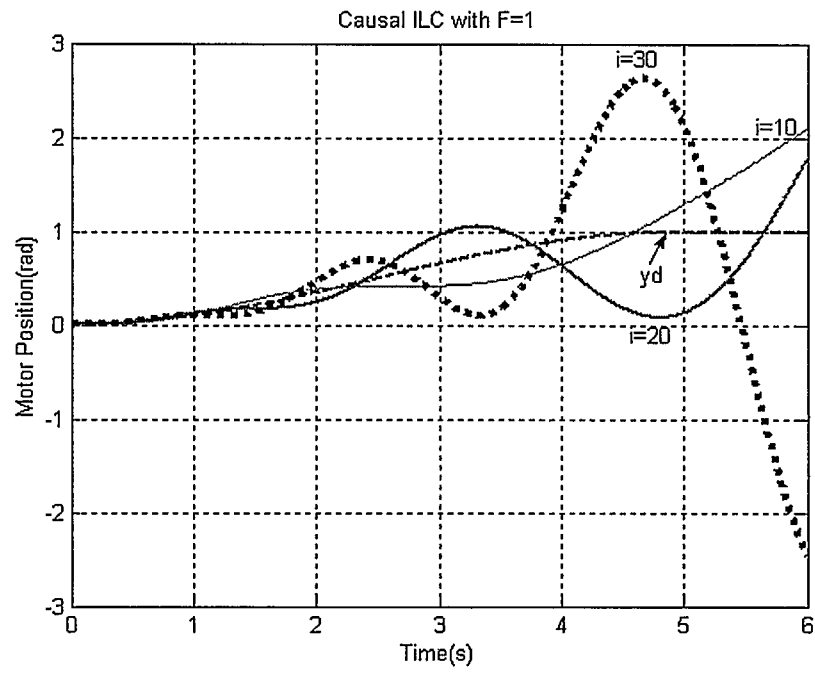


Figure 3.7: Causal ILC, on Simulated Motor

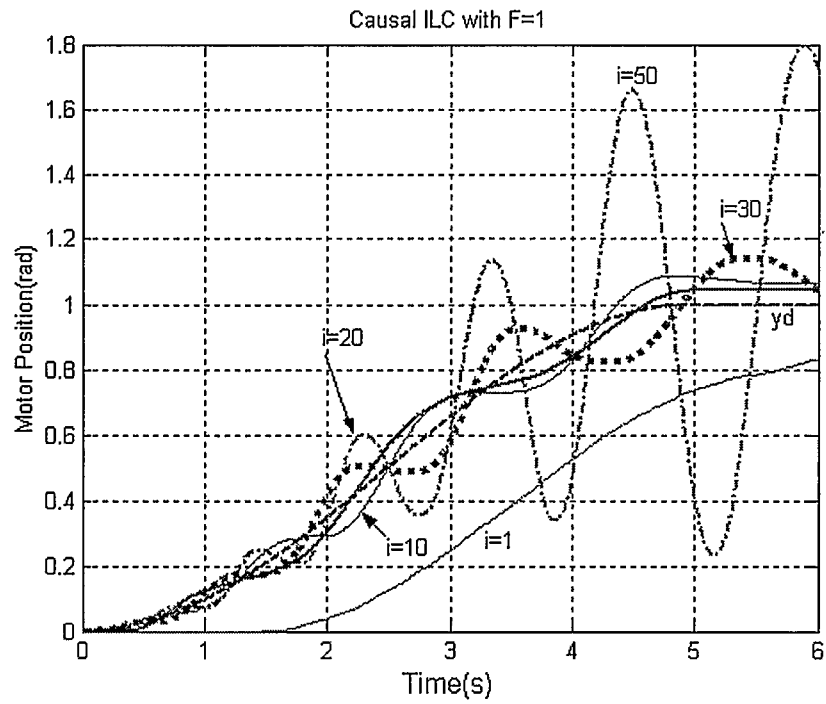


Figure 3.8: Causal ILC, on Real Motor

3.7.3.2 Causal ILC when $F \neq 1$

In order to make the system convergent, we set $F = L = \frac{1}{(0.5s+1)^2}$ and $D = LP^{-1}$

in (3.4). After performing simulation test we observe that the system is convergent as shown in Figure 3.9. Experimenting on the DC motor, the system also converges as shown in Figure 3.10. It is easy to see that the trade-off to making the system convergent is that ultimate error $e_\infty \neq 0$ at the fixed point.

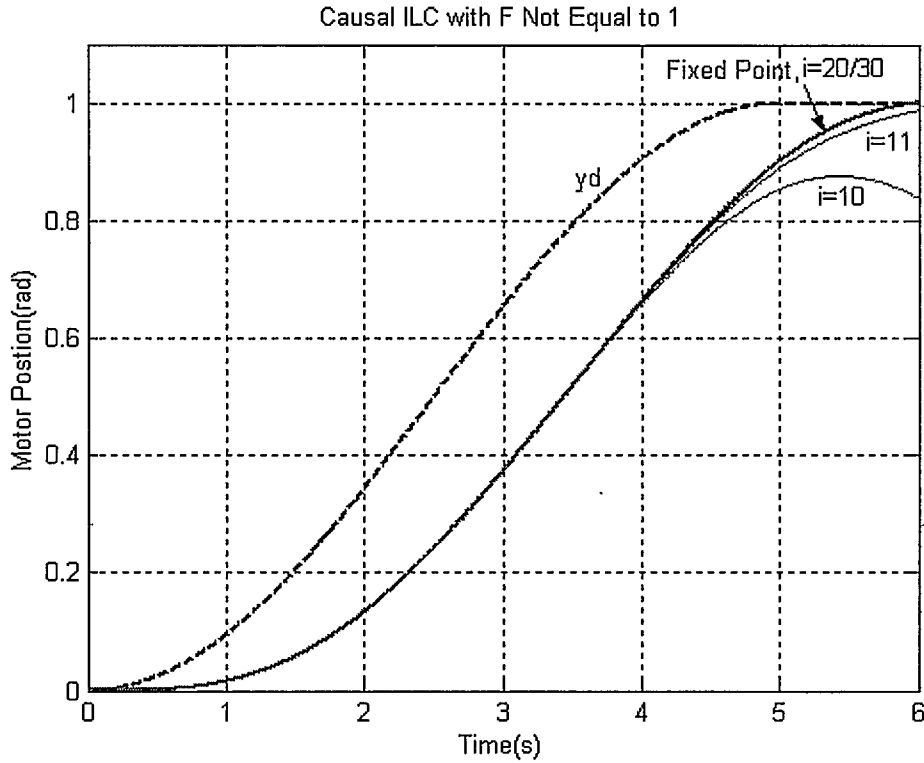


Figure 3.9: Causal ILC with $F \neq 1$, on Simulated Motor

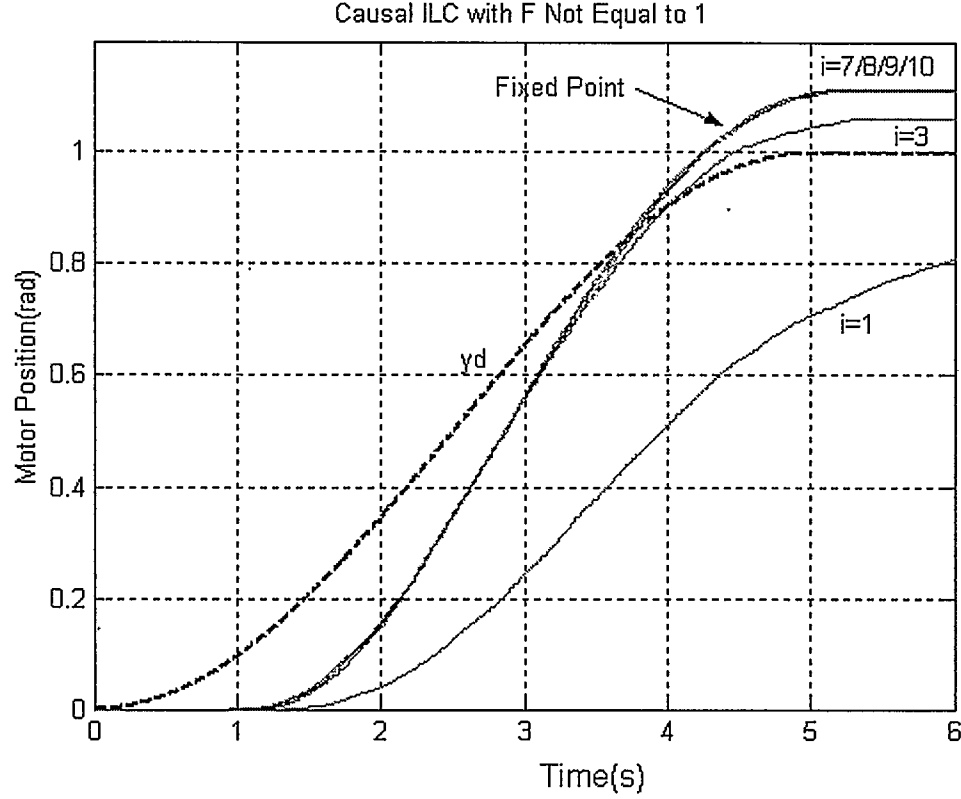


Figure 3.10: Causal ILC with $F \neq 1$, on Real Motor

3.7.3.3 Equivalent Feedback Control for Nonzero Ultimate Error

According to Theorem 1, the equivalent feedback control $u = Ke$ achieves $e = e_\infty$ without iterations, where $K = (1 - F)^{-1}(C + D)$ and $e_\infty = (1 + PK)^{-1}y_d$. Applying the feedback control to the system, the simulation result is shown in Figure 3.11.

In the experimental control scheme illustrated in Figure 2.7, replacing the PID controller with K and setting ILC input to zero gives the experimental result shown in Figure 3.12. Comparing Figure 3.11 with Figure 3.9, and Figure 3.12 with Figure 3.10 respectively, we find the fixed point in Figure 3.9 is the same as the output of Figure 3.11, and the fixed point in Figure 3.10 is the same as the output of Figure 3.12. Thus Theorem 1 is verified by experiment.

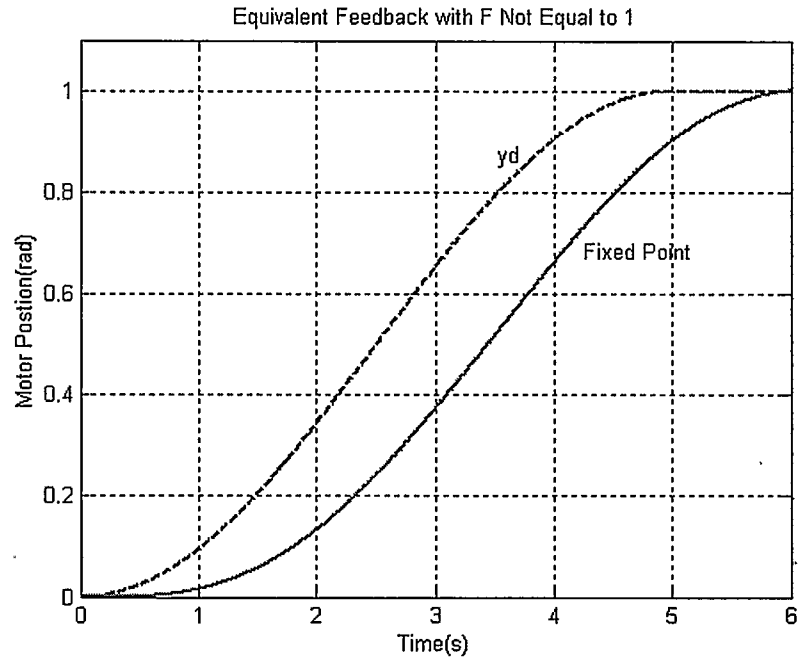


Figure 3.11: Equivalent Feedback Control with $F \neq 1$, on Simulated Motor

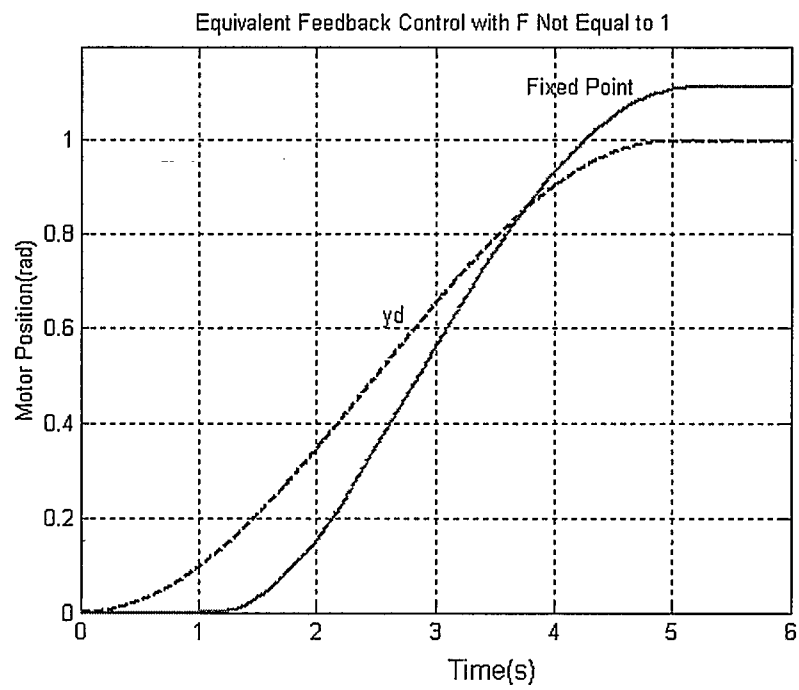


Figure 3.12: Equivalent Feedback Control with $F \neq 1$, on Real Motor

3.8 Summary of Causal ILC Results

Summarizing the simulation and experimental results, we have Table 3.1.

Table 3.1: Verification Results of Lemma 2 and Equivalent Feedback Control Theory

Verification of theory	Case	Simulation result	Experiment Result
Converse of Lemma 2	NMP, $P(s) = \frac{s-1}{(s+1)^2}$	Divergent (Fig. 3.4)	none
	$P(s) = \frac{1}{s+1}$ with $F = 1$	Convergent (Fig. 3.5)	none
Converse of Lemma 2	DC motor with $F = 1$, $P(s) = \frac{1}{s(0.03s+1)}$	Divergent (Fig. 3.7)	Divergent (Fig. 3.8)
	DC motor with $F \neq 1$, $P(s) = \frac{1}{s(0.03s+1)}$	Convergent (Fig. 3.9)	Convergent (Fig. 3.10)
Equivalent Feedback Control Theorem 1	$P(s) = \frac{1}{s+1}$ with $F = 1$	Convergent (Fig. 3.6)	none
Equivalent Feedback Control Theorem 2	DC motor with $F \neq 1$, $P(s) = \frac{1}{s(0.03s+1)}$	Convergent (Fig. 3.11)	Convergent (Fig. 3.12)

From the theoretical analysis and experimental validation, we obtained the following conclusions about causal ILC:

- Causal ILC has limitations for NMP processes and those plants with relative degree greater than 1. For these systems causal ILC cannot achieve zero tracking error. The tracking performance is either divergent or $e_{\infty} \neq 0$.
- Causal ILC has no advantages over Feedback Control because equivalent feedback control can achieve the same accuracy in just one trial, while causal ILC may need many trials to achieve this accuracy.
- Since an equivalent feedback control exists for any causal LTI ILC algorithm, it is better to use equivalent feedback control instead of ILC for causal LTI systems in order to avoid iterations.

Chapter 4

Noncausal ILC

Since causal ILC has some fundamental drawbacks, it is necessary to explore noncausal ILC. The reason causal ILC fails is that it satisfies the convergence condition only for MP plants with relative degree less than one. In [32] the authors promoted applying noncausal operators in ILC, while in paper [29] the author suggested a symmetrical noncausal filter to guarantee convergence for most common processes.

In this chapter, we will illustrate noncausal signal convolution and explain how the symmetrical noncausal operator influences the convergence of ILC system. Since few experiments has been implemented on real systems with noncausal ILC algorithms [33], in this chapter we present some experiments and simulations to verify theoretical results on noncausal ILC and show the benefit of noncausal ILC.

4.1 Convolution of Noncausal Signals

For an LTI system, the input and output relationship is determined by (3.38). But if P is a noncausal operator as shown in Figure 4.1(b), the output will be different. Figure 4.1(c) shows the convolution procedure and the value of the output, which is the shaded area before time t . We see that future values of the input signal u (the part when $\tau > t$) contribute to output $y(t)$ since p has nonzero values before $t = 0$. This property can be applied to improve causal ILC.

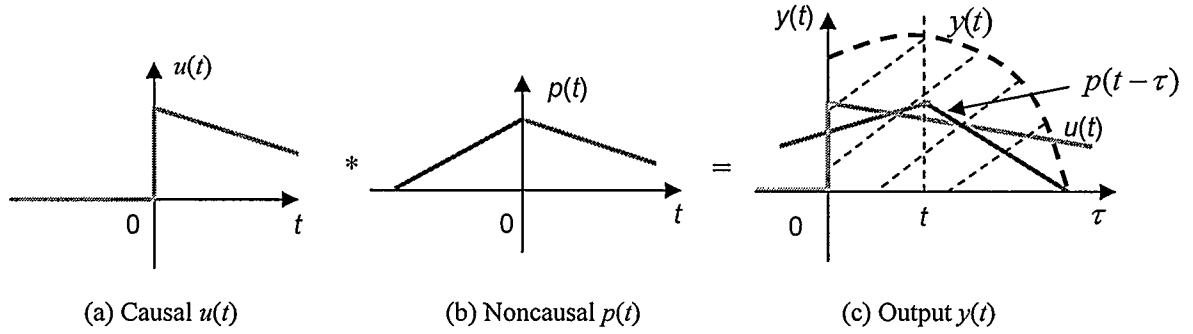


Figure 4.1: The Convolution Operation of Noncausal Signals

4.2 A Noncausal Symmetric Low-Pass Filter

From Lemma 2 in Chapter 3 we know that causal ILC can achieve zero ultimate error for only the simplest plants. When the relative degree of P is greater than one or P is NMP, the convergence condition $|H(j\omega)| < 1$ cannot be satisfied. In order to solve this problem, we need to find a suitable filter to satisfy the convergence condition even when the relative degree of P is greater than one or P is NMP. This kind of noncausal symmetric filter is proposed in [29]. An example of this kind of filter is

$$L(s) = \frac{1}{1 - \left(\frac{s}{\omega_0}\right)^2} = \frac{1}{\left(1 + \frac{s}{\omega_0}\right)\left(1 - \frac{s}{\omega_0}\right)}, \quad (4.1)$$

where ω_0 is the cut-off frequency. Setting $\omega_0 = 10$, the filter is plotted in time domain and frequency domain respectively in Figure 4.2 and Figure 4.3. If we suppose P is a stabilized process, then we may set $C = 0$ in the ILC algorithm (3.4). Thus $S = 1$ according to (3.23). Setting $F = 1$ to achieve $e_\infty = 0$, the H in (3.22) becomes

$$H = 1 - DP . \quad (4.2)$$

With $D = LP^{-1}$, rearranging (4.2) gives

$$H = 1 - L . \quad (4.3)$$

Substituting $s = j\omega$ into (4.1) gives

$$L(j\omega) = \frac{1}{1 + \left(\frac{\omega}{\omega_0}\right)^2} . \quad (4.4)$$

It is clear that $L(j\omega)$ has no complex part and $0 < L(j\omega) < 1$. Substituting (4.4) into (4.3) guarantees $|H(j\omega)| < 1$.

More generally, L can be chosen as

$$L(s) = \frac{1}{\prod_{j=1}^n 1 - \left(\frac{s}{\omega_j}\right)^2} , \quad (4.5)$$

where the ω_j are the cut-off frequencies, and the relative degree of L is $2n$. If $2n$ exceeds the relative degree of P , then D is proper. Substituting (4.5) into (4.3) gives $|H(j\omega)| < 1$ at all frequencies. Thus, this L results in a proper D and guarantees convergence for any relative degree process.

Figure 4.2 and Figure 4.3 show a filter L with relative degree of 2 in the time domain and in the frequency domain respectively.

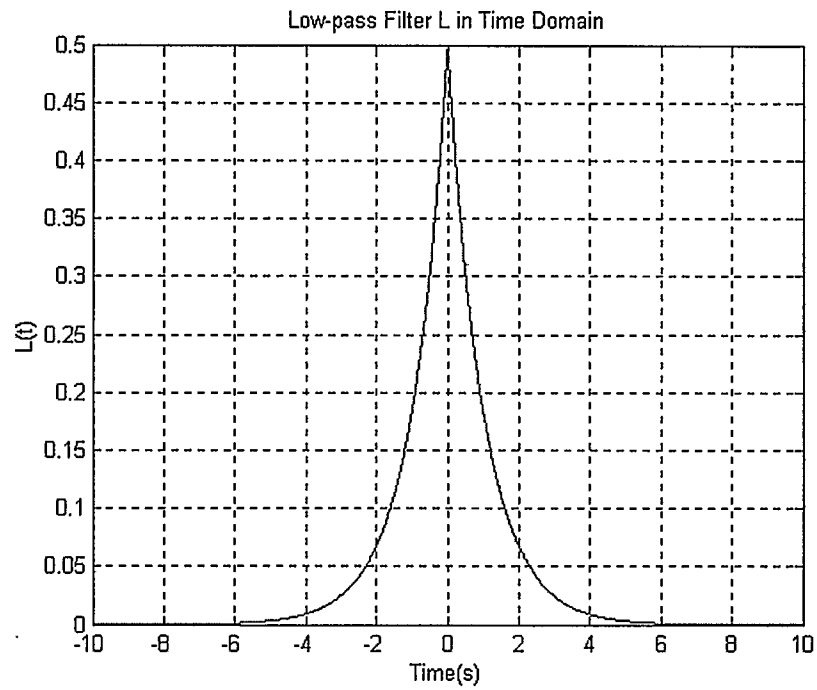


Figure 4.2: Filter L in the Time Domain

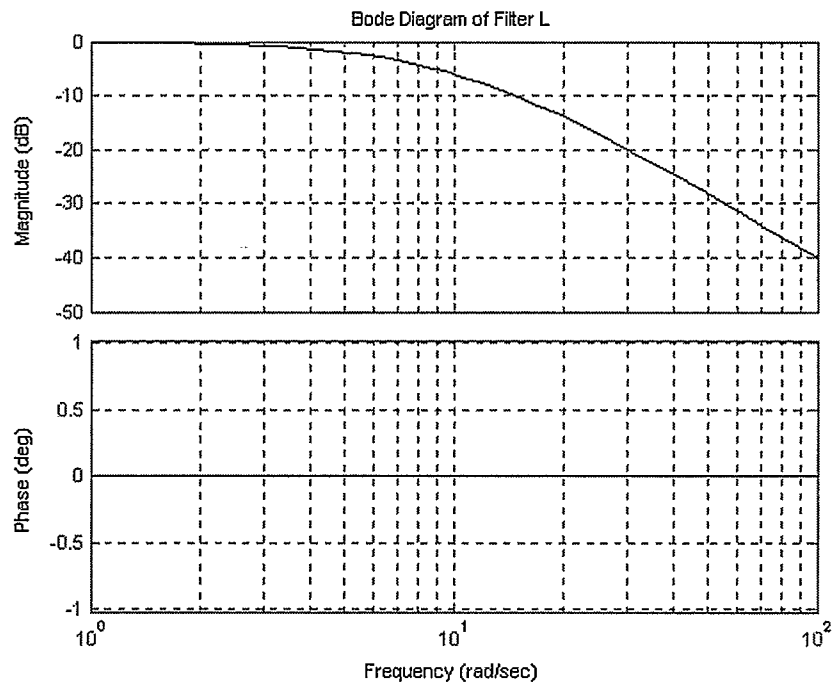


Figure 4.3: Filter L in the Frequency Domain

4.3 Noncausal ILC on a NMP Plant

Lemma 2 in Chapter 3 shows theoretically that if P is NMP, perfect tracking of a reference y_d is not possible with a causal ILC algorithm. Consider the example in Chapter 3, which is a NMP system. In this example, the output is given by

$$Y(s) = P(s)U(s), \quad (4.6)$$

where
$$P(s) = \frac{s-1}{(s+1)^2}. \quad (4.7)$$

Suppose the reference to be tracked is also the same as in the causal case in Chapter 3,

$$Y_d(s) = \frac{1}{(s+1)^2}. \quad (4.8)$$

To track it perfectly, the input should be

$$U(s) = P^{-1}(s)Y_d(s) = \frac{1}{s-1}. \quad (4.9)$$

The two-sided inverse Laplace transform of (4.9) has two solutions: a causal input $u(t) = e^t h(t)$ and an anti-causal input $u(t) = -e^t h(-t)$. In Chapter 3, we saw that the causal input drives the system unstable (and is itself unstable). So this time the anti-causal input is applied. The simulation result in Figure 4.4 shows that the system tracks the reference perfectly.

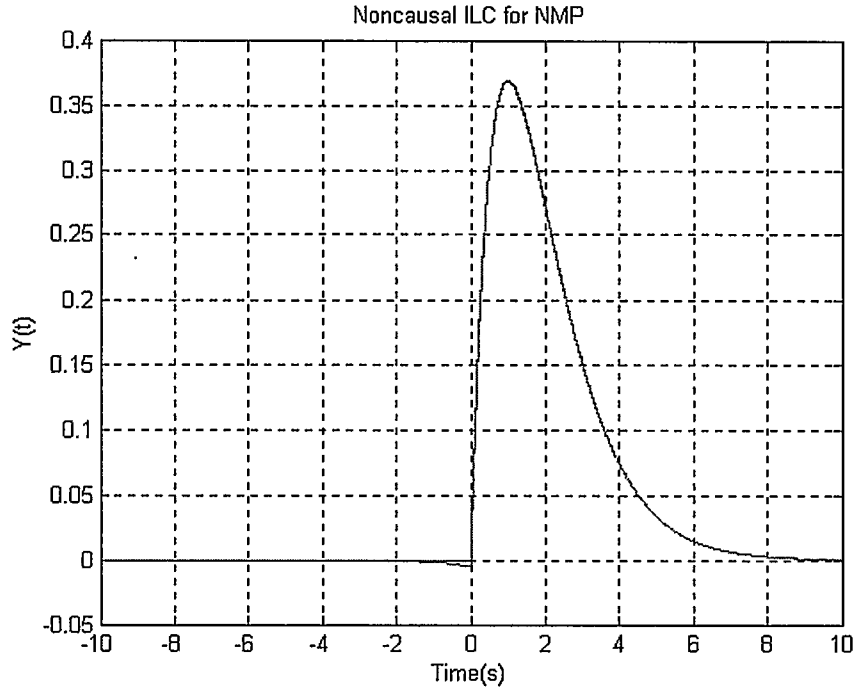


Figure 4.4: A NMP Process with Noncausal Input

4.4 Noncausal ILC on a DC Motor

In Chapter 3 we showed that it is not possible for a DC motor, which has a relative degree of 2, to converge to zero tracking error using causal ILC. To achieve convergence, we set $F = L$, which resulted in nonzero ultimate tracking error. To achieve convergence and zero ultimate error, we now apply noncausal operators in the ILC algorithm (3.4).

Recall that the DC motor has the open-loop model

$$G(s) = \frac{1.5}{s(0.03s + 1)} . \quad (4.10)$$

The closed-loop system is

$$e = y_d - G(w + u), \quad (4.11)$$

where u is ILC input and w is the feedback control, and

$$w = k_p e, \quad (4.12)$$

where k_p is proportional control to stabilize the system.

Substituting (4.12) into (4.11) gives

$$e = Sy_d - SGu, \quad (4.13)$$

$$\text{where} \quad S = (1 + Gk_p)^{-1}. \quad (4.14)$$

Then the ILC system can be written as

$$e_i = r - Pu_i, \quad (4.15)$$

$$\text{where} \quad r = Sy_d, \quad (4.16)$$

$$P = SG. \quad (4.17)$$

If we set $F = 1$, the ILC update law (3.4) can therefore be written as

$$u_i = u_{i-1} + De_{i-1}, \quad (4.18)$$

where $C=0$ since P is a stabilized closed-loop plant, and

$$D = LP^{-1}. \quad (4.19)$$

According to (4.17),

$$P(s) = \frac{1.5}{0.03s^2 + s + 1.5k_p}. \quad (4.20)$$

We set

$$L(s) = \frac{1}{\left(1 + \frac{s}{\omega_0}\right)^2 \left(1 - \frac{s}{\omega_0}\right)^2}, \quad (4.21)$$

where $\omega_0 = 1$. Since L is a noncausal operator, so is D .

Applying the noncausal ILC algorithm indicated by (4.18) in simulation, we get the result shown in Figure 4.5. When we apply (4.18) to the real motor, the result is shown in

Figure 4.6. These figures show that noncausal ILC converges to $e_\infty = 0$. In contrast, Figures 3.7 and 3.8 in Chapter 3 show that causal ILC with $F = 1$ made the system divergent. Thus noncausal ILC improves on causal ILC when the relative degree of the plant is greater than 1.

Whereas we obtained noncausal ILC results in the frequency domain, we need to implement them in the time domain. Since ILC must operate over a limited time interval, signals must be truncated at the beginning and end of the interval. This results in loss of information.

The task in this example was to follow the reference between 0 and 5 seconds. Figure 4.5 and Figure 4.6 show that the tracking is relatively poor between 5 and 6 seconds. This “follow-through” region was used for learning, but is not part of the task. Without this follow-through region in Figure 4.5 or Figure 4.6, there will be no future information to be calculated in the noncausal ILC algorithm at the end point of the task. Comparing the tracking accuracy between 4 seconds and 5 seconds with that between 5 seconds and 6 seconds in Figure 4.5, we see that the loss of future information at the end of the task reduces the tracking performance. For a ball and beam experiment described in the next section, we will improve the performance at the end of a task by using a follow-through region.

Comparing Figures 4.5 and 4.6, we see a difference between the experimental result and the simulation result. This is caused by unmodeled parameters and uncertainties in the plant.

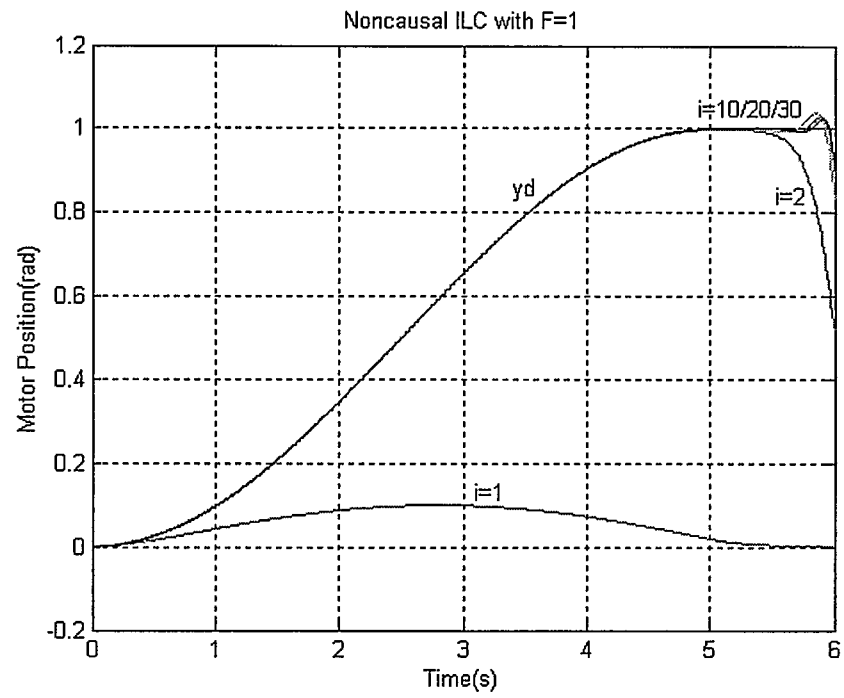


Figure 4.5: Noncausal ILC on Simulated Motor

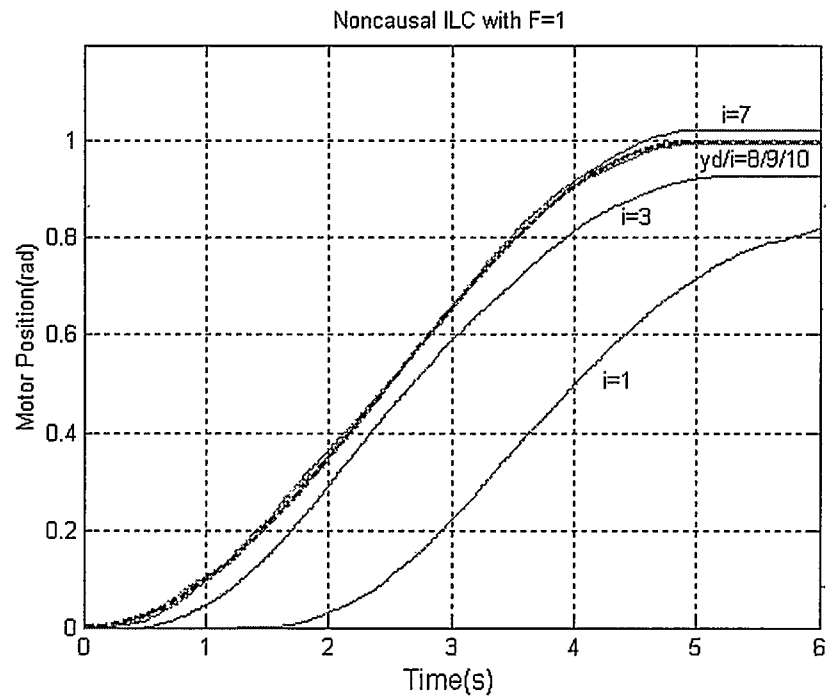


Figure 4.6: Noncausal ILC on Real Motor

4.5 ILC on a Higher Order System—Ball and Beam

The experiment on the DC motor proves that noncausal ILC can improve on causal ILC, since noncausal ILC allows the DC motor to achieve zero ultimate tracking error while causal ILC does not. A DC motor is a second order process with relative degree of 2. What happens if noncausal ILC is applied to a higher order process with a higher relative degree? This question can be investigated by an experiment on a Ball and Beam system, which has a relative degree of 4.

4.5.1 Mathematical Model of Ball and Beam System

As shown in Figure 4.7, Ball and Beam system is composed of two modules, which are a Ball and Beam module and a servomotor module. A lever arm is attached to one of the gears of the servomotor. The servomotor drives the lever arm. As a result, the angle of the beam is changed and the ball resting on the beam will be moved to a desired position under the effect of gravity. This configuration is illustrated in Figure 4.8.

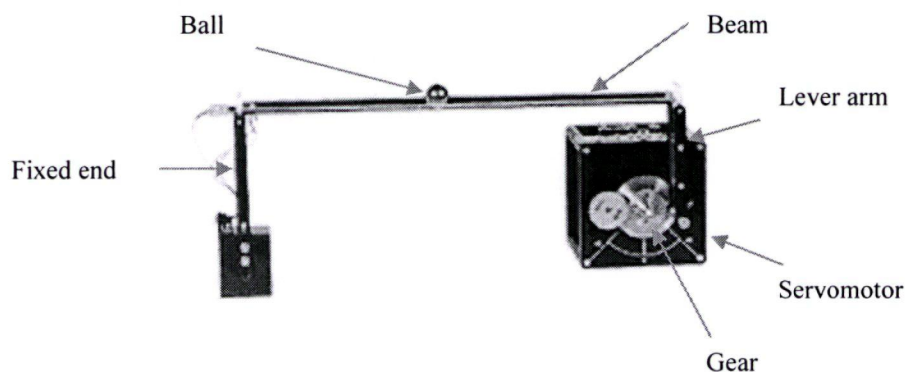


Figure 4.7: Ball and Beam System [40]

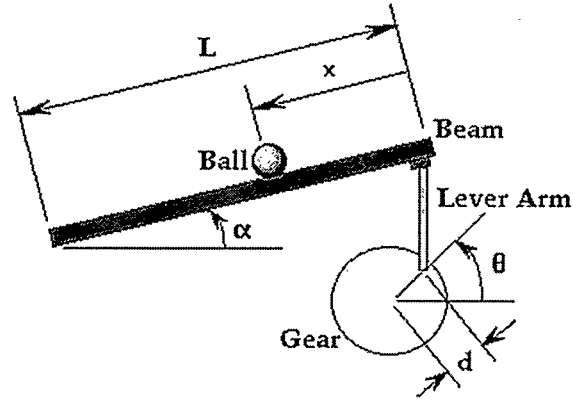


Figure 4.8: Configuration of Ball and Beam System [41]

Consider the ball, which has mass m , inertia J and diameter d , we have

$$J\ddot{\phi} = \frac{1}{2}mgd \sin \alpha, \quad (4.22)$$

and
$$x = \frac{1}{2}d\phi. \quad (4.23)$$

Differentiating (4.23) twice gives

$$\ddot{\phi} = \frac{2\ddot{x}}{d}. \quad (4.24)$$

Substituting (4.24) into (4.22) gives

$$\ddot{x} \approx \frac{5}{7}g\alpha. \quad (4.25)$$

When α and θ are small, we have

$$\alpha = \frac{d}{L}\theta, \quad (4.26)$$

where $\frac{d}{L} = \frac{1}{17}$ in this particular device.

Substituting (4.26) into (4.25) gives

$$\ddot{x} = 0.412\theta, \quad (4.27)$$

or in the frequency domain,

$$\frac{X(s)}{\theta(s)} = \frac{0.412}{s^2}. \quad (4.28)$$

Recall the motor's transfer function

$$\frac{\theta(s)}{V_{in}(s)} = \frac{1.5}{s(0.03s + 1)}. \quad (4.29)$$

Transferring (4.29) into time domain gives

$$\ddot{\theta} = -33.3\dot{\theta} + 50v. \quad (4.30)$$

The Ball and Beam system transfer function is obtained by multiplying (4.28) with (4.29) as

$$G(s) = \frac{X(s)}{V_{in}(s)} = \frac{0.618}{s^3(0.03s + 1)}. \quad (4.31)$$

It is easy to see that the open loop system is not stable.

4.5.2 Stabilization of the Plant

We designed a control system in Simulink as schematically shown in Figure 4.9. This control system is primarily composed of a reference signal generator, a feedback controller (middle part), an ILC controller and the physical plant Ball and Beam module. We use a GUI to operate the system and use a Matlab code to process the data obtained from each iteration. The scope labelled “ILC Input” records the previous ILC input signal u_{i-1} , and the scope labelled “Error” records the previous error signal e_{i-1} . By applying

From (4.27) and (4.30), we have

$$\dot{X} = AX + Bv \quad (4.33)$$

$$y = CX \quad (4.34)$$

where
$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.412 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -33.3 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 50 \end{bmatrix}, \text{ and } C = [1 \ 0 \ 0 \ 0].$$

Suppose the control is

$$v = KX + u, \quad (4.35)$$

where K is feedback control gain and u is ILC input.

Substituting (4.35) into (4.33) gives

$$\dot{X} = A_{cl}X + Bu, \quad (4.36)$$

where
$$A_{cl} = A + BK \quad (4.37)$$

is the closed-loop system matrix.

The open-loop characteristic equation is

$$\det(sI - A) = s^4 + 33.3s^3. \quad (4.38)$$

Then
$$a = [33.3 \ 0 \ 0 \ 0]. \quad (4.39)$$

Suppose we choose the closed-loop characteristic equation as following:

$$\begin{aligned} \det(sI - A_{cl}) &= (s+1)(s+2)(s+3)(s+30) \\ &= s^4 + 36s^3 + 191s^2 + 336s + 180, \end{aligned} \quad (4.40)$$

then
$$r = [36 \ 191 \ 336 \ 180]. \quad (4.41)$$

From (4.39), we have Toeplitz matrix

$$A_T = \begin{bmatrix} 1 & 33.3 & 0 & 0 \\ 0 & 1 & 33.3 & 0 \\ 0 & 0 & 1 & 33.3 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

and

$$A_T^{-1} = \begin{bmatrix} 1 & -33 & 1109 & -36926 \\ 0 & 1 & -33 & 1109 \\ 0 & 0 & 1 & -33 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Since $W_c = [B \ AB \ A^2B \ A^3B]$, we have

$$W_c^{-1} = \begin{bmatrix} 0 & 0 & 0.66 & 0.02 \\ 0 & 1.6165 & 0.02 & 0 \\ 1.6165 & 0.0485 & 0 & 0 \\ 0.0485 & 0 & 0 & 0 \end{bmatrix}.$$

Designing K according to Bass-Gura formula gives, [47]

$$\begin{aligned} K &= [a - r] A_T^{-1} W_c^{-1} \\ &= [-8.7379 \quad -16.3107 \quad -3.8200 \quad -0.0540] \end{aligned} \quad (4.42)$$

From the control scheme in Figure 4.9, we have

$$v = K_p(\theta_d - \theta) + K_d\dot{\theta} + u, \quad (4.43)$$

and

$$\theta_d = \frac{L}{d} [K_{bp}(x_d - x) + K_{bd}\dot{x}]. \quad (4.44)$$

Substituting (4.44) into (4.43) and rearranging give

$$\begin{aligned} v &= \begin{bmatrix} -\frac{L}{d} K_{bp} K_p & \frac{L}{d} K_{bd} K_p & -K_p & K_d \end{bmatrix} * \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \frac{L}{d} x_d K_{bp} + u \\ &= KX + \frac{L}{d} x_d K_{bp} + u \end{aligned} \quad (4.45)$$

Comparing (4.42) and (4.45) gives

$$\begin{aligned} K_{bp} &= 0.1346 \\ K_{bd} &= -0.2512 \\ K_p &= 3.82 \\ K_d &= -0.054 \end{aligned}$$

Substituting (4.42) into (4.37) gives closed-loop system matrix as

$$A_{cl} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.412 & 0 \\ 0 & 0 & 0 & 1 \\ -436.8932 & -815.534 & -191 & -36 \end{bmatrix}. \quad (4.46)$$

4.5.3 Closed-loop Transfer Function

From (4.36), we have

$$X = (sI - A_{cl})^{-1} Bu. \quad (4.47)$$

Substituting (4.47) into (4.34) gives

$$\begin{aligned} P_0(s) &= \frac{y(s)}{u(s)} = C(Is - A_{cl})^{-1} B \\ &= \frac{20.6}{(s+1)(s+2)(s+3)(s+30)} \end{aligned} \quad (4.48)$$

4.5.4 Setting Initial Point of the Ball in Iteration

Since Iterative Learning Control strongly depends on the same initial point, it is important to make sure that for each trial the ball is always at the same initial point. If the initial point is not the same from trial to trial, the ILC control system will be affected. To set the initial condition, feedback control is applied, as shown in the control scheme of Figure 4.9.

Even with feedback control applied, the initial condition sometimes varied. As an example, Figures 4.10 and 4.11 show the 20th trial of two separate experiments. In Figure 4.10, the initial position (at $t = 10$ s) is constant and zero, whereas in Figure 4.11, it has not yet settled. The result is the poorer tracking performance observed in Figure 4.11.

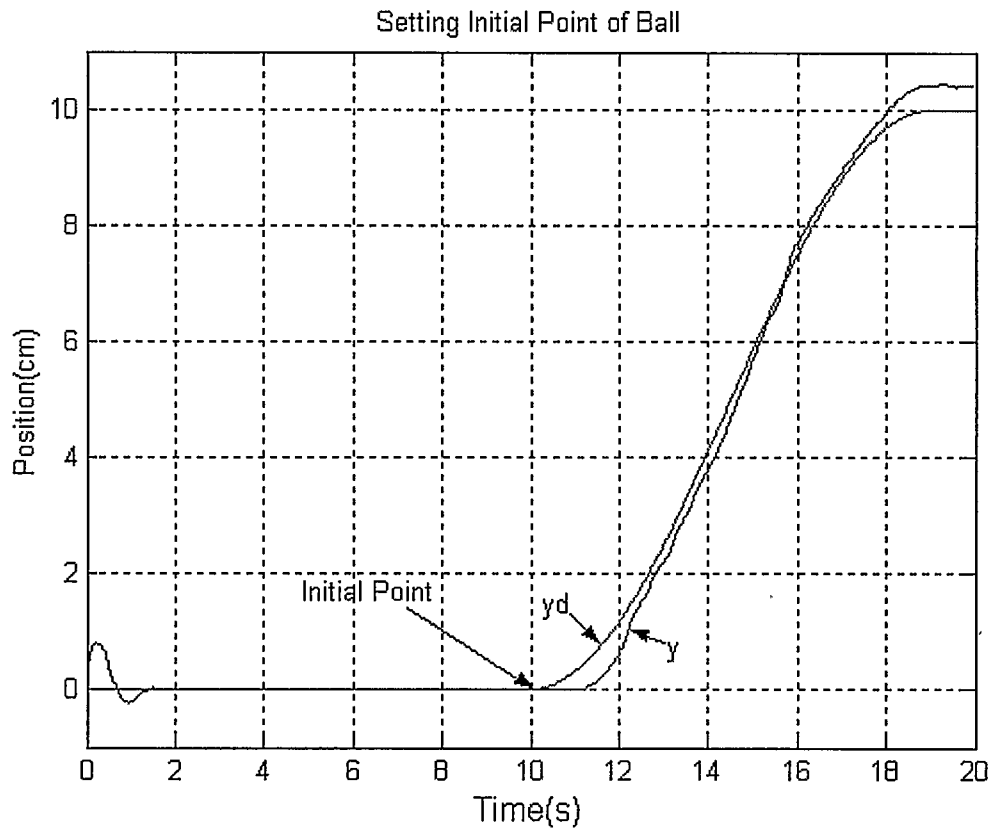


Figure 4.10: Setting Initial Point of Ball at the 20th trial

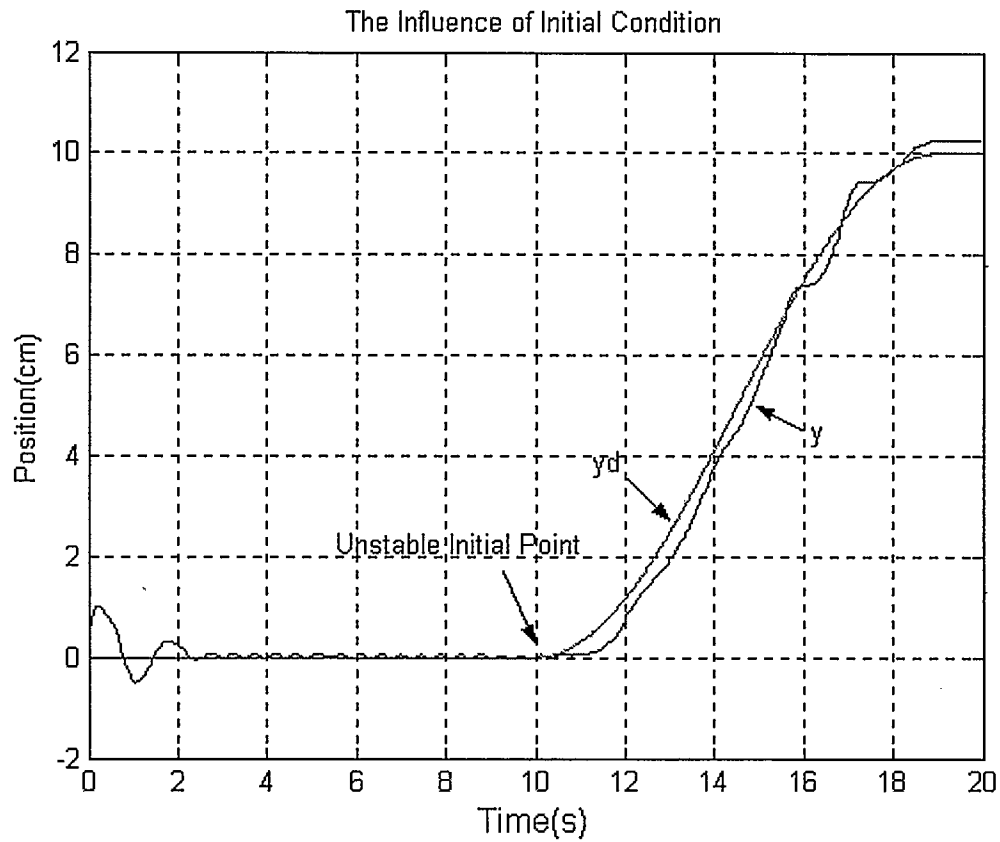


Figure 4.11: Influence of unsteady Initial Point to ILC System Performance at the 20th trial

4.5.5 Causal ILC on Ball and Beam System

4.5.5.1 For the Case $F = 1$

Initially we choose $F = 1$ to attempt to get a zero ultimate tracking error. Then the ILC update law is $u_i = u_{i-1} + De_{i-1}$. To make D proper, we set $L = \frac{1}{(0.5s+1)^4}$ in $D = LP_0^{-1}$. Since L is causal, so is D . The simulation results in Figure 4.12 show that the system is divergent and thus validates the Lemma 2 since this system's relative degree is 4. The real system results are also divergent as shown in Figure 4.13, which further

verifies the correctness of Lemma 2. From the two figures, we may see that the vibration becomes bigger and bigger, and the divergence is very clear at $i = 150$.

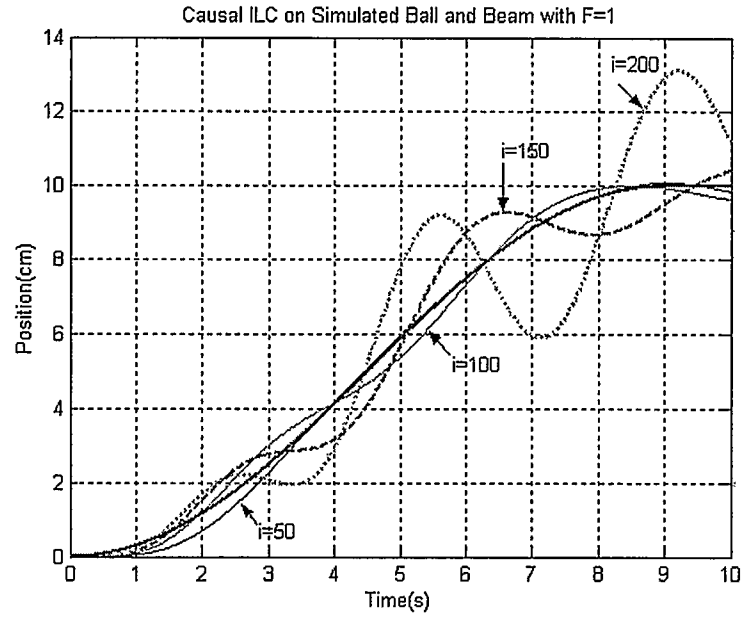


Figure 4.12: Causal ILC on Simulated Ball and Beam System with $F = 1$

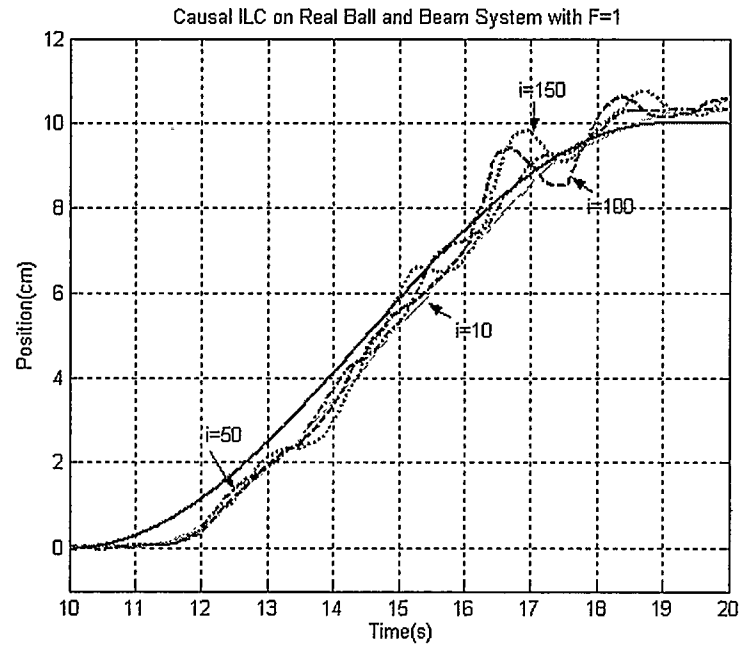


Figure 4.13: Causal ILC on Real Ball and Beam System with $F = 1$

4.5.5.2 In the Case $F \neq 1$

The simulation and experimental results in Figure 4.12 and Figure 4.13 prove that the system will diverge when applying causal ILC with $F = 1$. To avoid the divergence,

we set $F = L = \frac{110}{(0.5s + 1)^4}$ and $D = LP_0^{-1}$. Thus the ILC law is $u_i = Fu_{i-1} + De_{i-1}$.

Applying this on the simulated system gives results shown in Figure 4.14, while applying it on the real system gives results shown in Figure 4.15. Both the simulation results and the real system results show that the system converges with $F=L$, but the trade-off is that $e_\infty \neq 0$.

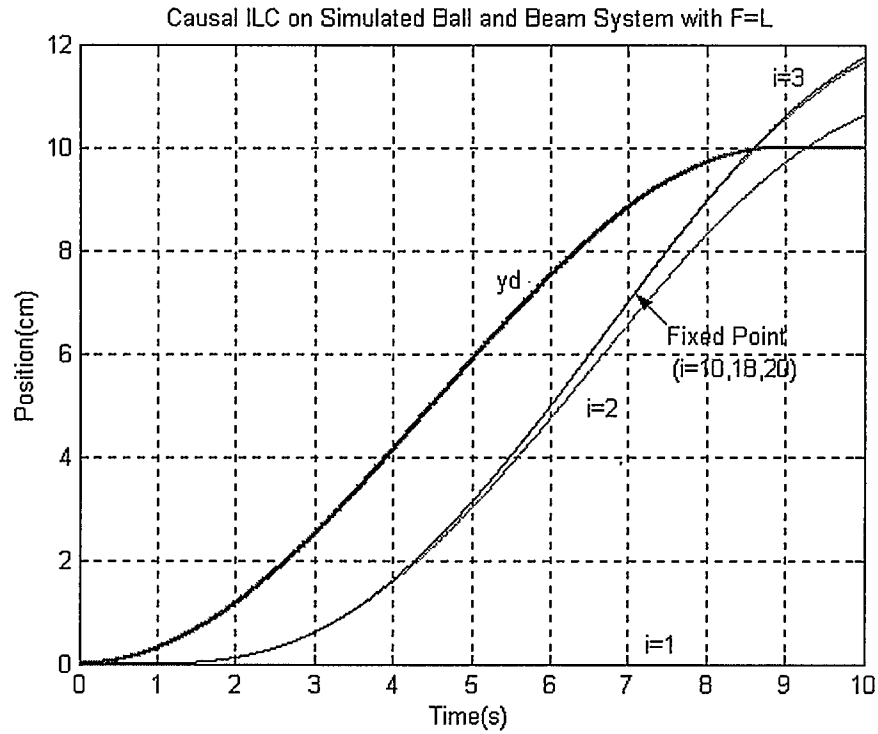


Figure 4.14: Causal ILC on Simulated on Ball and Beam System with $F \neq 1$

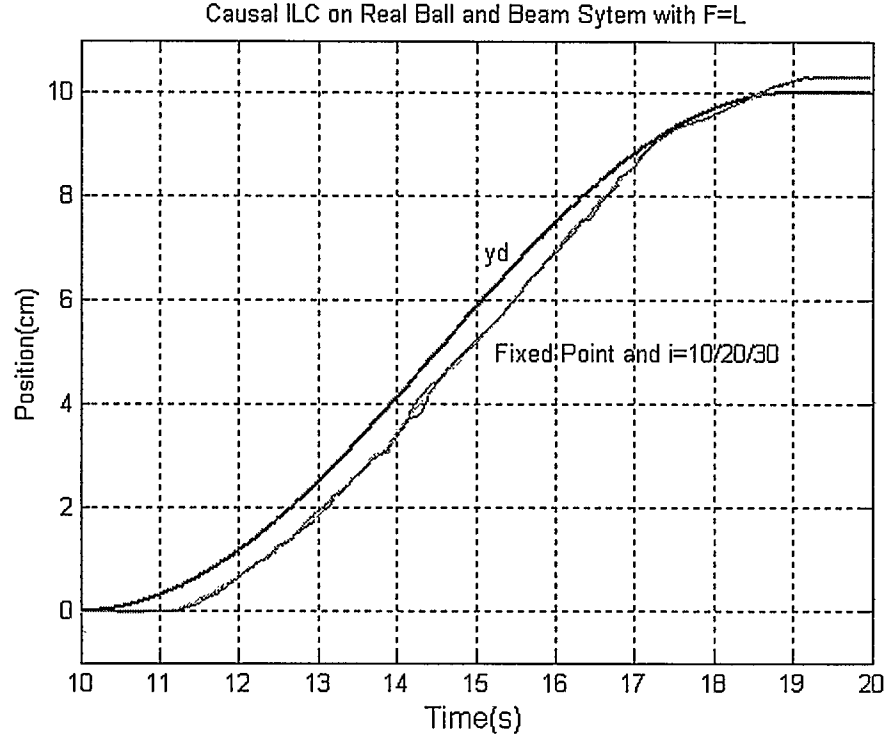


Figure 4.15: Causal ILC on Real Ball and Beam System with $F \neq 1$

4.5.6 Equivalent Feedback Control on Ball and Beam System

Since all operators L , F , and D are causal, there exists an equivalent feedback control for the causal ILC. According to Theorem 1, the equivalent feedback control gain is $K = (1 - F)^{-1}(C + D)$, where $F \neq 1$. Correspondingly, the output should be $y = PK(1 + PK)^{-1}y_d$. In the simulation, the result is as shown in Figure 4.17. Applying K in the control scheme as shown in Figure 4.16, we get the equivalent feedback control result from the real system as shown in Figure 4.18. Comparing Figure 4.14 with Figure 4.17, and Figure 4.15 with Figure 4.17 respectively, we see that the equivalent feedback control can make the system achieve the same fixed point as causal ILC with only one trial.

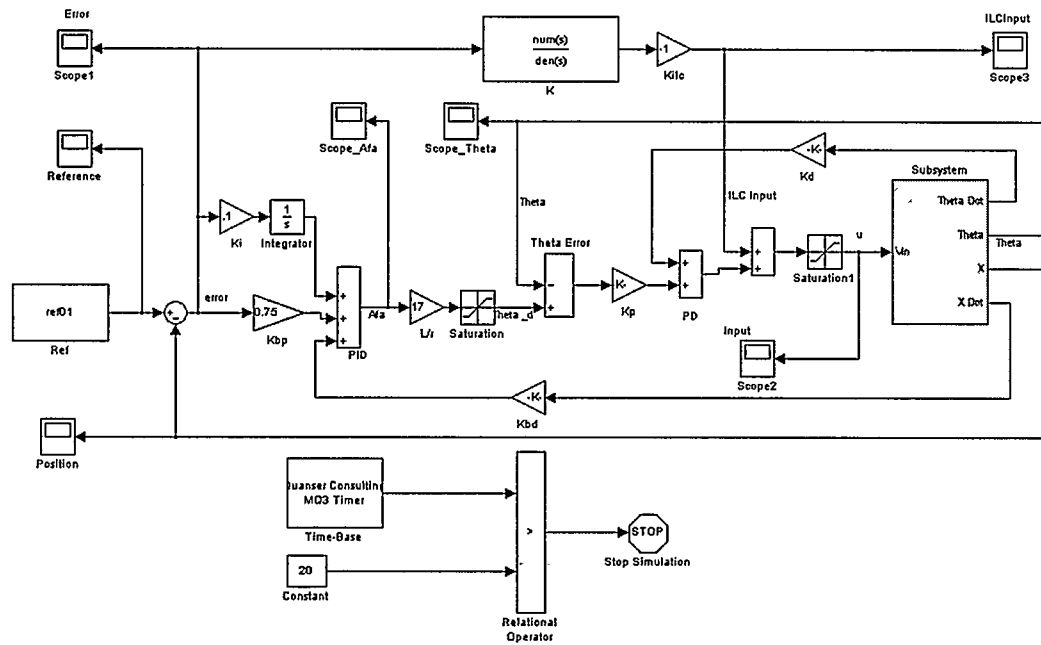


Figure 4.16: Equivalent Feedback Control on Ball and Beam

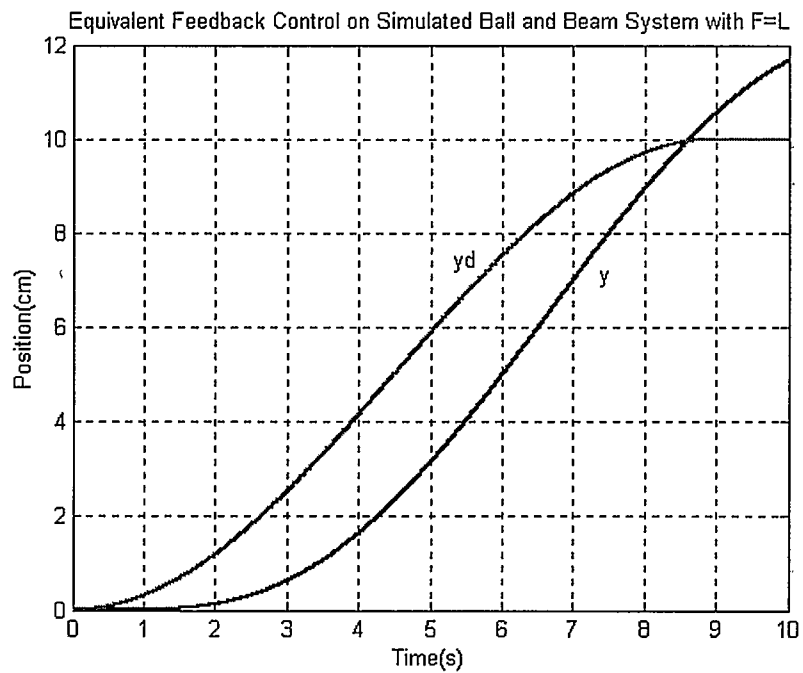


Figure 4.17: Equivalent Feedback Control on Simulated Ball and Beam System

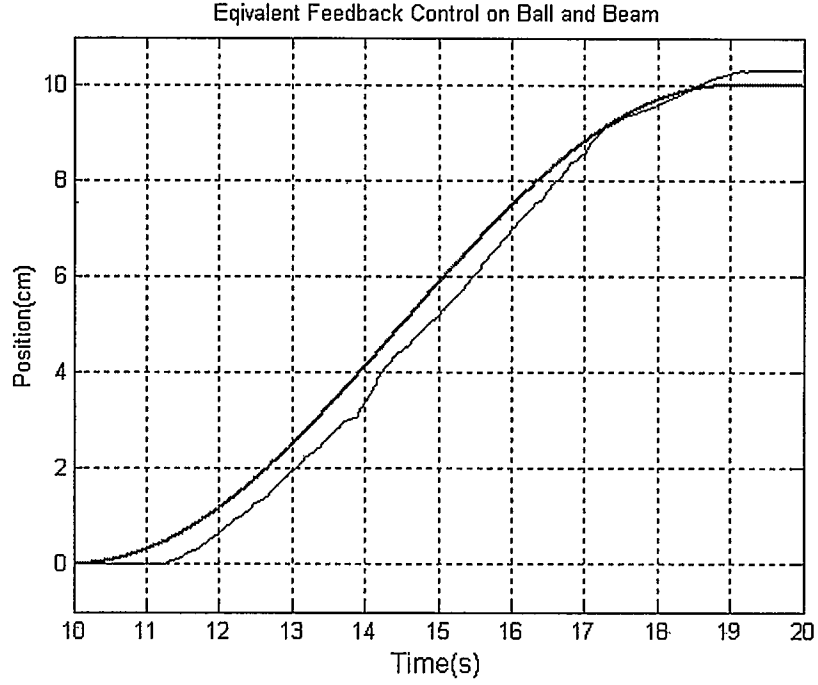


Figure 4.18: Equivalent Feedback Control on Real Ball and Beam System

4.5.7 Noncausal ILC on the Ball and Beam System

In order to achieve zero ultimate error $e_\infty = 0$, we apply a noncausal ILC algorithm on the system as

$$u_i = u_{i-1} + D e_{i-1}, \quad (4.49)$$

where

$$D(s) = L P_0^{-1} = \frac{(s+1)(s+2)(s+3)(s+30)}{\left(1 - \frac{s}{\omega_0}\right)^2 \left(1 + \frac{s}{\omega_0}\right)^2}. \quad (4.50)$$

When $\omega_0 = 0.01$, we get the simulation results shown in Figure 4.19. These simulation results show that the system follows the reference quite well when the iteration number reaches $i=100$. The results obtained from the real system are shown in Figure 4.20.

They are very similar to those in the simulation. At $i=100$, the tracking error is approximately zero. .

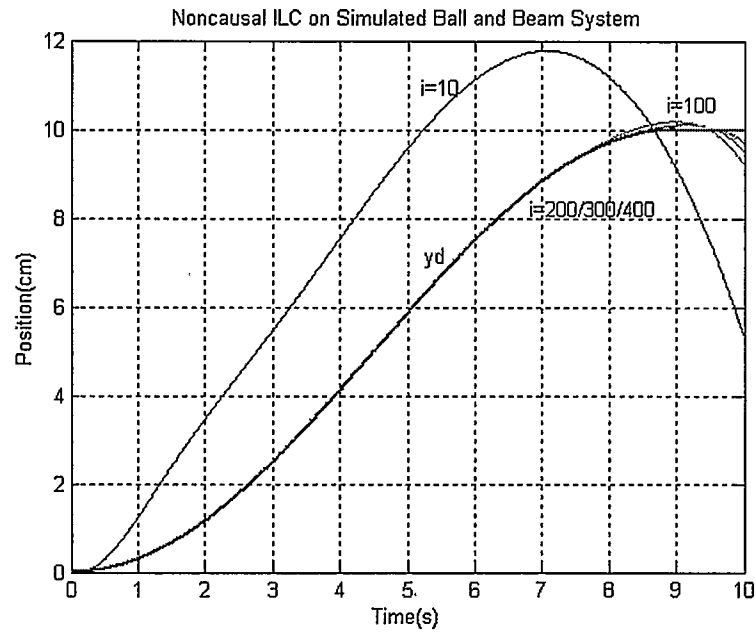


Figure 4.19: Noncausal ILC on Simulated Ball and Beam System

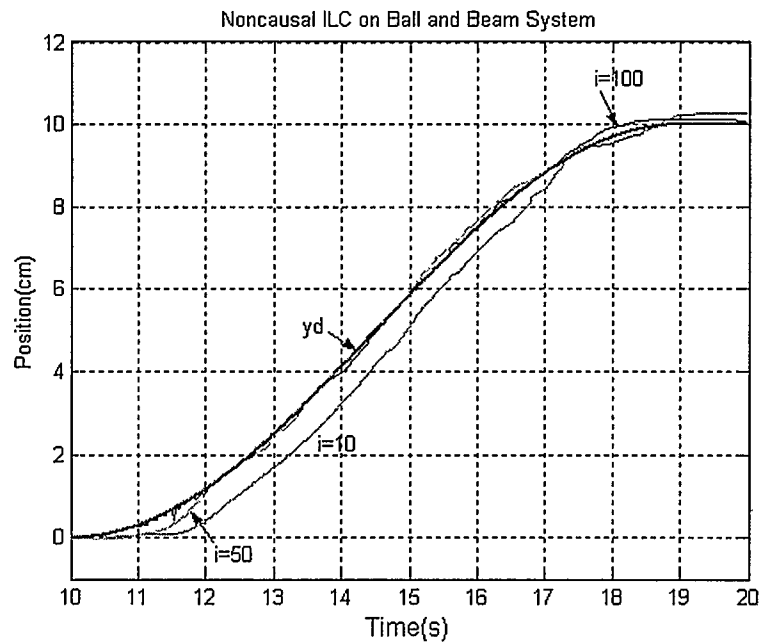


Figure 4.20: Noncausal ILC on Real Ball and Beam System

4.5.8 Noncausal ILC Reduces Phase Delay

Since the Ball and Beam system is a 4th order system and its relative degree is 4, when it tracks a higher frequency reference, a phase delay will appear. Figure 4.21 shows an example where the system is required to follow a higher frequency sine wave from rest. The feedback control performs very well in the first task, but produces a big delay when tracking the higher frequency signal. For this example the angular frequency of the sine signal is $\omega = 1.4$ rad/s. From the Bode plot of P_0 (Figure 4.22), the system's phase delay is greater than 90° at this frequency.

Applying noncausal ILC (4.49) produces the simulation results shown in Figure 4.23 and the experimental results shown in Figure 4.24. From the simulation results we notice that the tracking is satisfactory when iteration number reaches one thousand. On the real system experiment improvement is apparent when the iteration number reaches one hundred.

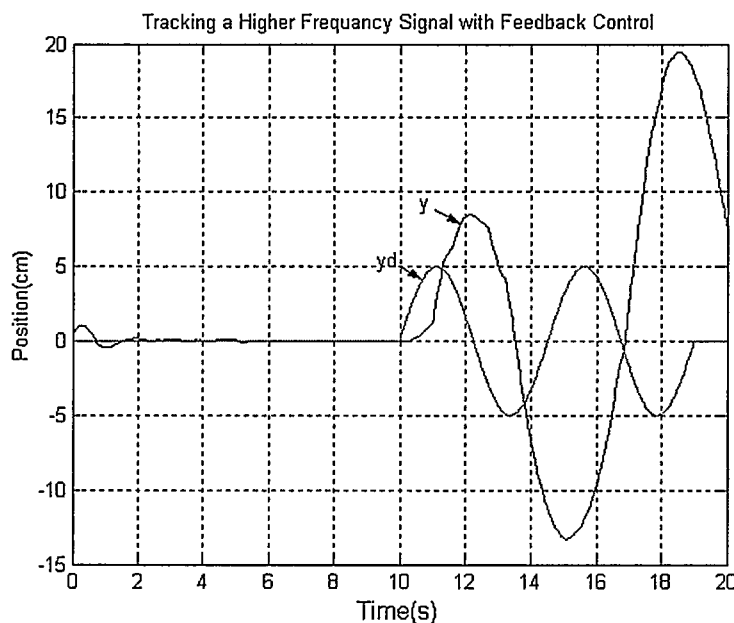


Figure 4.21: Tracking a High Frequency Signal with Feedback Control

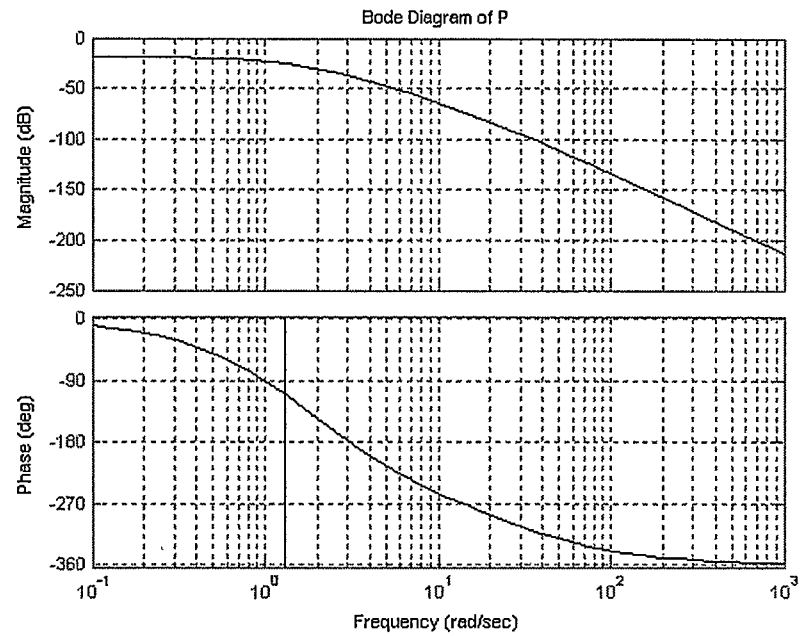


Figure 4.22: Bode Plot of Process Model

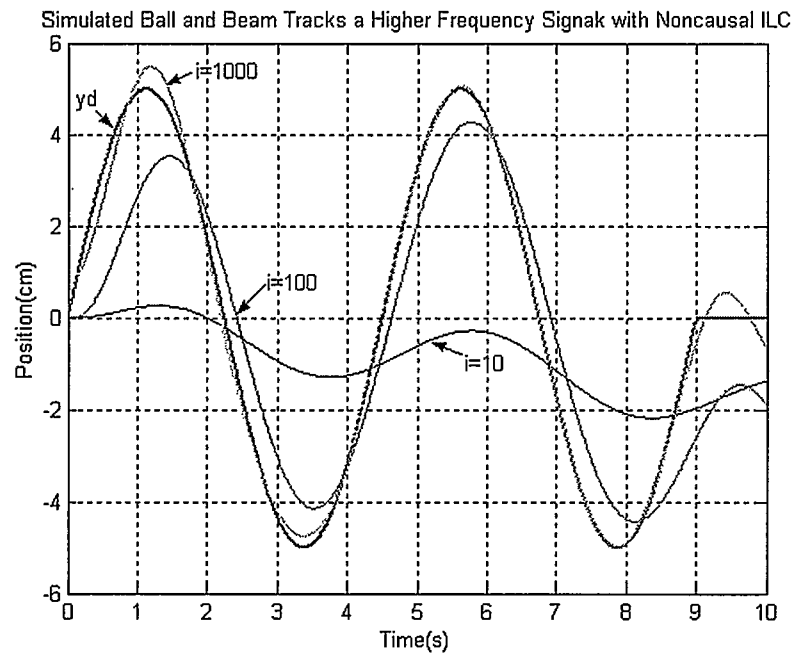


Figure 4.23: Tracking a Higher Frequency Signal with Noncausal ILC on Simulated Ball and Beam

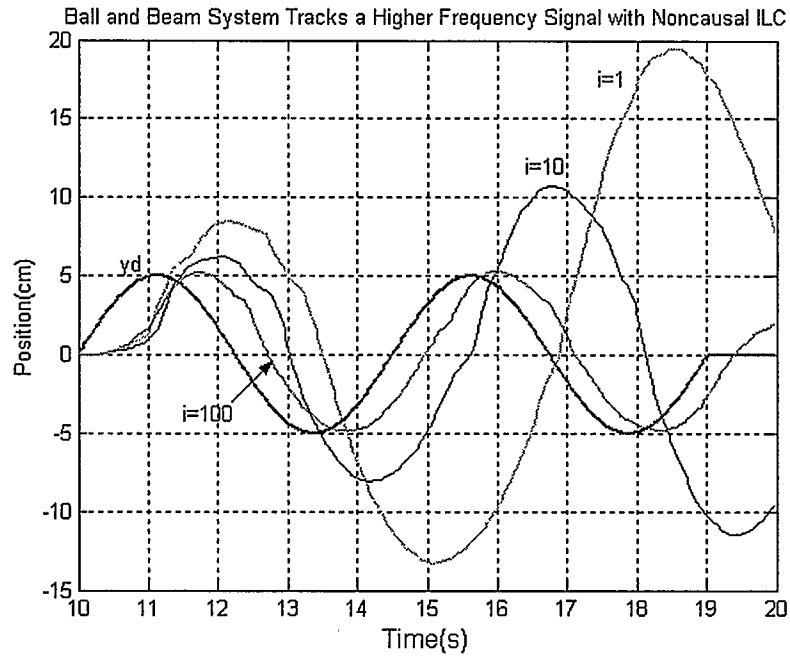


Figure 4.24: Tracking a Higher Frequency Signal with Noncausal ILC on Real Ball and Beam

4.6 Conclusions About Noncausal ILC

Recalling the simulations and experiments on the DC motor and a NMP plant in Chapter 3, and summarizing all the simulation and experiment results in this chapter, we construct Table 4.1.

Table 4.1: Simulation and Experiment Results Comparison

Verified Theory	Case	Simulation Result	Experiment Result
	$\text{NMP, } P(s) = \frac{s-1}{(s+1)^2}$ <p>with $F = 1$, rel.deg.=1</p>	Divergent (Fig. 3.4)	

Causal ILC	DC motor with $F = 1$, rel.deg.=2 $P(s) = \frac{1}{s(0.03s + 1)}$	Divergent (Fig. 3.7)	Divergent (Fig. 3.8)
	DC motor with $F \neq 1$, rel.deg.=2	Convergent (Fig. 3.9)	Convergent (Fig. 3.10)
	Ball and Beam with $F = 1$, rel.deg.=4 $P_0(s) = \frac{20.6}{(s+1)(s+2)(s+3)(s+30)}$	Divergent (Fig. 4.12)	Divergent (Fig. 4.13)
	Ball and Beam with $F \neq 1$, rel.deg.=4	Convergent (Fig. 4.14)	Convergent (Fig. 4.15)
Equivalent Feedback	DC motor with $F \neq 1$, rel.deg.=2	Convergent (Fig. 3.11)	Convergent (Fig. 3.12)
Control	Ball and Beam with $F \neq 1$, rel.deg.=4	Convergent (Fig. 4.17)	Convergent (Fig. 4.18)
Noncausal ILC	NMP, with $F = 1$, rel.deg.=1	Convergent (Fig. 4.4)	
Design	DC motor with $F = 1$, rel.deg.=2	Convergent (Fig. 4.5)	Convergent (Fig. 4.6)
	Ball and Beam with $F = 1$, rel.deg.=4	Convergent (Fig. 4.19)	Convergent (Fig. 4.20)

Noncausal ILC Tracking Higher Frequency Signal	Ball and Beam with $F = 1$, rel.deg.=4	Convergent, Phase delay is reduced. (Fig. 4.23)	Convergent, Phase delay is reduced. (Fig. 4.24)
---	--	--	--

Examining the results in Table 4.1, we can obtain some conclusions about noncausal ILC as follows:

- With the application of a noncausal symmetrical low-pass filter, the convergence condition of ILC is improved. For most LTI systems, even if a plant's relative degree is greater than one and/or it is unstable NMP, noncausal ILC can achieve zero ultimate tracking error.
- Simulations and experiments on the Ball and Beam system validate the theorems and Lemmas in Chapter 3 for higher order systems and higher relative degree systems. Noncausal ILC can achieve zero ultimate tracking error for these systems.
- When applying feedback control on a higher relative degree system tracking a higher frequency trajectory, a large phase delay exists in the output. Noncausal ILC eliminates this phase delay.
- ILC strongly depends on the same initial point throughout the experiment. An unsteady initial point will affect ILC performance.

Chapter 5

Robustness of Noncausal ILC for LTI Systems

Robustness is an important property for any control design. Many papers [e.g., 34-37] discuss a robust performance condition for various ILC designs. But most of them limit their discussions to causal operators. Lemma 2 in Chapter 3 shows that if P is NMP or has relative degree greater than one, it is impossible for the tracking error to converge to zero with causal ILC. Theorem 1 and Theorem 2 in Chapter 3 show that there exists an equivalent feedback control for any causal ILC. Several experiments in Chapter 4 prove that a noncausal ILC design as proposed in [29] can improve on causal ILC. Even if the process is NMP or has higher relative degree, noncausal ILC can make the system track the reference perfectly. This chapter will concentrate on robustness of noncausal ILC for LTI systems, including NMP systems.

5.1 Robust Performance Condition of Noncausal ILC

Suppose P is a stable or stabilized LTI system. We may model uncertainty in P as

$$P = P_0(1 + \Delta W_2), \quad (5.1)$$

where P_0 is a known stable mathematical model of system in transfer function form, W_2 is a known and stable transfer function representing the size of the plant uncertainty, and Δ is an unknown stable transfer function with a norm that satisfies , [42].

Since P is stable, we may set $C = 0$ in the ILC algorithm (3.4). Then $S = 1$ in (3.23). If we choose $F = 1$ to get $e_\infty = 0$, H in (3.22) becomes

$$H = 1 - DP, \quad (5.2)$$

where

$$D = LP_0^{-1} \quad (5.3)$$

to make D proper. According to Lemma 2, for most LTI systems, D must be a noncausal operator to achieve $e_\infty = 0$. Let L be a noncausal symmetric low-pass filter such as (4.5). Then,

$$0 < L(j\omega) \leq 1. \quad (5.4)$$

Lemma 3: *If $|W_2(j\omega)| < 1$, the system (3.24) converges to $e_\infty = 0$. [29]*

Proof: Substituting (5.1) and (5.3) into (5.2) gives

$$|H(j\omega)| = |1 - L(j\omega)(1 + \Delta(j\omega)W_2(j\omega))|. \quad (5.5)$$

Since $\|\Delta\|_\infty \leq 1$,

$$\begin{aligned} |H(j\omega)| &= |1 - L(j\omega) - L(j\omega)\Delta(j\omega)W_2(j\omega)| \\ &\leq 1 - L(j\omega) + L(j\omega)|W_2(j\omega)|. \end{aligned} \quad (5.6)$$

Since $|W_2(j\omega)| < 1$, and taking (5.4) into consideration, we get

$$|H(j\omega)| \leq 1 - L(j\omega)(1 - |W_2(j\omega)|) < 1, \quad (5.7)$$

which satisfies the convergence condition $|H(j\omega)| < 1$. Since $F = 1$, $e_\infty = 0$.

Q.E.D.

However if $|W_2(j\omega)| > 1$, then we need to set $F \neq 1$ to achieve convergence, which results in $e_\infty \neq 0$. Since P is stable, we can still choose $C = 0$ which gives $S = 1$. Then H in (3.22) becomes

$$H = F - DP, \quad (5.8)$$

where D is the same as it was in (5.3). Substituting (5.1) and (5.3) into (5.8) gives

$$H = F - L(1 + \Delta W_2). \quad (5.9)$$

Lemma 4: *If $F = L$ and $|L(j\omega)| < |W_2(j\omega)|^{-1}$, then the system (3.24) converges to*

$$e_\infty = \frac{1-L}{1+L\Delta W_2} y_d, [29].$$

Proof: Let $F = L$, where $|L(j\omega)| < |W_2(j\omega)|^{-1}$. Then

$$\begin{aligned} |H(j\omega)| &= |F(j\omega) - L(j\omega)(1 + \Delta(j\omega)W_2(j\omega))| \\ &= |F(j\omega) - L(j\omega) - L(j\omega)\Delta(j\omega)W_2(j\omega)| \\ &\leq |F(j\omega) - L(j\omega)| + |L(j\omega)W_2(j\omega)| \\ &= |L(j\omega)W_2(j\omega)| \\ &< 1, \end{aligned} \quad (5.10)$$

and thus the convergence condition of Lemma 1 is satisfied.

From (3.14) and (3.12), we have

$$e_\infty = \frac{1-F}{1-F+P(C+D)} y_d. \quad (5.11)$$

Substituting $C = 0$, $F = L$, (5.1), and (5.3) into (5.11) gives

$$e_\infty = \frac{1-L}{1+L\Delta W_2} y_d. \quad (5.12)$$

Q.E.D.

Remark: The choice $F = L$ minimizes the upper bound on $|H(j\omega)|$ in (5.10).

5.2 Case Study of the Robustness of Noncausal ILC

Suppose the plant we wish to control is as shown in Figure 5.1.

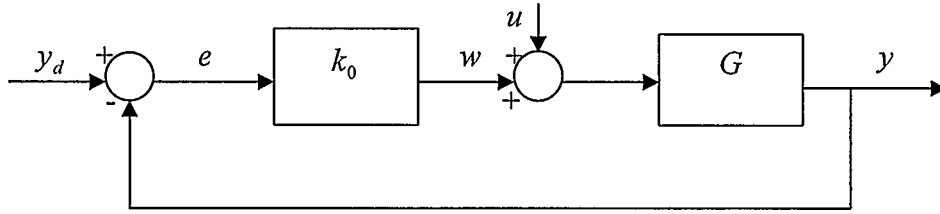


Figure 5.1: Plant

This plant can be expressed as

$$e = y_d - G(w + u), \quad (5.13)$$

where G is open loop system, w is feedback control,

$$w = k_0 e, \quad (5.14)$$

u is ILC input, and y_d is reference.

Substituting (5.14) into (5.13) gives

$$e = S y_d - S G u, \quad (5.15)$$

where

$$S = (1 + G k_0)^{-1}. \quad (5.16)$$

Thus we have

$$e = r - P u_i \quad (5.17)$$

where

$$r = Sy_d \quad (5.18)$$

is the reference of the closed-loop system, and

$$P = SG \quad (5.19)$$

is the closed-loop system.

In order to validate Lemma 3 and Lemma 4, simulations are presented for three kinds of perturbations.

5.2.1 Case 1: Uncertain Feedback Gain

5.2.1.1 Range of Gain for Convergence to Zero Error ($F = 1$)

Suppose the open-loop plant in Figure 5.1 is

$$G = \frac{1}{s(s+1)}. \quad (5.20)$$

Feedback control k_0 in (5.14) is designed to place the closed-loop poles at 45° from the real axis. By calculation, we get $k_0 = 0.5$. From (5.16) and (5.19), the closed-loop model of the plant is obtained as

$$P_0 = \frac{1}{s^2 + s + 0.5}. \quad (5.21)$$

However the real plant may be expressed as

$$P = \frac{1}{s^2 + s + k}, \quad (5.22)$$

where k varies and brings some uncertainty into plant. According to (5.1), the uncertainty is

$$\Delta W_2 = \frac{P}{P_0} - 1 = \frac{0.5 - k}{s^2 + s + k}. \quad (5.23)$$

According to Lemma 3, a sufficient condition for the system to achieve zero ultimate error is $|W_2(j\omega)| < 1$. Then we have

$$|\Delta(j\omega)W_2(j\omega)| = \left| \frac{0.5 - k}{s^2 + s + k} \right| < 1. \quad (5.24)$$

Substituting $s = j\omega$ into (5.24) gives

$$\left| \frac{0.5 - k}{\sqrt{\omega^2 + (k - \omega^2)^2}} \right| < 1. \quad (5.25)$$

Taking into account the fact that ω is real number and $\omega \geq 0$, solving (5.25) gives

$$0.25 \leq k \leq 1.7, \quad (5.26)$$

which is the convergence range for the gain with $e_\infty = 0$.

5.2.1.2 Simulation Results When $F = 1$ and k is within Range in (5.26)

Within the convergence range in (5.26), the system should be convergent. Since this result is derived with $F = 1$, the ultimate error will be $e_\infty = 0$. The Bode plot of ΔW_2 is shown in Figure 5.2 with $k = 0.3$ and L as described in (4.21) with $\omega_0 = 10$. We see that the magnitude $|\Delta(j\omega)W_2(j\omega)| < 1$ at all frequencies. By applying the noncausal ILC

$$u_i = u_{i-1} + De_{i-1}, \quad (5.27)$$

where $D = LP_0^{-1}$, we get the simulation results shown in Figure 5.3. Here, the reference y_d is $y_d = 0.5 - 0.5 \cos(\pi t)$. We see that the system converges to $e_\infty = 0$. Later we will use the same reference y_d in all simulations.

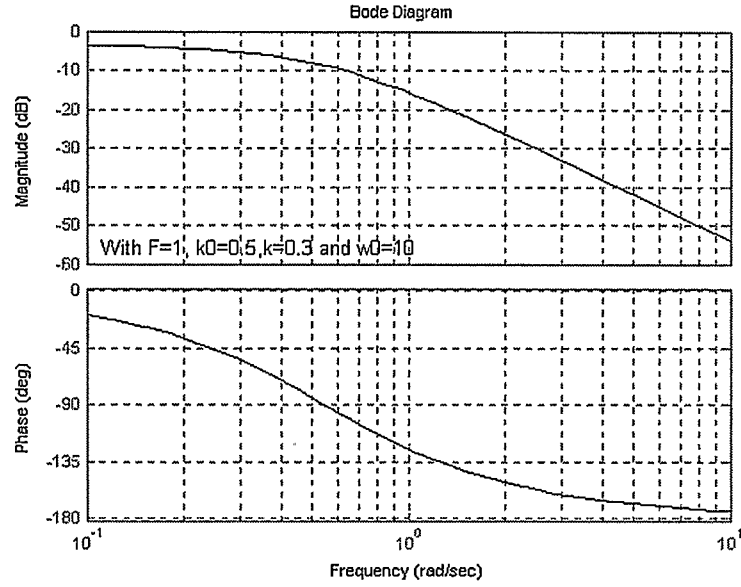


Figure 5.2: Bode Plot of ΔW_2 with $F = 1$ and $k = 0.3$

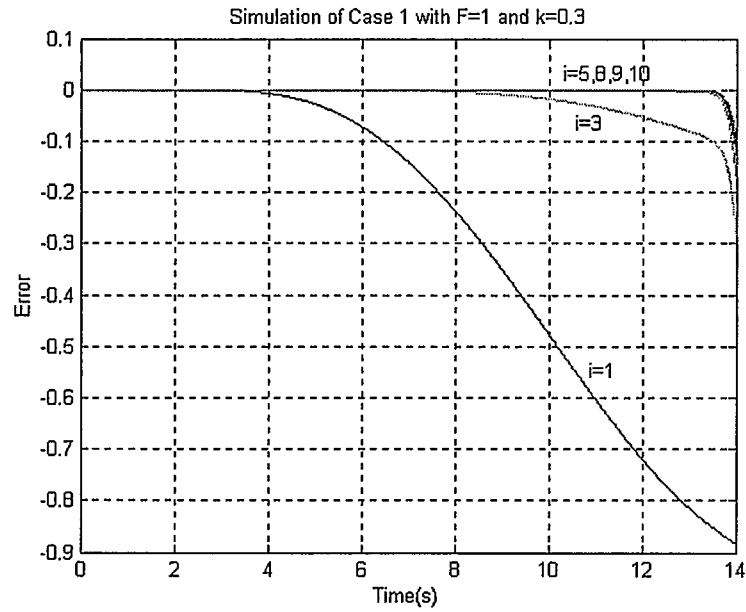


Figure 5.3: Simulation of Case 1 with $F = 1$ and $k = 0.3$

5.2.1.3 Simulation Results When $F = 1$ and k is Outside of Range in (5.26)

When k is increased to a value outside the convergence range (5.26), the system begins to diverge. This is because for k outside the convergence range $|\Delta(j\omega)W_2(j\omega)| > 1$. Figure 5.4 shows a Bode plot created using the same values of L and D stated previously, and a gain of $k = 10$. Figure 5.5 shows the divergence of this system.

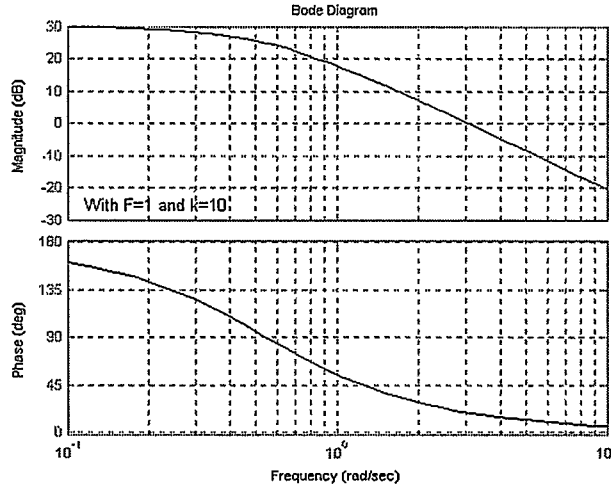


Figure 5.4: Bode Plot of ΔW_2 with $F = 1$ and $k = 10$

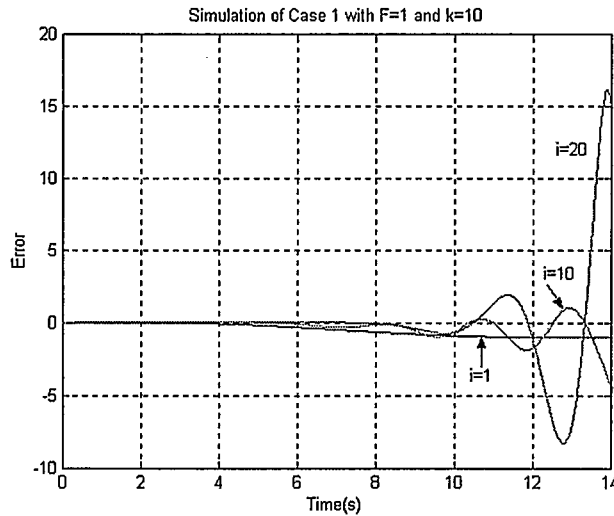


Figure 5.5: Simulation of Case 1 with $F = 1$ and $k = 10$

5.2.1.4 Convergence to Nonzero with k Outside of Range (5.26)

Suppose $k = 3$, which is outside of the range in (5.26). Since $|W_2(j\omega)| > 1$, we apply Lemma 4, which suggests choosing $F = L$ and $|L(j\omega)| < |\Delta(j\omega)W_2(j\omega)|^{-1}$ to

make the system converge. If we choose $L = \frac{1}{(1 - \frac{s}{\omega_0})^2 (1 + \frac{s}{\omega_0})^2}$ with $\omega_0 = 5$, the Bode

plot of $L(j\omega)$ and $[\Delta(j\omega)W_2(j\omega)]^{-1}$ is as shown in Figure 5.6, where the thick line represents $[\Delta(j\omega)W_2(j\omega)]^{-1}$. Applying noncausal ILC

$$u_i = Fu_{i-1} + De_{i-1}, \quad (5.28)$$

where $D = LP_0^{-1}$, the system converges as shown in Figure 5.7, although $e_\infty \neq 0$. The value of e_∞ is determined by (5.12) as follows. For the specific values of all operators in this example, we have

$$R(s) = \frac{s^2(s - 7.071)(s + 7.071)(s^2 + s + 3)}{(s - 5.591)(s - 4.116)(s + 5.68)(s + 3.771)(s^2 + 1.255s + 0.6339)}. \quad (5.29)$$

Taking partial fraction expansion of (5.29) gives

$$R(s) = 1 + \frac{3.1168}{s + 5.68} - \frac{3.8547}{s - 5.591} + \frac{5.1989}{s - 4.116} - \frac{4.7962}{s + 3.771} + \frac{0.3362s + 0.1881}{s^2 + 1.255s + 0.6339}. \quad (5.30)$$

The two-sided inverse Laplace transform of (5.30) gives

$$r(t) = \delta(t) + 3.1168e^{-5.68t} - 4.7962e^{-3.771t} + 0.3362e^{-0.6275t} \cos(0.49t) - 0.0467e^{-0.6275t} \sin(0.49t) + (5.1989e^{4.116t} - 3.8547e^{5.591t})h(-t). \quad (5.31)$$

Convoluting of $r(t)$ and $y_d(t)$ gives e_∞ as shown in Figure 5.7.

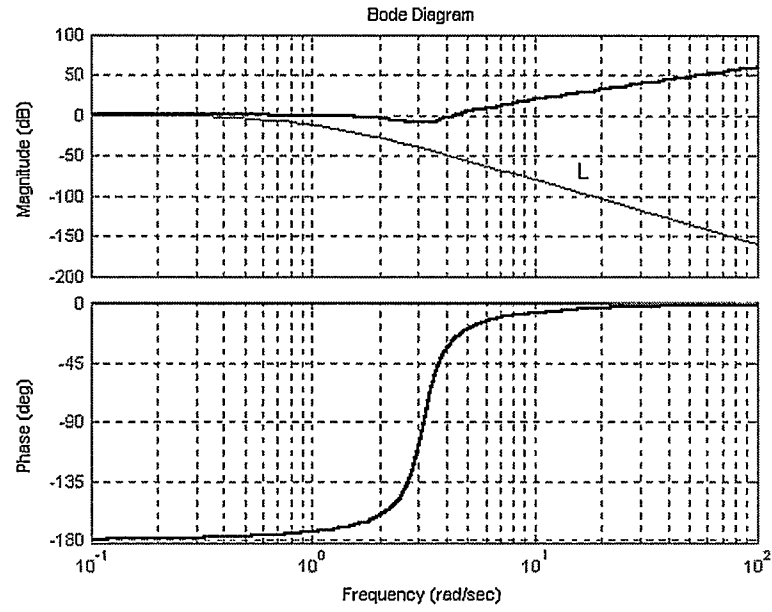


Figure 5.6: Bode Plot of $(\Delta W_z)^{-1}$ and L with $F = L$ and $k = 3$

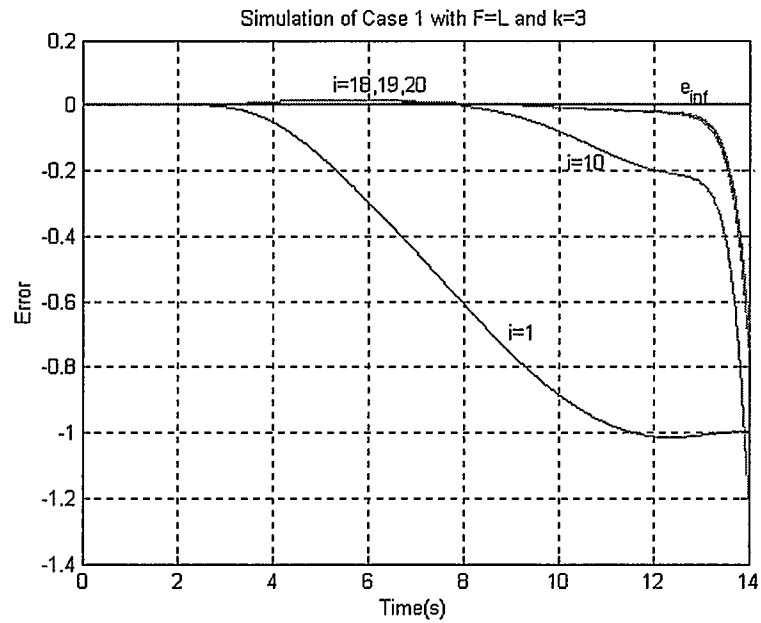


Figure 5.7: Simulation of Case 1 with $F = L$ and $k = 3$

5.2.2 Case 2: Uncertain Parameter in Plant Model

5.2.2.1 Range of Parameter for Convergence to Zero Error

In this section, we discuss another possible parameter variation in the plant. Suppose the real plant is

$$P = \frac{1}{s^2 + \alpha s + k_0}, \quad (5.32)$$

while the mathematical model is still

$$P_0 = \frac{1}{s^2 + s + 0.5}. \quad (5.33)$$

According to (5.1), the uncertainty is

$$\Delta W_2 = \frac{P}{P_0} - 1 = \frac{(1-\alpha)s}{s^2 + \alpha s + k_0}, \quad (5.34)$$

and its magnitude is

$$|\Delta(j\omega)W_2(j\omega)| = \left| \frac{(1-\alpha)j\omega}{j\alpha\omega + k_0 - \omega^2} \right| = \left| \frac{(1-\alpha)\omega}{\sqrt{(\alpha\omega)^2 + (k_0 - \omega^2)^2}} \right|. \quad (5.35)$$

If

$$\left| \frac{(1-\alpha)\omega}{\sqrt{(\alpha\omega)^2 + (k_0 - \omega^2)^2}} \right| < 1, \quad (5.36)$$

the system will converge to $e_\infty = 0$. Since

$$\left| \frac{(1-\alpha)\omega}{\sqrt{(\alpha\omega)^2 + (k_0 - \omega^2)^2}} \right| = \left| \frac{(1-\alpha)\omega}{\sqrt{(1-\alpha)^2\omega^2 + (k_0 - \omega^2)^2 - \omega^2 + 2\alpha\omega^2}} \right|,$$

if
$$Q = (k_0 - \omega^2)^2 - \omega^2 + 2\alpha\omega^2 > 0, \quad (5.37)$$

(5.36) will be true. Substituting $k_0 = 0.5$ into (5.37) and rearranging gives

$$Q = \omega^4 + 2(\alpha - 1)\omega^2 + 0.25. \quad (5.38)$$

So if $\alpha > 1$, (5.37) is satisfied, and thus (5.36) is satisfied as well.

5.2.2.2 Simulation Results When $F = 1$ and $\alpha = 20$

When $\alpha = 20$, which is much greater than 1, Figure 5.8 shows a Bode plot of ΔW_2 . Applying the noncausal ILC indicated in (5.27) with L as in (4.21) and $\omega_0 = 10$, after several iterations, we get the simulation results shown in Figure 5.9, which indicates that the system converges to $e_\infty = 0$.

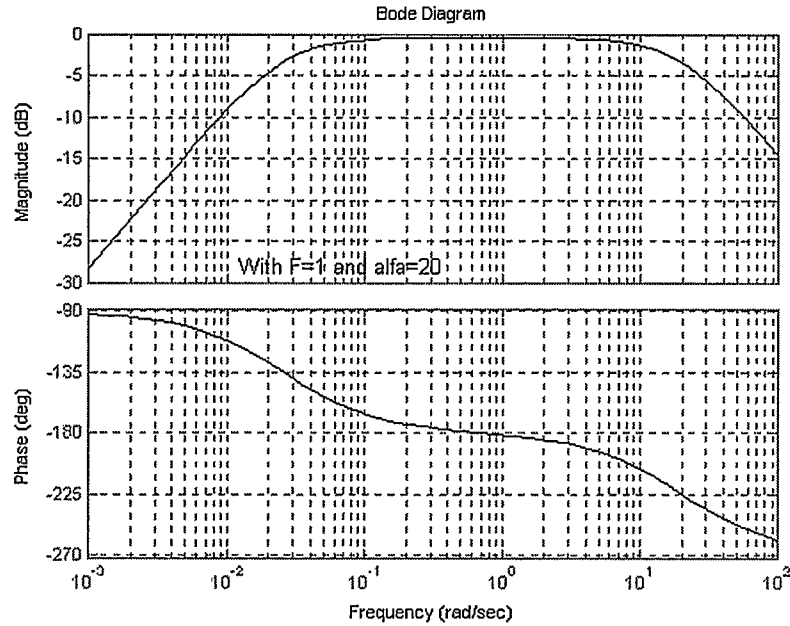


Figure 5.8: Bode Plot of ΔW_2 with $F = 1$ and $\alpha = 20$

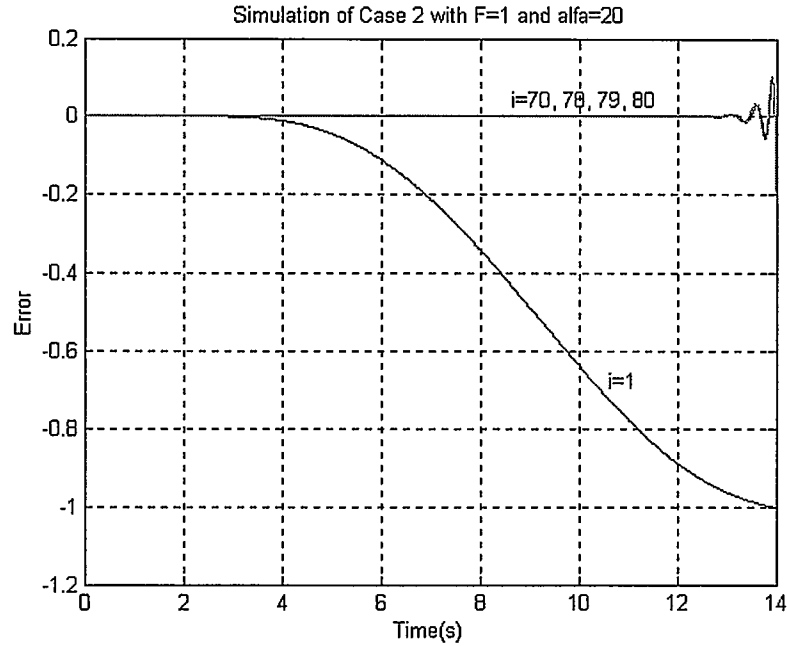


Figure 5.9: Simulation of Case 2 with $F = 1$ and $\alpha = 20$

5.2.2.3 Simulation Results when $F = 1$ and $\alpha = 0.5$

When α is out of the convergence range, for example, $\alpha = 0.5$, with the same ILC law as in 5.2.2.2 but $\omega_0 = 2$ in (4.21), the system diverges as in Figure 5.11. This is because the magnitude of ΔW_2 is greater than 1 at some frequencies, as shown in Figure 5.10.

5.2.2.4 Convergence to Nonzero Error when $\alpha = 0.5$

Since $|W_2(j\omega)| > 1$, we choose $F = L$. For this example, with $\alpha = 0.5$, we take $\omega_0 = 2$ in (4.21). Figure 5.12 shows a Bode plot of $|L(j\omega)| < |\Delta(j\omega)W_2(j\omega)|^{-1}$. Applying the ILC law as shown in (5.28), the system converges to fixed point with

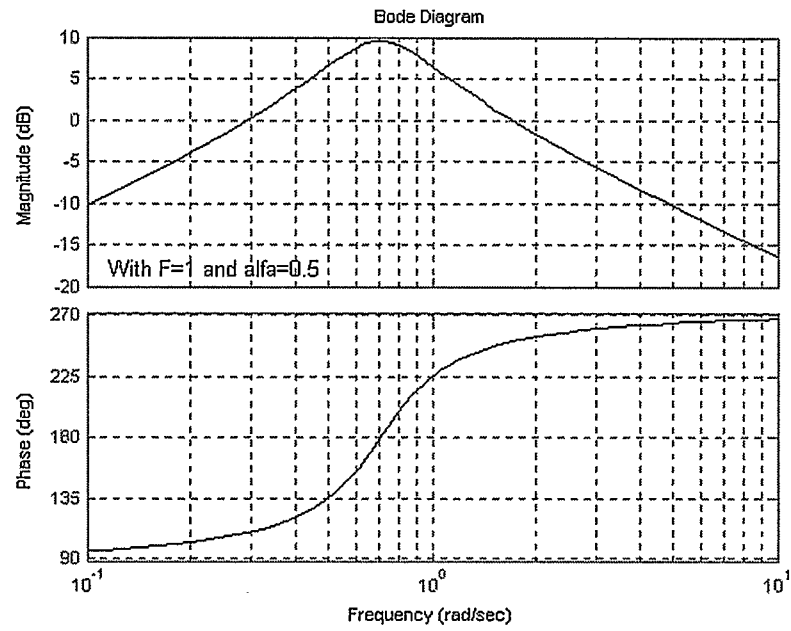


Figure 5.10: Bode Plot of ΔW_2 with $F = 1$ and $\alpha = 0.5$

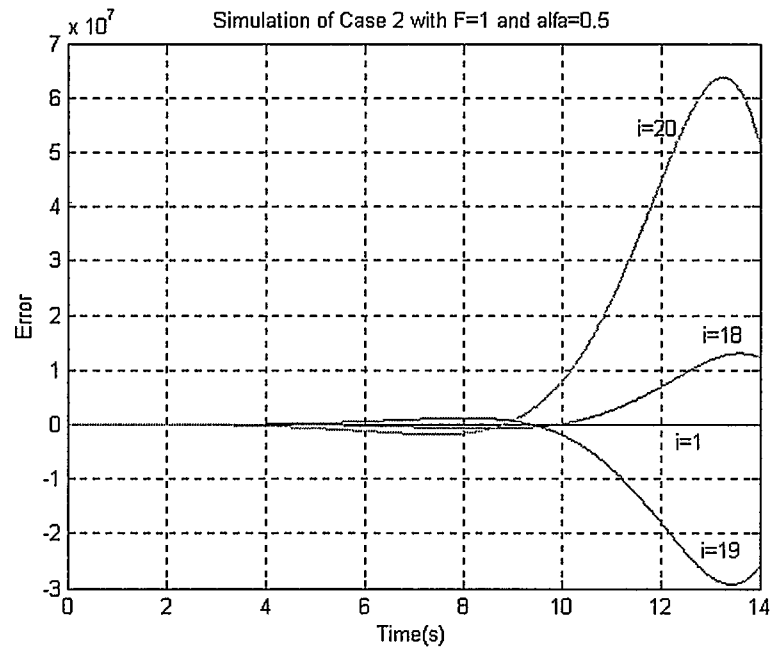


Figure 5.11: Simulation of Case 2 with $F = 1$ and $\alpha = 0.5$

$e_\infty \neq 0$, as is determined by (5.12). For the specific values of all operators in this example, we have

$$R(s) = \frac{s^2(s-2.828)(s+2.828)(s^2+0.5s+0.5)}{(s+2.436)(s+1.125)(s+5.68)(s^2-4.134s+4.438)(s^2+1.074s+0.6581)} \quad (5.39)$$

Taking partial fraction expansion of (5.39) gives

$$R(s) = 1 + \frac{0.5994}{s+2.436} - \frac{1.0554}{s-1.125} + \frac{-0.08s+0.1143}{s^2+1.074s+0.6582} + \frac{0.5352s-2.137}{s^2-4.134s+4.438} \quad (5.40)$$

Using two-sided Laplace transform, in the time domain R can be written as

$$\begin{aligned} r(t) = & \delta(t) + 0.5994e^{-2.436t} - 1.0554e^{-1.125t} - 0.08e^{-0.537t} \cos(0.6081t) \\ & + 0.2586e^{-0.537t} \sin(0.6081t) + [0.5352e^{2.067t} \cos(0.4068t) - 2.5338e^{2.067t} \sin(0.4068t)]h(-t) \end{aligned} \quad (5.41)$$

Convoluting of $r(t)$ and $y_d(t)$ gives e_∞ as shown in Figure 5.13.

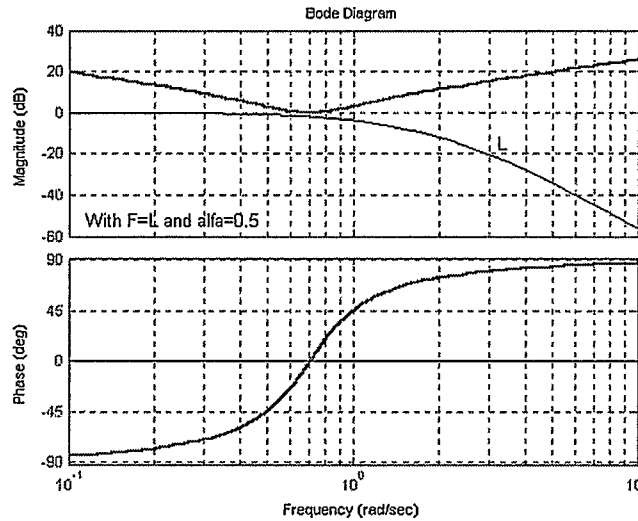


Figure 5.12: Bode Plot of $(\Delta W_2)^{-1}$ and L with $F = L$ and $\alpha = 0.5$

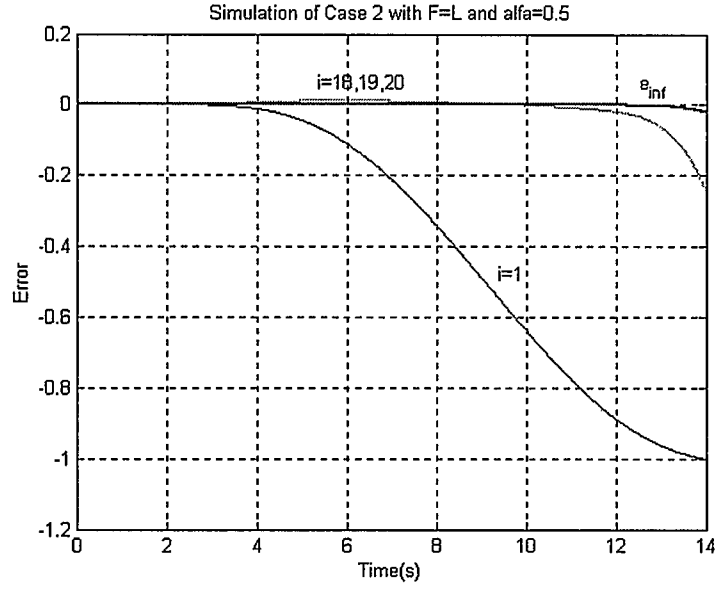


Figure 5.13: Simulation of Case 2 with $F = L$ and $\alpha = 0.5$

5.2.3 Case 3: Multiplication of Plant by Low-pass Filter

5.2.3.1 Range of Cut-off Frequency for Convergence to Zero Error

The third possible uncertainty is brought into plant by P_1 in the form of

$$P = P_0 P_1, \quad (5.42)$$

where

$$P_0 = \frac{1}{s^2 + s + 0.5}, \quad (5.43)$$

and

$$P_1 = \frac{1}{\frac{s}{\omega_u} + 1} = \frac{\omega_u}{s + \omega_u}. \quad (5.44)$$

The uncertainty can be written as

$$\Delta W_2 = \frac{P}{P_0} - 1 = \frac{-\frac{s}{\omega_u}}{\frac{s}{\omega_u} + 1} = \frac{-s}{s + \omega_u}. \quad (5.45)$$

So

$$|\Delta(j\omega)W_2(j\omega)| = \left| \frac{\omega}{\sqrt{\omega^2 + \omega_u^2}} \right| < 1, \quad (5.46)$$

for all $\omega_u \neq 0$.

5.2.3.2 Simulation Result

With $\omega_u = 5$ as an example, the Bode plot for ΔW_2 is as shown in Figure 5.14. Applying

(5.27), and (4.21) with $\omega_0 = 10$, the system converges to $e_\infty = 0$ very quickly, as shown

in Figure 5.15.

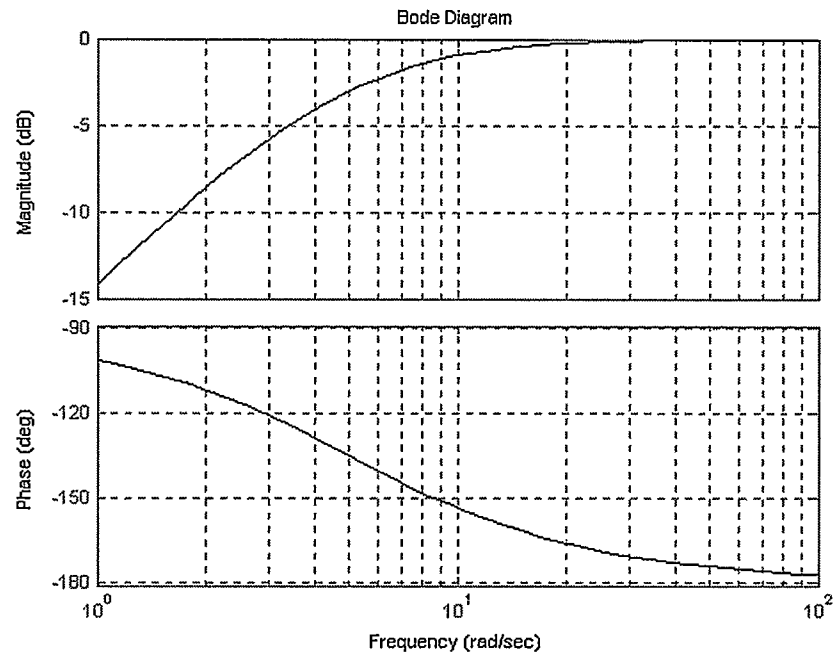


Figure 5.14: Bode Plot of ΔW_2 with $F = 1$ and $\omega_n = 5$

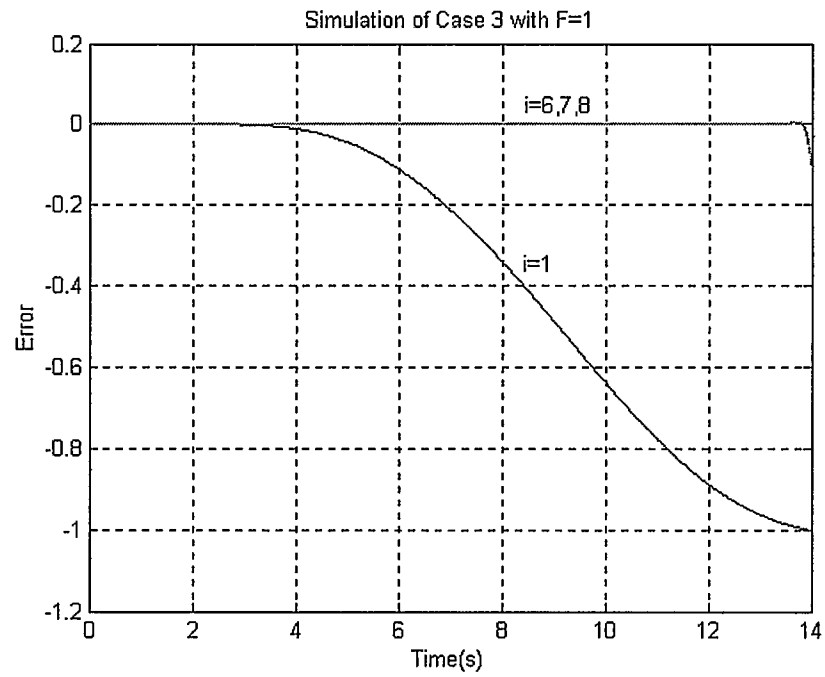


Figure 5.15: Simulation of Case 3 with $F = 1$ and $\omega_n = 5$

5.3 Summary

In this chapter, we discussed robust convergence condition of noncausal ILC. Lemma 3 states that if $F = 1$ and $|W_2(j\omega)| < 1$, the system is convergent and the ultimate tracking error will be $e_\infty = 0$. Lemma 4 shows that if $|W_2(j\omega)| > 1$, we need to choose $F = L$ and $|L(j\omega)| < |W_2(j\omega)|^{-1}$ to make the system converge, and the ultimate tracking error will be $e_\infty = \frac{1-L}{1+L\Delta W_2} y_d$. Lemma 3 and Lemma 4 are applicable to a process of any order and any relative degree. To verify Lemma 3 and Lemma 4, we selected three possible uncertainties that may exist in the plant and simulated them. Table 5.1 gives an outline of the results. For Case 3, the situation of $|W_2(j\omega)| > 1$ does not occur. From Table 5.1, we can see that in all three cases simulation results obey Lemmas 3 and 4.

Table 5.1: Verification Results of Lemma 3 and Lemma 4

Verification of Lemma	Robust Condition	Perturbation	Simulation Results
Lemma 3 ($F = 1 \Leftrightarrow e_\infty = 0$)	$ W_2(j\omega) < 1$	Gain	Convergent (Fig. 5.3)
		Coefficient of first order item in denominator of P	Convergent (Fig. 5.9)

	$ W_2(j\omega) > 1$	Adding a low-pass filter	Convergent (Fig. 5.15)
		Gain	Divergent (Fig. 5.5)
		Coefficient of first order item in denominator of P	Divergent (Fig. 5.11)
Lemma 4 $(F = L \neq 1 \Rightarrow e_\infty \neq 0)$	$ W_2(j\omega) > 1$, but $ L(j\omega) < W_2(j\omega) ^{-1}$	Gain	Convergent (Fig. 5.7)
		Coefficient of first order item in denominator of P	Convergent (Fig. 5.13)

Chapter 6

Summary and Conclusions

6.1 Summary

This thesis reviewed some recent theoretical results in ILC and validated them via simulation and experiment. These results are namely the equivalence of causal ILC and feedback control, the limitations of causal ILC when zero tracking error is required, and the improvement in performance and robustness provided by noncausal ILC.

Since ILC was proposed in the 1980's, most ILC algorithms and designs have been based on causal operators. Recent research shows that causal ILC cannot achieve zero ultimate error for processes that are NMP or have relative degree greater than one and that there exists an equivalent feedback control for any causal ILC [8]. The experiments and simulations presented in Chapter 3 demonstrate these performance limitations of causal ILC and validate this equivalence result.

The results in Chapter 3 tell us that there is no reason to use causal ILC since equivalent feedback control can achieve the same ultimate tracking error without iterations. To improve causal ILC, a noncausal ILC design is proposed in [29] which guarantees that the convergence condition $|H(j\omega)| < 1$ is satisfied for LTI systems of any relative degree, including NMP plants. In Chapter 4, simulations and experiments on a DC motor (with relative degree of 2) and a Ball and Beam system (with relative degree of 4) validate the feedback equivalence result for higher relative degree processes and

demonstrate that noncausal ILC can achieve zero ultimate error for such systems with relative degree greater than one.

In Chapter 5, we investigated the robustness of this noncausal ILC design. Lemma 3 states that if $|W_2(j\omega)| < 1$, the ILC system converges robustly to $e_\infty = 0$, while Lemma 4 states that if $|W_2(j\omega)| > 1$ and $|L(j\omega)| < |W_2(j\omega)|^{-1}$, the system converges robustly to $e_\infty = \frac{1-L}{1+L\Delta W_2} y_d$. These lemmas were validated via simulation for three different perturbations in the system model.

6.2 Conclusions

Based on our analysis, simulation and experimental verification, we can conclude the following points regarding ILC:

- Causal ILC has limitations for NMP processes and those plants with relative degree greater than one. Zero ultimate tracking error is unachievable for these processes. If the system converges, $e_\infty \neq 0$.
- An Equivalent Feedback Control exists for any causal LTI ILC algorithm. This equivalent feedback control design depends only on the causal operators in the causal ILC algorithm or design. No more information is required. This equivalent feedback control can achieve the same tracking accuracy of causal ILC in just one trial.
- Causal ILC has no advantages over feedback control because equivalent feedback control can achieve the same accuracy as causal ILC without iteration while the causal ILC may need a large number of trials to achieve this accuracy.

- Noncausal ILC can improve on causal ILC. Even if a plant's relative degree is greater than one and/or it is an unstable NMP plant, noncausal ILC can converge to zero tracking error. A significant improvement in tracking performance occurs when tracking a higher frequency trajectory, since the phase delay is drastically reduced (ideally, to zero).
- Noncausal ILC has very good robustness properties. When $|W_2(j\omega)| < 1$, the system is convergent and ultimate tracking error is $e_\infty = 0$, even if the system is of higher relative degree or NMP. If $|W_2(j\omega)| > 1$, we only need to select $F = L$ and $|L(j\omega)| < |W_2(j\omega)|^{-1}$ to make the system convergent, and the ultimate tracking error is $e_\infty = \frac{1-L}{1+L\Delta W_2} y_d$.
- Our theoretical results are obtained in the frequency domain with an infinite frequency interval, while our simulations and experiments are implemented in the time domain with a truncated time interval. So the simulation and experimental results are the approximation of the theoretical results.
- Simulation results are slightly different from their corresponding experimental results because the mathematical models we applied in simulations are approximations to the real systems. Uncertainty in the real systems also affects the experimental results.
- Since it is difficult in practice to maintain constant initial conditions, even if feedback control is applied, an extra error may be brought into the system performance in practice.

6.3 Recommendations for Future Work

To continue this research in future, some recommendations are given as follows:

- The value of ω_0 in the filter L is an important factor that influences the convergence of noncausal ILC and the value of e_∞ . It would be useful to develop a design approach for calculating values of ω_0 for optimal performance.
- An experiment on an NMP plant, such as inverted pendulum, to validate the effectiveness of noncausal ILC on a real NMP system.
- An experimental investigation of the robustness of noncausal ILC.
- An investigation of the truncation effects that occur from implementing an ILC designed in the frequency domain on a finite time interval.
- An investigation of the influence of varying initial conditions on noncausal ILC, and the development of a solution to this problem.

Bibliography

- [1] Kevin L. Moore, Iterative Learning Control for Deterministic Systems. Springer-Verlag, Landon, 1993.
- [2] Kevin L. Moore, Iterative Learning Control: An Expository Overview. Applied and Computational Controls, Signal Processing, and Circuits, 1:151-214.1999.
- [3] M.Uchiyama, "Formation of high speed motion pattern of mechanical arm by trial", Transactions of the society of Instrumentation and control Engineers, Vol.19, pp. 706-712, May, 1978.
- [4] S. Arimoto, Sadao Kawamura and Fumio Miyazaki, Bettering Operation of Robots by Learning. J. Robotic Systems, 1(2): 123-140, 1984.
- [5] S.Saab, On the P-type Learning Control, IEEE Trans. Automatic Control, 39(11): 2298-2302, 1994.
- [6] S. Arimoto, S. kawamura, F. Miyazaki, and S. Tamaki, "Learning Control Theory for dynamical Systems", In Proc. 24th conf. Decis. Cont., Ft. Lauderdale, FL, Dec, 1985, pp.1375-1380.
- [7] H. Hashimoto and J. X. Xu. "Learning control systems with feedback. In Proceedings of the IEEE Asian Electronics Conference, Hong Kong, September 1987.
- [8] Peter B. Goldsmith. On the equivalence of causal LTI iterative learning control and feedback control. Automatica, 38, 703-708, 2002.

- [9] N. Amann, D. H. Owens, and E. Rogers. Iterative learning control using optimal feedback and feedforward actions. *International Journal of Control*, 65(2):277-293, September 1996.
- [10] Y. Chen, J. X. Xu, and T. H. Lee. Feedback-assisted high-order iterative learning control of uncertain nonlinear discrete-time systems. In *Proceedings of the International Conference on Control, Automation, Robotics, and Vision*, Singapore, December 1996.
- [11] Yangquan Chen, Jian-Xin Xu, and Tong Heng Lee. Current iteration tracking error assisted higher order iterative learning control of discrete-time uncertain nonlinear systems. In *Proceedings of the 2nd Asian Control Conference*, Seoul, Korea, July 1997.
- [12] Peter B. Goldsmith, *Stability, Convergence, and Feedback Equivalence of LTI Iterative Learning Control*.
- [13] Chong-ho Choi and Tae-Jeong Jang, Iterative Learning Control for a general Class of Nonlinear Feedback systems. *Proceedings of the American Control Conference*, Seattle, Washington. June 1995.
- [14] Z. Bien and J. -X. Xu. *Iterative Learning Control: Analysis, Design, Integration and Application*. Kluwer Academic Publishers, 1998.
- [15] D. H. Hwang, Z. Bien, and S. R. Oh. Iterative learning control method for discrete-time dynamic systems. In *IEEE Proceedings Part D, Control Theory and applications*, volume 138, pages 139-144, March 1991.
- [16] C. Chen. A discrete iterative learning control for a class of nonlinear time-varying systems. *IEEE Trans. Automatic Control*, 43(5):748-752, 1998.

- [17] N. Amann and D. H. Owens. Non-minimum phase plants in iterative learning control. In Second International Conference on Intelligent System Engineering, pages 107-112, Hamburg-Harburg, Germany, September 1994.
- [18] L. Hideg and R. Judd. Frequency domain analysis of learning systems. In Proceedings of the 27th Conference on Decision and Control, pages 586-591, Austin, Texas, December 1998.
- [19] R. P. Judd, R. P. Van Til, and L. Hideg. Equivalent Lyapunov and frequency domain stability conditions for iterative learning control systems. In Proceedings of 8th IEEE International Symposium on Intelligent Control, pages 487-492, 1993.
- [20] L. M. Hideg. Stability and convergence issues in iterative learning control. In Proceedings Intelligent Engineering Systems Through Artificial Neural Networks in Engineering Conference, volume 4, pages 211-216, Louis, MO, November 1994.
- [21] T. Sogo and N. Adachi. Convergence rates and robustness of iterative learning control. In Proceedings of 35th IEEE Conference on Decision and Control, volume 3, pages 3050-3055, Kobe, Japan, December 1996.
- [22] S. Kawamura, F. Miyazaki, and S. Arimoto. Convergence, stability and robustness of learning control schemes for robot manipulators. In Proceedings of the International Symposium on Robotic Manipulators: Modelling, Control, and Education, Albuquerque, New Mexico, November 1986.
- [23] K. L. Moore, M. Dahleh, and S. P. Bhattacharyya. Adaptive gain adjustment for a learning control method for robotics. In Proceedings of 1990 IEEE International Conference on Robotics and Automation, Cincinnati, Ohio, May 1990.

- [24] D. de Roover. Synthesis of a robust iterative learning controller using an H_∞ approach. In Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan, December 1996.
- [25] Tae-Yong Doh, Jung-Hoo Moon, Kyung Bog Jin, and Myung Jin Chung. An iterative learning control for uncertain systems using structured singular value. In Proceedings of the 2nd Asia Control Conference, Seoul, Korea, July 1997.
- [26] Yangquan Chen, Jian-Xin Xu, and Tong Heng Lee. Current iterative tracking error assisted iterative learning control of uncertain nonlinear discrete-time systems. In Proceedings of the 35th IEEE Conference on Decision and Control, volume3, pages 3038-3043, Kobe, Japan, December 1996.
- [27] J. -X. Xu and Z. Qu. Robust learning control for a class of non-linear systems. In Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan, December 1996.
- [28] Peter B. Goldsmith. The Fallacy of Causal Iterative Learning Control. In: IEEE Conference on Decision and Control. Orlando, FL. Pp4475-4480. 2001.
- [29] Peter B. Goldsmith. Noncausal Iterative Learning Control for Uncertain LTI systems. ICAR03, Coimbra, Portugal, June 30 –July 3, 2003.
- [30] György Fodor. Laplace Transforms in Engineering. Publishing House of the Hungarian Academy of Sciences. Budapest, 1965.
- [31] Peter B. Goldsmith. The General Equivalence of Causal Iterative Learning Control and Feedback Control, internal report, University of Calgary, 2002.

- [32] M. Verwoerd, G. Meinsma, and T. J. A. de Vries. On the use of noncausal lti operators in iterative learning control. In Proc. IEEE Conference on Decision and Control, pages 3362-3366, Las Vegas, Nevada, 2002.
- [33] Héctor G. Chiacchiarini, and Pablo S. Mandolesi. Unbalance Compensation for Active Magnetic Bearings using ILC. Proceedings of the 2001 IEEE International Conference on Control Applications. September 5-7, 2001, Mexico City, Mexico.
- [34] A. Tayebi and M. B. Zaremba. Robust Iterative Learning Control Design is Straightforward for Uncertain LTI Systems Satisfying the Robust Performance Condition. IEEE Transactions on Automatic Control, Vol. 48. No. 1, January 2003.
- [35] Tae-Yong Doh, Jung-Hoo Moon, Kyung Bog Jin, and Myung Jin Chung. An Iterative Learning Control for Uncertain Systems Using Structured Singular value. In Proceedings of the 2nd Asian Control Conference, Seoul, Korea, July 1997.
- [36] D. de Roover. Synthesis of a Robust Iterative Learning Controller Using an H_∞ Approach. In Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan, December 1996.
- [37] J. -X. Xu and Z. Qu. Robust Learning Control for a Class of Non-linear Systems. In Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan, December 1996.
- [38] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. Feedback Control of Dynamic Systems. Prentice Hall, New Jersey, 2002.
- [39] Ming Xia, and Peter B. Goldsmith. Noncausal ILC Applied to a DC Motor. submitted to CDC, 2004.

- [40] Quanser Consulting Inc. A Comprehensive and Modular Laboratory for Control Systems Design and Implementation. 1997.
- [41] Control Tutorials for MATLAB.
<http://www.engin.umich.edu/group/ctm/examples/ball/ball.html>
- [42] John C. Doyle, Bruce A. Francis, and Allen R. Tannenbaum. Feedback Control Theory. Macmillan Publishing Company. New York, 1992.
- [43] Quanser Consulting Inc. WinCon 3.2 Manual. 1997
- [44] The MathWorks, Inc. Simulink—Dynamic System Simulation for Matlab. Version 4. Natick, MA, November, 2000.
- [45] Duane Hanselman, Bruce Littlefield. Mastering Matlab 6—A Comprehensive Tutorial and Reference. Prentice Hall, Inc. Upper Saddle River, New Jersey, 2001.
- [46] Hugh F. Vanlandingham. Introduction to Digital Control Systems. Macmillan Publishing Company, New York, 1985.
- [47] Thomas Kailath. Linear Systems. Prentice Hall Information and System Science Series. 1979.

Appendix A

A1: Code of Experiments on Real Systems (Including GUI)

```
function varargout = moto_noncau(varargin)

% MOTO_NONCAU Application M-file for moto_noncau.fig
% FIG = MOTO_NONCAU launch moto_noncau GUI.
% MOTO_NONCAU('callback_name', ...) invoke the named callback.
% THIS CODE IS FOR DC MOTOR EXPERIMENTS.

if nargin == 0 % LAUNCH GUI
    fig = openfig(mfilename,'reuse');
    % Use system color scheme for figure:
    set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);
    if nargin > 0
        varargout{1} = fig;
    end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
    try
        if (nargout)
            [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        else
            feval(varargin{:}); % FEVAL switchyard
        end
    catch
        disp(lasterr);
    end
end

% -----
%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
```



```

%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback. You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks. Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
%| <MFILENAME>(<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.

% -----
function varargout = figure1_CreateFcn(h, eventdata, handles, varargin)

wc_run; % run WinCon Server

% -----
function varargout = figure1_CloseRequestFcn(h, eventdata, handles, varargin)

closereq; % close the GUI

%-----
function varargout = startstop_Callback(h, eventdata, handles, varargin)

if get(handles.startstop,'Value') % button pressed down
    set(handles.startstop,'BackgroundColor',[1 0 0]);
    set(handles.startstop,'String','STOP');

    wc_start;
else
    wc_stop;
    set(handles.startstop,'BackgroundColor',[0 0.9 0]);
    set(handles.startstop,'String','START');
    wc_saveplot('Scope - mo_noncau\Scope1', 'error.mat'); %previous error.
    wc_saveplot('Scope - mo_noncau\Scope3', 'u0.mat'); %u0 is previous input.
    load error.mat;
    load u0.mat; % load it to workspace???
    short_time=find(plot_time<=6.0001);
    error=mo_noncau_Scope1(1:length(short_time));
    u0=mo_noncau_Scope3(1:length(short_time));

    %computer learning operator D=L*P0^-1.
    k=1;
    w_0=1;
    n=length(short_time);
    T=plot_time(1:n);
    t=T';

```

```

dt=t(2)-t(1);

%The following lines are the calculation of  $D=L*P0^{-1}$ , where L is a noncausal operator.
%This is noncausal convergence case, where  $F=1$ , D is noncausal and  $e_{\infty}=0$ .
D=learn(k,w_0,n);
du=dt*conv(D,error);
du=du(n:2*n-1);

%These following lines are the calculation of  $D=L*P0^{-1}$ , where L is a causal operator.
%This is causal divergence case, where  $F=1$  and D is causal.
    %d0=0.08;
    %D2=2.3467*exp(-2*t) - 5.0133*prod([t;exp(-2*t)],1); %causal  $D=L*(P0.^{-1})$ 
    %du=dt*conv(D2,error);
    %du=du(1:n);
    %du=d0*error+du;

%These following lines are the calculation of  $D=L*P0^{-1}$ , where L is a causal operator.
%This is causal convergence case, where  $F=L$  and D is causal and  $e_{\infty}$  is non-zero.
    %d0=0.08;
    %D2=2.3467*exp(-2*t) - 5.0133*prod([t;exp(-2*t)],1); %causal  $D=F(P0.^{-1})$ 
    %du=dt*conv(D2,error);
    %du=du(1:n); %du is n*1 vector
    %du=d0*error+du;
    %F=L=4*prod([t;exp(-2*t)],1);
    %ui=dt*conv(F,u0);
    %ui=ui(1:n);

newsig=u0+du; % This law is for noncausal ILC, where  $F=1$  and D is noncausal.  $ui=u0+D*ei$ .

%newsig=u0+du; % This law is for causal divergence case, where  $F=1$  and D is causal.  $ui=u0+D*ei$ .

%newsig=ui+du; % This law is for causal convergence case, where  $F=L$  and D is causal.  $ui=F*u0+D*ei$ .

newsig2=newsig;
plot2=plot_time(1:length(short_time));
strp2=num2str(plot2');
strsig2=num2str(newsig2');
set_param('mo_noncau/Repeating Sequence', 'rep_seq_t', [' ', strp2, ''], 'rep_seq_y', [' ', strsig2, '']);

end

%THE FOLLOWING CODE IS FOR THE EXPERIMENTS OF BALL AND BEAM SYSTEM
%if get(handles.startstop,'Value') % button pressed down
    % set(handles.startstop,'BackgroundColor',[1 0 0]);
    %set(handles.startstop,'String','STOP');
    %wc_start;
%else
    %wc_stop;
    %set(handles.startstop,'BackgroundColor',[0 0.9 0]);
    %set(handles.startstop,'String','START');
    %wc_saveplot('Scope - mo_noncau\Scope1', 'error.mat'); %previous error.
    %wc_saveplot('Scope - mo_noncau\Scope3', 'u0.mat'); %u0 is previous input.

```

```

%load error.mat;
%load u0.mat; % load it to workspace.
%short_time=find(plot_time<=20.0001);% find the number of sample points.
%n=length(short_time);
%m=floor(0.5*n);
%error=mo_noncau_Scope1((m+1):2*m);
%u0=mo_noncau_Scope3((m+1):2*m);
%w_0=0.005;

%T=plot_time(1:m);
%t=T';
%dt=t(2)-t(1);

%The following 4 lines are the calculation of  $D=L*P0^{-1}$  for noncausal ILC, where L is a noncausal operator.
%This is noncausal convergence case, where  $F=1$ , D is noncausal and  $e_{inf}=0$ .
%Dc=w_0.^4*[-359990000*exp(-w_0*t) + 1820000*prod([t;exp(-w_0*t)],1)]; %causal,w0=0.005
%Da=w_0.^4*[359990000*exp(-w_0*t) + 1780000*prod([t;exp(-w_0*t)],1)]; %anticausal,w0=0.005
%D=[la(m:-1:2),lc]; %reverse and concatenate Da and Dc;  $D=L*P0^{-1}$ 
%d0=w_0.^4; % constant item of D
%du=dt*conv(D,error);
%du=du(m:2*m-1);

%These following lines are the calculation of  $D=L*P0^{-1}$  for causal ILC,where L is a causal operator.
%This is causal divergence case, where  $F=1$  and D is causal.
%d0=16/20.6;
%D1=tf([28 111 82],[1 6 12 8]);
%D=d0*D1;
%d = dt*impulse(D,t);
%du=dt*conv(d,error);
%du=du(1:m); % for causal

%These following lines are the calculation of  $D=L*P0^{-1}$  for causal ILC,where L is a causal operator.
%This is causal convergence case, where  $F=L$  and D is causal.
%d0=16/20.6;
%D1=tf([28 111 82],[1 6 12 8]);
%D=d0*D1;
%F=L=tf([1], [0.0625 0.5000 1.5000 2.0000 1.0000]);
%d = dt*impulse(D,t);
%f=dt*impulse(F,t);
%du=dt*conv(d,error);
%du=du(1:m); % for causal
%df=dt*conv(f,u0);
%df=df(1:m);

%newsig=u0+D0*error+du'; % This law is for noncausal convergence case, where  $F=L$  and D is
% noncausal.  $ui=u0+D*ei$ .

%newsig=u0+D0*error+du'; % This law is for causal divergence case, where  $F=1$  and D is causal.
%  $ui=u0+D*ei$ .

%newsig=df+d0*error+du; %This law is for causal convergence case, where  $F=L$  and D is causal.
%  $ui=F*u0+D*ei$ .

```

```

% newsig2 = [0*t, newsig']; % newsig2 is n dimation.
%plot2=plot_time(1:2*m);
%strp2=num2str(plot2');
%strsig2=num2str(newsig2);
%set_param('mo_noncau/Repeating Sequence', 'rep_seq_t', [' ',strp2, ''], 'rep_seq_y', [' ',
%strsig2, '']);

%end

guidata(h,handles);

% -----
function D=learn(k,w_0,n)

%Apply learning operator L to inoput x.
%Inverse nominal C.L. plat  $P=1/(s^2 + s + k)$ 
% and applies real 4th-order filter with bandwith= $w_0$ .
%sample time dt assumed.

dt=.01; % sample time
t_max=(n-1)*dt;%max time
t=0:dt:t_max;

%coefficients of  $l(t)$ =impulse response of L:

c=.25*w_0*(k-w_0^2);
cc=.25*w_0^2*(w_0^2-w_0+k);
ca=0.25*w_0^2*(w_0^2+w_0+k);
lc=c*exp(-w_0*t) + cc*prod([t;exp(-w_0*t)],1); %causal
la=c*exp(-w_0*t) + ca*prod([t;exp(-w_0*t)],1); %anticausal
D=[la(n:-1:2),lc]; %reverse and concatenate la and lc

% -----
function F = filter4(w_0,n)

% Impulse response of real 4th order filter.
% Sample time dt assumed.
dt = .01; % sample time
t_max = (n-1)*dt; % max time
t = 0:dt:t_max; % time vector
fc = .25*w_0*exp(-w_0*t) + .25*w_0^2*(t.*exp(-w_0*t)); % causal part
F = dt*[fc(n:-1:2), fc]; %reverse and concatenate to add noncausal part

%-----
function varargout = checkbox_F_Callback(h, eventdata, handles, varargin)

if get(handles.checkbox_F,'Value') % checked
    set(handles.F_num,'Enable','On');
    set(handles.F_den,'Enable','On');
else
    set(handles.checkbox_F,'String','0');

```

```

        set(handles.F_num,'Enable','Off');
        set(handles.F_den,'Enable','Off');
    end

    guidata(h,handles);

% -----
function varargout = F_num_Callback(h, eventdata, handles, varargin)

NewNum = get(handles.F_num,'String');
if check(NewNum)
    F_numer=NewNum;
else
    set(handles.F_num,'String','0');
end

    guidata(h,handles);

% -----
function varargout = F_den_Callback(h, eventdata, handles, varargin)

NewDen = get(handles.F_den,'String');
if check(NewDen)
    F_deno=NewNum;
else

    set(handles.F_den,'String','0');
end

    guidata(h,handles);

% -----
function varargout = checkbox_C_Callback(h, eventdata, handles, varargin)

if get(handles.checkbox_C,'Value') % checked
    set(handles.C_num,'Enable','On');
    set(handles.C_den,'Enable','On');
else
    set(handles.checkbox_C,'String','0');
    set(handles.C_num,'Enable','Off');
    set(handles.C_den,'Enable','Off');
end

    guidata(h,handles);

% -----
function varargout = C_num_Callback(h, eventdata, handles, varargin)

NewNum = get(handles.C_num,'String');
if check(NewNum)
    C_numer=NewNum;

```

```

else
    set(handles.C_num,'String','0');
end

guidata(h,handles);

% -----
function varargout = C_den_Callback(h, eventdata, handles, varargin)

NewDen = get(handles.C_den,'String');
if check(NewDen)
    C_deno=NewNum;
else
    set(handles.C_den,'String','0');
end

guidata(h,handles);

% -----
function varargout = checkbox_D_Callback(h, eventdata, handles, varargin)

if get(handles.checkbox_D,'Value') % checked
    set(handles.D_num,'Enable','On');
    set(handles.D_den,'Enable','On');
else
    set(handles.checkbox_D,'String','0');
    set(handles.D_num,'Enable','Off');
    set(handles.D_den,'Enable','Off');
end

guidata(h,handles);

% -----
function varargout = D_num_Callback(h, eventdata, handles, varargin)

NewNum = get(handles.D_num,'String');
if check(NewNum)
    D_numer=NewNum;
else
    set(handles.D_num,'String','0');
end

guidata(h,handles);

% -----
function varargout = D_den_Callback(h, eventdata, handles, varargin)

NewDen = get(handles.D_den,'String');
if check(NewDen)
    D_deno=NewNum;
else

```

```

    set(handles.D_den,'String','0');
end

guidata(h,handles);

% -----
function varargout = checkbox_I_Callback(h, eventdata, handles, varargin)

if get(handles.checkbox_I,'Value') % checked
    set(handles.Value_I,'Enable','On');
    set(handles.slider_I,'Enable','On');
else
    set(handles.checkbox_I,'String','0');
    set(handles.Value_I,'Enable','Off');
    set(handles.slider_I,'Enable','Off');
end

guidata(h,handles);

% -----
function varargout = Value_I_Callback(h, eventdata, handles, varargin)

NewStrVal = get(handles.Value_I,'String');
Max = get(handles.slider_I,'Max');
Min = get(handles.slider_I,'Min');
NewVal = str2double(NewStrVal);

if isnan(NewVal)
    OldVal = get(handles.slider_I,'Value');
    set(handles.Value_I,'String',num2str(OldVal));
    return
elseif (NewVal > Max)
    NewVal = Max;
elseif (NewVal < Min)
    NewVal = Min;
end
set(handles.Value_I,'String',num2str(NewVal));
set(handles.slider_I,'Value',NewVal);% set I-gain in Simulink model
set_param('mo_noncau/PID Controller/K2','Gain',num2str(NewVal));

guidata(h,handles);

% -----
function varargout = slider_I_Callback(h, eventdata, handles, varargin)

NewVal = get(handles.slider_I,'Value');
set(handles.Value_I,'String',num2str(NewVal)); % set I-gain in Simulink model
set_param('mo_noncau/PID Controller/K2','Gain',num2str(NewVal));

guidata(h,handles);

```

```

% -----
function varargout = checkbox_Der_Callback(h, eventdata, handles, varargin)

if get(handles.checkbox_Der,'Value') % checked
    set(handles.Value_Der,'Enable','On');
    set(handles.slider_Der,'Enable','On');
else
    set(handles.checkbox_Der,'String','0');
    set(handles.Value_Der,'Enable','Off');
    set(handles.slider_Der,'Enable','Off');
end

guidata(h,handles);

% -----
function varargout = Value_Der_Callback(h, eventdata, handles, varargin)

NewStrVal = get(handles.Value_Der,'String');
Max = get(handles.slider_Der,'Max');
Min = get(handles.slider_Der,'Min');
NewVal = str2double(NewStrVal);
if isnan(NewVal)
    OldVal = get(handles.slider_Der,'Value');
    set(handles.Value_Der,'String',num2str(OldVal));
    return
elseif (NewVal > Max)
    NewVal = Max;
elseif (NewVal < Min)
    NewVal = Min;
end
set(handles.Value_Der,'String',num2str(NewVal));
set(handles.slider_Der,'Value',NewVal); %set I-gain in Simulink model
set_param('mo_noncau/PID Controller/K1','Gain',num2str(NewVal));

guidata(h,handles);

% -----
function varargout = slider_Der_Callback(h, eventdata, handles, varargin)

NewVal = get(handles.slider_Der,'Value');
set(handles.Value_Der,'String',num2str(NewVal)); % set I-gain in Simulink model
set_param('mo_noncau/PID Controller/K1','Gain',num2str(NewVal));

guidata(h,handles);

% -----
function status = check(string)

% checks if STRING is in default MATLAB notation for numerator or denominator of
% transfer function.
% STATUS is 1 if all is well, 0 otherwise.
status = 1;

```



```

% check the first and last character that are not whitespace
n = length(string);
i0 = 1;
j0 = 0;
first = 0;
last = 0;
for i = 1:n
    if ~first % the first character has not been found yet
        if isspace(string(i))
            i0 = i0 + 1;
        else
            if (string(i) ~= '[')
                status = 0;
                return
            end
            first = 1; % checked first non-whitespace character
        end
    end
    if ~last % the last character has not been found yet
        if isspace(string(n+1-i))
            j0 = j0 + 1;
        else
            if (string(n+1-i) ~= ']')
                status = 0;
                return
            end
            last = 1; % checked last non-whitespace character
        end
    end
    if (first & last)
        break % break out of the FOR loop
    end
end

k = 0; % previous character is white space if k = 0
checkstring = ""; % initialise checkstring
for i = (1+i0):(length(string)-j0-1) % check rest of the string
    if isspace(string(i))
        if k % previous character was not whitespace
            if (isnan(str2double(checkstring)) | isinf(str2double(checkstring)))
                status = 0;
                return
            end
            k = 0;
        end
    else
        if k % previous character was not whitespace, add string(i) to checkstring
            checkstring = strcat(checkstring,string(i));
        else
            checkstring = string(i);
            k = 1;
        end
    end
end
end
if (isnan(str2double(checkstring)) | isinf(str2double(checkstring)))

```

```

    status = 0;
end

% -----
function varargout = Time_edit_Callback(h, eventdata, handles, varargin)

NewStrVal=get(handles.Time_edit,'String');
NewVal=str2double(NewStrVal);
if isempty(NewVal) | (NewVal<0)
    set(h, 'String', '00');
else
    TimeVal=NewVal;
    set_param('mo_noncau/Constant','value',num2str(TimeVal));
end

guidata(h,handles);

% -----
function varargout = Number_edit_Callback(h, eventdata, handles, varargin)

NSVal_n=get(handles.Number_edit,'String');
NVal_n=str2num(NSVal_n);
if isempty(NVal_n)|(NVal_n<0)
    set(handles.Number_edit, 'String', '00');
else
    NumVal=NVal_n;
end

guidata(h,handles);

% -----
function varargout = Initialise_Callback(h, eventdata, handles, varargin)

if (exist('mo_noncau','file')~=4)
    errordlg('The Simulink model cannot be found.',...
        'Error opening "mo_noncau.mdl","modal");
    return
else
    open_system('mo_noncau.mdl');
    if (exist('mo_noncau.wcl','file')~=2)
        wc_build('mo_noncau');
    else
        wc_download('mo_noncau');
    end
    wc_openplot('mo_noncau');
end

set(handles.startstop,'Enable','on');
set(handles.Time_edit,'Enable','on');
set(handles.Number_edit,'Enable','on');
set(handles.checkbox_I,'Enable','on');
checkboxbox_I_Callback(h, eventdata, handles, varargin);

```

```

set(handles.checkbox_Der,'Enable','on');
checkbox_Der_Callback(h, eventdata, handles, varargin);
set(handles.checkbox_F,'Enable','on');
checkbox_F_Callback(h, eventdata, handles, varargin);
set(handles.checkbox_C,'Enable','on');
checkbox_C_Callback(h, eventdata, handles, varargin);
set(handles.checkbox_D,'Enable','on');
checkbox_D_Callback(h, eventdata, handles, varargin);
set(handles.popupmenu2,'Enable','on');
popupmenu2_Callback(h, eventdata, handles, varargin);
guidata(h,handles);

% -----
function varargout = popupmenu2_Callback(h, eventdata, handles, varargin)

NewVal = get(handles.popupmenu2,'Value');
switch NewVal
case 1
    set_param('mo_noncau/G1','gain','1');
    set_param('mo_noncau/G2','gain','0');
case 2
    set_param('mo_noncau/G2','gain','1');
    set_param('mo_noncau/G1','gain','0');
end

guidata(h,handles);

```

A.2: Code of Figure 3.4

```

%Figure 3.4 is produces by this code.
clear;
P=tf([1 -1],[1 2 1]);
Yd=tf([1],[1 2 1]);
dt=0.01;
t1=0:dt:20;
p=impz(P,t1);
yd=impz(Yd,t1);
u=exp(t1);
y = dt*conv(p,u); %y = Pu
y((length(t1)+1):length(y)) = []; % right truncate
plot(t1,yd,t1,y), grid on;

```

A3: Code of Figure 3.5

```

% Figure 3.5: ILC in time domain: P=1/s+1, C=0, D=1, F=1
clear
dt = .01; % time interval
t=0:dt:pi;
yd=1+sin(0.1*t-pi/2);
p = exp(-t);
ui = 0*t; %initial input
ei = yd; %initial error
for i = 1:20, %ILC iterations

```

```

    ui = ui + ei;
    y = dt*conv(p,ui); %y = Pu
    y((length(t)+1):length(y)) = []; % right truncate
    yi(i,:)=y;
    ei = yd - y;
    u(i,:) = ui;
    e(i,:) = ei;
end
plot(t,yi([1 2 5 10 20],:));grid on

```

A4: Code of Figure 3.6

```

%This code produces Figure 3.6
%P=1/(s+1)^2
clear;
D=0;
n=50;
m=30;

for k=1:m
    u(1,k)=0;
    t(k)=k*0.1;
    e(1,k)=0;
    y_d(k)=0.5+0.5*sin(0.1*k-pi/2);
    P(k)=exp(-t(k));% P=1/(s+1)
    C(k)=1;
    F(k)=1;
end

for i=1:n
    for k=1:m

        y(i,k)=P(k)*u(i,k);

        e(i+1,k)=y_d(k)-y(i,k); % add i in the first colonm

        u(i+1,k)=F(k)*u(i,k)+C(k)*e(i+1,k)+D*e(i,k); % e(i+1,k)is current error.
    end
end

% Equivalent Feedback control
for k=1:m
    q=5000; % q is a gain constant
    K(k)=C(k)+q*(C(k)+D);
    G(k)=1+P(k)*K(k);
    M(k)=G(k).^(-1);
    E(k)=M(k)*y_d(k);
    U(k)=K(k)*E(k);
    Y(k)=P(k)*U(k);
end

plot(t,y_d,t,y(10,:),t,Y),grid on;

```

A5: Code of Figure 3.7

```

% Figure 3.7: DC motor model:  $G=1.5/s(0.03s+1)$ 
%  $D=F(P)^{-1}$ ,  $F=1/(0.5s+1)^2$ .
clear
t_s = 0; % start of stroke (end of backswing)
t_f = 5; % end of stroke (start of follow through)
t_max = 6; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
ui = 0*t; %initial input
ei = yd; %initial error
ei_1=yd;

for k=1:n

    t(k)=(k-1)*0.01;

    p(k)=1.5-1.5*exp(-33.33*t(k)); %  $P=1.5/s(0.03s+1)$ ;

    D2(k)=2.3467*exp(-2*t(k)) - 5.0133*t(k)*exp(-2*t(k)); %  $D=F(P)^{-1}$ 

end

C=0.0005;
D1=0.08;
F=1;

%i=1
du = dt*conv(D2,ei_1);
du((length(t)+1):length(du)) = []; % right truncate
ui=ui+D1*ei_1+C*ei+du; % ILC control law
y = dt*conv(p,ui); %y = Pu
y((length(t)+1):length(y)) = []; % right truncate
ei = yd - y;
Y(1,:)=y;
u(1,:)=ui;
e(1,:)=ei;

%i=2
du = dt*conv(D2,ei_1);
du((length(t)+1):length(du)) = []; % right truncate
ui=ui+D1*ei_1+C*e(1,:)+du;% ILC control law
y = dt*conv(p,ui); %y = Pu
y((length(t)+1):length(y)) = []; % right truncate
ei = yd - y;
Y(2,:)=y;
u(2,:)=ui;
e(2,:)=ei;

for i = 3:50, %ILC iterations
    du = dt*conv(D2,e(i-2,:));

```

```

du((length(t)+1):length(du)) = []; % right truncate
ui=ui+D1*e(i-2,:)+C*e(i-1,:)+du;% ILC control law
y = dt*conv(p,ui); %y = Pu
y((length(t)+1):length(y)) = []; % right truncate
ei = yd - y;
Y(i,:)=y;
u(i,:) = ui;
e(i,:) = ei;
end

```

```

plot(t,yd,t,Y([10 20 30],:));grid on

```

A6: Code of Figure 3.9

```

% This code produces Figure 3.9: DC motor model:  $G=1.5/s(0.03s+1)$ 
%  $D=F(P)^{-1}$ ,  $F=1/(.5s+1)^2$ ,
clear
t_s = 0; % start of stroke (end of backswing)
t_f = 5; % end of stroke (start of follow through)
t_max = 6; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
ui = 0*t; %initial input
ei = yd; %initial error
ei_1=yd;

for k=1:n
    t(k)=(k-1)*0.01;
    p(k)=1.5-1.5*exp(-33.33*t(k)); %  $P=1.5/s(0.03s+1)$ ;
    F(k)=4*t(k)*exp(-2*t(k)); %  $F=1/((0.5*s+1)^2)$ ;
    D2(k)=2.3467*exp(-2*t(k)) - 5.0133*t(k)*exp(-2*t(k)); %  $D=F(P)^{-1}$ 
end

C=0.0005;
D1=0.08;

%i=1
ui = dt*conv(F,ui) + dt*conv(D2,ei_1);
ui((length(t)+1):length(ui)) = []; % right truncate
ui=ui+D1*ei_1+C*ei;% ILC control law
y = dt*conv(p,ui); %y = Pu
y((length(t)+1):length(y)) = []; % right truncate
ei = yd - y;
Y(1,:)=y;
u(1,:) = ui;
e(1,:) = ei;

%i=2
ui = dt*conv(F,ui) + dt*conv(D2,ei_1);
ui((length(t)+1):length(ui)) = []; % right truncate
ui=ui+D1*ei_1+C*e(1,:);% ILC control law

```

```

y = dt*conv(p,ui); %y = Pu
y((length(t)+1):length(y)) = []; % right truncate
ei = yd - y;
Y(2,:)=y;
u(2,:) = ui;
e(2,:) = ei;
for i = 3:50, %ILC iterations
    ui = dt*conv(F,ui) + dt*conv(D2,e(i-2,:));
    ui((length(t)+1):length(ui)) = []; % right truncate
    ui=ui+D1*e(i-2,:)+C*e(i-1,:);% ILC control law
    y = dt*conv(p,ui); %y = Pu
    y((length(t)+1):length(y)) = []; % right truncate
    ei = yd - y;
    Y(i,:)=y;
    u(i,:) = ui;
    e(i,:) = ei;
end
plot(t,yd,t,Y([ 10 15 20 30 35],:));grid on

```

A7: Code of Figure 3.11

```

% This code produces Figure 3.11: DC motor model:P=1.5/s(0.03s+1)
% D=F(P).^-1, F=1/(.5s+1).^2,
clear;
% It is assumed that all time markers divide evenly by dt:
t_s = 0; % start of stroke (end of backswing)
t_f = 19; % end of stroke (start of follow through)
t_max = 20; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
P = tf([1.5],[0.03 1 0]);% Ballbeam closed-loop tranfer fuction.
L = tf([1 ],[0.25 1 1]);
F=L;
D=L*P^-1;
C=1
K=[(1-F)^-1]*(C+D);
M=(1+P*K)^-1;
tsh = dt:dt:t_max+dt; % time shifted forward by dt
m = dt*impulse(M,tsh);
e=dt*conv(m,yd);
e=e(1:n);
y=yd-e;
plot(t,y,t,yd);grid on

```

A8: Code of Figure 4.4

```

%This code produces Figure 4.4: Noncausal ILC on NMP plant
clear;
P=tf([1 -1],[1 2 1]);
Yd=tf([1], [1 2 1]);

```

```

dt=0.01;
t1=0:dt:20

for k=1001:-1:1 % Initializing
    t2(k)=-(k-1)*0.01;
    u0(k)=-exp(t2(k));
    %u2(k)=-exp(t2(k));
end

for k=1:1001
    u2(k)=u0(1001-k+1);
    yd(k)=0;
end

p(1)=1;
p1(1001)=1;

for k=2:1001
    t3(k)=(k-1)*0.01;
    u2(1000+k)=0;
    p(k)=exp(-t3(k))-2*t3(k)*exp(-t3(k));
    p(1000+k)=0;
    yd(1000+k)=t3(k)*exp(-t3(k));
    p1(1000+k)=p(k);
end

ui=u2;

for i = 1:50, %ILC iterations
    y = dt*conv(p,ui); %y = Pu
    y((length(t1)+1):length(y)) = []; % right truncate
    ei = yd - y;
    Y(i,:)=y;
    u(i,:)= ui;
    e(i,:)= ei;
    ui=ui+0.01*ei;
end

plot(t1,yd,t1,Y([50],:));grid on

```

A9: Code of Figure 4.5

```

% This code produces Figure 4.5, where DC motor model:  $G = 1.5/(0.03s^2 + s)$ ,
%  $r = Sy_d$ ,  $u_i = u_{i-1} + D \cdot e_{i-1}$ ,
%  $D$  is composed of  $P^{-1}$  and a real 4th-order filter  $F = 1/[(s/w_0 + 1)^2((s/w_0 - 1)^2]$ 
% with bandwidth =  $w_0$ .

```

```

clear;
k = 2; % feedback gain in C.L. plant
w_0 = 10; % bandwidth of learning filter

```

```

% It is assumed that all time markers divide evenly by dt:
t_s = 0; % start of stroke (end of backswing)
t_f = 5; % end of stroke (start of follow through)
t_max = 6; % end of trial

```



```

dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
G = tf([1.5],[0.03 1 0]);
S = (1+k*G)^(-1);
P = S*G;
tsh = dt:dt:t_max+dt; % time shifted forward by dt
s = dt*impulse(S,tsh);
p = dt*impulse(P,tsh);
r = conv(s,yd); %closed-loop reference
r = r(1:n); % truncate
ui = 0; % initial input
ei = r; %initial error
e(1,:) = ei; % plotting array
y(1,:)=yd+ei;
f = filter4(w_0,n); % compute filter used by D

for i = 2:30,
    du = conv(f,ei); % filter the error
    du = [du(n:2*n-1),zeros(1,n-1)]; % truncate and pad with zeros
    du = deconv(du,p); % apply P inverse
    ui = ui + du; % update control input
    yi = conv(p,ui); % y = Pu
    yi = yi(1:n); % truncate
    ei = r - yi; % error
    e(i,:) = ei; % store error for plotting
    y(i,:)=yd+ei;
end

plot(t,y);grid on

```

A10: Code of Figure 4.12

```

% Apply Causal ILC on BB. F=1, divergent case.
%Apply a 4-order causal filter on it.
% u_i = u_(i-1) + De_(i-1)
% D is a causal operator.
% P is changed to have better poles placement. pole=[-30 -3 -2 -1].
clear;
w_0 = 2;

% It is assumed that all time markers divide evenly by dt:
t_s = 0; % start of stroke (end of backswing)
t_f = 19; % end of stroke (start of follow through)
t_max = 20; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max

```

```

P = tf([20.6],[1 36 191 336 180]);% Ballbeam closed-loop tranfer fuction.
D = tf([16],[1 8 24 32 16]);
F=1;
d0=16/20.6;
D1=tf([28 111 82],[1 6 12 8]); % a causal operator
D=d0*D1;
tsh = dt:dt:t_max+dt; % time shifted forward by dt
d = dt*impulse(D,tsh);
p = dt*impulse(P,tsh);
ui = 0*t; % initial input
ei=yd;
e(1,:)= ei; % plotting array

for i = 2:200,
    du=dt*conv(d,ei);
    du=du(n:2*n-1);
    ui = ui + (d0*ei + du); % update control input: ui=Fu(i-1)+De(i-1).
    yi = conv(p,ui); % y = Pu
    yi = yi(1:n); % truncate
    ei=yd-yi;
    e(i,:)= ei; % store error for plotting
    y(i,:)=yi;
end

plot(t,y([ 50 100 150 200],:),t,yd);grid on

```

A11: Code of Figure 4.14

```

% This code produces Figure 4.14. Apply a 4-order causal filter on it.
% u_i = Fu_(i-1) + De_(i-1)
% D is composed of P^-1 and a real 4th-order filter L.
% P is changed to have better poles placement. pole=[-30 -3 -2 -1].
clear;
% It is assumed that all time markers divide evenly by dt:
t_s = 0; % start of stroke (end of backswing)
t_f = 19; % end of stroke (start of follow through)
t_max = 20; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
P = tf([20.6],[1 36 191 336 180]);% Ballbeam closed-loop tranfer fuction.
L = tf([110 ],[0.0625 0.5 1.5 2 1]);
F=L;
D=L*P^-1;
tsh = dt:dt:t_max+dt; % time shifted forward by dt
d = dt*impulse(D,tsh);
p = dt*impulse(P,tsh);
f = dt*impulse(F,tsh);
ui = 0*t; % initial input
ei=yd;
e(1,:)= ei; % plotting array

```

```

for i = 2:20,

    du=dt*conv(d,ei);
    du=du(1:n);
    du1=dt*conv(f,ui);
    du1=du1(1:n);
    ui = du1 + du; % update control input: ui=Fu(i-1)+De(i-1).
    yi = conv(p,ui); % y = Pu
    yi = yi(1:n); % truncate
    ei=yd-yi;
    e(i,:)=ei; % store error for plotting
    y(i,:)=yi;
end

plot(t,y([ 1 2 3 10 18 20],:),t,yd);grid on

```

A12: Code of Figure 4.17

```

% This code produces Figure 4.17. Apply Equivalent Feedback Control on BB. F is not equal to 1.
clear;
% It is assumed that all time markers divide evenly by dt:
t_s = 0; % start of stroke (end of backswing)
t_f = 19; % end of stroke (start of follow through)
t_max = 20; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
P = tf([20.6],[1 36 191 336 180]); % Ballbeam closed-loop transfer function.
L = tf([1 10 ],[0.0625 0.5 1.5 2 1]);
F=L;
D=L*P^-1;
C=0;
K=[(1-F)^-1]*(C+D);
M=(1+P*K)^-1;
tsh = dt:dt:t_max+dt; % time shifted forward by dt
m = dt*impulse(M,tsh);
e=dt*conv(m,yd);
e=e(1:n);
y=yd-e;
plot(t,y,t,yd);grid on

```

A13: Code of Figure 4.19

```

% This code produces Figure 4.19. Apply Noncausal ILC to BB.
clear;
w_0 = 0.01; % bandwidth of learning filter.

% It is assumed that all time markers divide evenly by dt:
t_s = 0; % start of stroke (end of backswing)
t_h=10;% half point of t

```

```

t_f = 19; % end of stroke (start of follow through)
t_max = 20; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
t1=-t_max:dt:t_max;
yd1 = 10-10*cos(pi/(t_f-t_s)*(dt:dt:(t_h-t_s)));
yd2 = 10-10*cos(pi/(t_f-t_s)*(t_h-t_s)) + 1-1*cos(10*pi/(t_f-t_s)*(dt:dt:(t_h-t_s)));
yd = [0*(0:dt:t_s), yd1, yd2]; % yd = 0 from t = 0 thru t_s
P = tf([20.6],[1 36 191 336 180]);% Ballbeam closed-loop transfer function.
tsh = dt:dt:t_max+dt; % time shifted forward by dt
p = dt*impz(P,tsh);
ui = 0; % initial input
ei=yd;
e(1,:) = ei; % plotting array
lc=w_0.^4*[-359990000*exp(-w_0*t) + 1820000*prod([t;exp(-w_0*t)],1)]; %causal,w0=0.005
la=w_0.^4*[359990000*exp(-w_0*t) + 1780000*prod([t;exp(-w_0*t)],1)]; %anticausal,w0=0.005
L=[la(n:-1:2),lc]; %reverse and concatenate la and lc; L=F*P0^-1
L1=w_0.^4; % constant item of L

for i = 2:100,
    du=dt*conv(L,ei);
    du=du(n:2*n-1);
    ui = ui + du +L1*ei; % update control input
    yi = conv(p,ui); % y = Pu
    yi = yi(1:n); % truncate
    ei=yd-yi;
    e(i,:) = ei; % store error for plotting
    y(i,:)=yi;
end

plot(t,y([ 10 100 200 300 400],:),t,yd),grid on;

```

A14: Code of Figure 4.23

```

% This code produces Figure 4.23. Tracking a higher frequency signal.
clear;
w_0 = 0.01; % bandwidth of learning filter.
% It is assumed that all time markers divide evenly by dt:
t_s = 0; % start of stroke (end of backswing)
t_f = 9; % end of stroke (start of follow through)
t_max = 10; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
t1=-t_max:dt:t_max;

yd=5*sin(2*pi/4.5*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, 0*ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
P = tf([20.6],[1 36 191 336 180]);% Ballbeam closed-loop transfer function.
tsh = dt:dt:t_max+dt; % time shifted forward by dt
p = dt*impz(P,tsh);
ui = 0; % initial input

```

```

ei=yd;
e(1,:)=ei; % plotting array
lc=w_0.^4*[-359990000*exp(-w_0*t) + 1820000*prod([t;exp(-w_0*t)],1)]; %causal,w0=0.005
la=w_0.^4*[359990000*exp(-w_0*t) + 1780000*prod([t;exp(-w_0*t)],1)]; %anticausal,w0=0.005
L=[la(n:-1:2),lc]; %reverse and concatenate la and lc; L=F*P0^-1
L1=w_0.^4; % constant item of L

for i = 2:1000,
    du=dt*conv(L,ei);
    du=du(n:2*n-1);
    ui = ui + du +L1*ei; % update control input
    yi = conv(p,ui); % y = Pu
    yi = yi(1:n); % truncate
    ei=yd-yi;
    e(i,:)=ei; % store error for plotting
    y(i,:)=yi;
end

plot(t,y([ 10 100 1000],:),t,yd),grid on;

```

A15: Code of Figure 5.3

```

%Figure 5.3 is produced by this code.
% u_i = u_(i-1) + De_(i-1)
% D is composed of P^-1 and a real 4th-order filter with bandwidth = w_0.
clear;
k0 =0.5 ; % feedback gain in C.L. plant
k = 0.3; % can introduce model error via k_0 neq k; when k0=9, it diverges.
w_0 = 10; % bandwidth of learning filter

% It is assumed that all time markers divide evenly by dt:
t_s = 2; % start of stroke (end of backswing)
t_f = 12; % end of stroke (start of follow through)
t_max = 14; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
P=tf([1],[1 1 k]);
tsh = dt:dt:t_max+dt; % time shifted forward by dt
s = dt*impulse(S,tsh);
p = dt*impulse(P,tsh);
S_0 = (1+k0*G)^(-1); % model of S
s0 = dt*impulse(S,tsh);
%P_0 = S_0*G; % model of P
P_0=tf([1],[1 1 k0]);
p0 = dt*impulse(P_0,tsh);
W2=P/P_0-1;
F=tf([1],[1/w_0.^4 0 -2/w_0.^2 0 1]);
r = conv(s0,yd); %closed-loop reference
r = r(1:n); % truncate
ui = 0*t; % initial input
ei = r; %initial error

```

```

e(1,:) = ei; % plotting array
f = filter4(w_0,n); % compute filter used by L

for i = 2:20,
    du = conv(f,ei); % filter the error
    du = [du(n:2*n-1),zeros(1,n-1)]; % truncate and pad with zeros
    du = deconv(du,p0); % apply P_0 inverse
    ui = ui + du; % update control input
    yi = conv(p,ui); % y = Pu
    yi = yi(1:n); % truncate
    ei = r - yi; % error
    e(i,:) = ei; % store error for plotting
end

plot(t,e([1 3 5 8 9 10],:));grid on

```

A16: Code of Figure 5.5

```

%Figure 5.5 is produced by this code.
% Apply ILC to C.L. plant e=r-Pu
% where  $P = 1/(s^2+s+k)$ ,
%  $r = S y_d$ ,
%  $u_i = u_{i-1} + D e_{i-1}$ 
% D is composed of  $P^{-1}$  and a real 4th-order filter with bandwidth =  $w_0$ .
clear;
k0 = 0.5; % feedback gain in C.L. plant
k = 10; % can introduce model error via  $k_0 \neq k$ ; when  $k=9$ , it diverges.
w_0 = 10; % bandwidth of learning filter

% It is assumed that all time markers divide evenly by dt:
t_s = 2; % start of stroke (end of backswing)
t_f = 12; % end of stroke (start of follow through)
t_max = 14; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
G = tf([1],[1 1 0]);
S = (1+k*G)^(-1);
P=tf([1],[1 1 k]);
tsh = dt:dt:t_max+dt; % time shifted forward by dt
s = dt*impz(S,tsh);
p = dt*impz(P,tsh);
S_0 = (1+k0*G)^(-1); % model of S
s0 = dt*impz(S,tsh);
P_0=tf([1],[1 1 k0]);
p0 = dt*impz(P_0,tsh);
W2=P/P_0 -1;
F=tf([1],[1/w_0.^4 0 -2/w_0.^2 0 1]);
r = conv(s0,yd); %closed-loop reference
r = r(1:n); % truncate
ui = 0*t; % initial input
ei = r; %initial error

```

```

e(1,:)=ei;% plotting array
f = filter4(w_0,n); % compute filter used by L

for i = 2:20,
    du = conv(f,ei); % filter the error
    du = [du(n:2*n-1),zeros(1,n-1)]; % truncate and pad with zeros
    du = deconv(du,p0); % apply P_0 inverse
    ui = ui + du; % update control input
    yi = conv(p,ui); % y = Pu
    yi = yi(1:n); % truncate
    ei = r - yi; % error
    e(i,:) = ei; % store error for plotting
end

plot(t,e([1 10 20],:));grid on

```

A17: Code of Figure 5.7

```

%Figure 5.7 and Figure 5.8 are produced by this code.
clear;
k0=0.5;% feedback gain in C.L. plant
k = 3; % can introduce model error via k_0 neq k; when k0=9, it diverges.
w_0 = 5; % bandwidth of learning filter

% It is assumed that all time markers divide evenly by dt:
t_s = 2; % start of stroke (end of backswing)
t_f = 12; % end of stroke (start of follow through)
t_max = 14; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
G = tf([1],[1 1 0]);
S = (1+k*G)^(-1);
P=tf([1],[1 1 k]);
tsh = dt:dt:t_max+dt; % time shifted forward by dt
s = dt*impulse(S,tsh);
p = dt*impulse(P,tsh);
S_0 = (1+k0*G)^(-1); % model of S
s = dt*impulse(S,tsh);
P_0=tf([1],[1 1 k0]);
p0 = dt*impulse(P_0,tsh);
W2=P/P_0 -1;
F=tf([1],[1/w_0.^4 0 -2/w_0.^2 0 1]);
R1=(1-F)/(1+F*W2);% R=1 + 3.1168/(s+5.68) - 3.8547/(s-5.591) + 5.1989/(s-4.116) - 4.7962/(s+3.771)
    %+ (0.3362s+0.1881)/(s^2+1.255s+0.6339).
r1c=3.1168*exp(-5.68*t) - 4.7962*exp(-3.771*t) + 0.3362*prod([cos(0.49*t);exp(-0.6275*t)],1)
    - 0.0467*prod([sin(0.49*t);exp(-0.6275*t)],1); %causal,w0=0.005
r1a=-3.8547*exp(-5.591*t) + 5.1989*exp(-4.116*t); %anticausal,w0=0.005
r1=[r1a(n:-1:2),r1c]; %reverse and concatenate la and lc; L=F*P0^-1
e_inf=conv(r1,yd);
e_inf=e_inf(1:n);
r = conv(s,yd); %closed-loop reference

```

```

r = r(1:n); % truncate
ui = 0*t; % initial input
ei = r; %initial error
e(1,:) = ei; % plotting array
f = filter4(w_0,n); % compute filter used by L

for i = 2:100,
    du = conv(f,ei); % filter the error
    du = [du(n:2*n-1),zeros(1,n-1)]; % truncate and pad with zeros
    du = deconv(du,p0); % apply P_0 inverse
    dui = conv(f,ui); % filter the ui
    dui = dui(n:2*n-1); % truncate
    ui = dui + du; % update control input
    yi = conv(p,ui); % y = Pu
    yi = yi(1:n); % truncate
    ei = r - yi; % error
    e(i,:) = ei; % store error for plotting
end

plot(t,e([1 10 18 19 20],:),t,e_inf);grid on

```

A18: Code of Figure 5.9

```

% Figure 5.9 is produced by this code.
% Apply ILC to C.L. plant  $e=r-Pu$ 
% where  $P= 1/(s^2+s+k)$ ,
%  $r = Sy_d$ ,
%  $u_i = u_{(i-1)} + De_{(i-1)}$ 
% D is composed of  $P^{-1}$  and a real 4th-order filter with bandwidth =  $w_0$ .
clear;
k_0=0.5; % feedback gain in C.L. plant
a = 20; % can introduce model error via a neq 1;
w_0 = 10; % bandwidth of learning filter

% It is assumed that all time markers divide evenly by dt:
t_s = 2; % start of stroke (end of backswing)
t_f = 12; % end of stroke (start of follow through)
t_max = 14; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
G = tf([1],[1 1 0]);
P=tf([1],[1 a k_0]);
tsh = dt:dt:t_max+dt; % time shifted forward by dt
p = dt*impulse(P,tsh);
S_0 = (1+k_0*G)^(-1); % model of S
P_0 = S_0*G; % model of P
p_0 = dt*impulse(P_0,tsh);
s_0 = dt*impulse(S_0,tsh)
W2=P/P_0-1;
r = conv(s_0,yd); %closed-loop reference
r = r(1:n); % truncate

```



```

ui = 0*t; % initial input
ei = r; %initial error
e(1,:) = ei; % plotting array
f = filter4(w_0,n); % compute filter used by L
for i = 2:80,
    du = conv(f,ei); % filter the error
    du = [du(n:2*n-1),zeros(1,n-1)]; % truncate and pad with zeros
    du = deconv(du,p_0); % apply P_0 inverse
    ui = ui + du; % update control input
    yi = conv(p,ui); % y = Pu
    yi = yi(1:n); % truncate
    ei = r - yi; % error
    e(i,:) = ei; % store error for plotting
end
plot(t,e([1 78 79 80],:));grid on

```

A19: Code of Figure 5.11

```

%Figure 5.11 is produced by this code.
% Apply ILC to C.L. plant e=r-Pu
% where  $P = 1/(s^2+s+k)$ ,
% r = Sy_d,
% u_i = u_(i-1) + De_(i-1)
% D is composed of  $P^{-1}$  and a real 4th-order filter with bandwidth = w_0.
clear;
k_0=0.5; % feedback gain in C.L. plant
a = -0.5; % can introduce model error via a neq 1;
w_0 = 2; % bandwidth of learning filter

% It is assumed that all time markers divide evenly by dt:
t_s = 2; % start of stroke (end of backswing)
t_f = 12; % end of stroke (start of follow through)
t_max = 14; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
G = tf([1],[1 1 0]);
P=tf([1],[1 a k_0]);
tsh = dt:dt:t_max+dt; % time shifted forward by dt
p = dt*impulse(P,tsh);
S_0 = (1+k_0*G)^(-1); % model of S
P_0 = S_0*G; % model of P
p_0 = dt*impulse(P_0,tsh);
s_0 = dt*impulse(S_0,tsh)
W2=P/P_0-1;
F=tf([1],[1/w_0.^4 0 -2/w_0.^2 0 1]);
r = conv(s_0,yd); %closed-loop reference
r = r(1:n); % truncate
ui = 0*t; % initial input
ei = r; %initial error
e(1,:) = ei; % plotting array
f = filter4(w_0,n); % compute filter used by L

```

```

for i = 2:20,
    du = conv(f,ei); % filter the error
    du = [du(n:2*n-1),zeros(1,n-1)]; % truncate and pad with zeros
    du = deconv(du,p_0); % apply P_0 inverse
    ui = ui + du; % update control input
    yi = conv(p,ui); % y = Pu
    yi = yi(1:n); % truncate
    ei = r - yi; % error
    e(i,:) = ei; % store error for plotting
end
plot(t,e([1 18 19 20],:));grid on

```

A20: Code of Figure 5.13

```

% This code ilc_a.m produces the plots in Case 2.
% Apply ILC to C.L. plant e=r-Pu
% where  $P = 1/(s^2+s+k)$ ,
%  $r = S_y_d$ ,
%  $u_i = u_{(i-1)} + L e_{(i-1)}$ 
% L is composed of  $P^{-1}$  and a real 4th-order filter with bandwidth =  $w_0$ .
clear;
k_0=0.5; % feedback gain in C.L. plant
a = 0.5; % can introduce model error via a neq 1;
w_0 = 2; % bandwidth of learning filter

% It is assumed that all time markers divide evenly by dt:
t_s = 2; % start of stroke (end of backswing)
t_f = 12; % end of stroke (start of follow through)
t_max = 14; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
G = tf([1],[1 1 0]);
%S = (1+k*G)^(-1);
%P = S*G;
P=tf([1],[1 a k_0]);
tsh = dt:dt:t_max+dt; % time shifted forward by dt
%s = dt*impulse(S,tsh);
p = dt*impulse(P,tsh);
S_0 = (1+k_0*G)^(-1); % model of S
P_0 = S_0*G; % model of P
p_0 = dt*impulse(P_0,tsh);
s_0 = dt*impulse(S_0,tsh)
%P = dt*toeplitz([p(1),zeros(1,n-1)],p); % plant matrix for RIGHT multiplication
W2=P/P_0-1;
F=tf([1],[1/w_0.^4 0 -2/w_0.^2 0 1]);
R1=(1-F)/(1+F*W2);
r1c=0.5994*exp(-2.436*t) - 1.0554*exp(-1.125*t) - 0.08*prod([cos(0.6081*t);exp(-0.537*t)],1) +
    0.2586*prod([sin(0.6081*t);exp(-0.537*t)],1); %causal,w0=0.005
r1a=0.5352*prod([cos(0.4068*t);exp(-2.067*t)],1) - 2.5338*prod([sin(0.4068*t);exp(-2.067*t)],1);
    %anticausal,w0=0.005
r1=[r1a(n:-1:2),r1c]; %reverse and concatenate la and lc; L=F*P0^-1

```

```

e_inf=conv(r1,yd);
e_inf=e_inf(1:n);
r = conv(s_0,yd); %closed-loop reference
r = r(1:n); % truncate
ui = 0*t; % initial input
ei = r; %initial error
e(1,:) = ei; % plotting array
f = filter4(w_0,n); % compute filter used by L
for i = 2:20,
    du = conv(f,ei); % filter the error
    du = [du(n:2*n-1),zeros(1,n-1)]; % truncate and pad with zeros
    du = deconv(du,p_0); % apply P_0 inverse
    dui = conv(f,ui); % filter the ui
    dui = dui(n:2*n-1); % truncate
    ui = dui + du; % update control input
    yi = conv(p,ui); % y = Pu
    yi = yi(1:n); % truncate
    ei = r - yi; % error
    e(i,:) = ei; % store error for plotting
end
plot(t,e([1 18 19 20],:),t,e_inf);grid on

```

A21: Code of Figure 5.15

```

% Figure 5.15 is produced by this code.
% Apply ILC to C.L. plant e=r-Pu
% where P= 1/(s^2+s+k),
% r = Sy_d,
% u_i = u_(i-1) + De_(i-1)
% D is composed of P^-1 and a real 4th-order filter with bandwidth = w_0.
clear;
k_0=0.5; % feedback gain in C.L. plant
wu=5; % can introduce model error via a neq 1;
w_0 = 10; % bandwidth of learning filter

% It is assumed that all time markers divide evenly by dt:
t_s = 2; % start of stroke (end of backswing)
t_f = 12; % end of stroke (start of follow through)
t_max = 14; % end of trial
dt = .01; % sample time
n = t_max/dt+1; % number of samples in e and u
t = 0:dt:t_max; % time vector
yd = .5-.5*cos(pi/(t_f-t_s)*(dt:dt:(t_f-t_s)));
yd = [0*(0:dt:t_s), yd]; % yd = 0 from t = 0 thru t_s
yd = [yd, ones(size(t_f+dt:dt:t_max))]; % yd = 1 from t = t_f to t_max
G = tf([1],[1 1 0]);
tsh = dt:dt:t_max+dt; % time shifted forward by dt
S_0 = (1+k_0*G)^(-1); % model of S
P_0=tf([1],[1 1 k_0]);
P_1=tf([1],[1/wu 1]);
p_0 = dt*impulse(P_0,tsh);
s_0 = dt*impulse(S_0,tsh);
P=P_0*P_1;
p = dt*impulse(P,tsh);
W2=P/P_0-1;

```

```

r = conv(s_0,yd); %closed-loop reference
r = r(1:n); % truncate
ui = 0*t; % initial input
ei = r; %initial error
e(1,:) = ei; % plotting array
f = filter4(w_0,n); % compute filter used by L

for i = 2:8,
    du = conv(f,ei); % filter the error
    du = [du(n:2*n-1),zeros(1,n-1)]; % truncate and pad with zeros
    du = deconv(du,p_0); % apply P_0 inverse
    ui = ui + du; % update control input
    yi = conv(p,ui); % y = Pu
    yi = yi(1:n); % truncate
    ei = r - yi; % error
    e(i,:) = ei; % store error for plotting
end

plot(t,e([1 6 7 8],:));grid on

```

A22: Impulse response of real 4th order filter

```

function f = filter4(w_0,n)
% Impulse response of real 4th order filter.
% Sample time dt assumed.
dt = .01; % sample time
t_max = (n-1)*dt; % max time
t = 0:dt:t_max; % time vector

fc = .25*w_0*exp(-w_0*t) + .25*w_0^2*(t.*exp(-w_0*t)); % causal part
f = dt*[fc(n:-1:2), fc]; %reverse and concatenate to add noncausal part

```