İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

ADAPTIVE INVERSE CONTROL

M.Sc. Thesis by

Mustafa UYSAL, B.Sc.

Department : Mechanical Engineering

Programme: System Dynamics and Control

FEBRUARY 2006

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

ADAPTIVE INVERSE CONTROL

M.Sc. Thesis by
Mustafa UYSAL, B.Sc.
503031610

Date of submission : 19 December 2005

Date of defence examination: 2 February 2006

Supervisor (Chairman): Prof. Dr. N. Aydın HIZAL

Members of the Examining Committee Prof.Dr. Ahmet Kuzucu

Prof.Dr. Melih Geçkinli

FEBRUARY 2006

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

UYARLAMALI TERS KONTROL

YÜKSEK LİSANS TEZİ

Mustafa UYSAL
503031610

Tezin Enstitüye Verildiği Tarih : 19 Aralık 2005
Tezin Savunulduğu Tarih : 2 Şubat 2006

Tez Danışmanı : Prof.Dr. N. Aydın HIZAL
Diğer Jüri Üyeleri Prof.Dr. Ahmet KUZUCU
Prof.Dr. Melih Geçkinli

ŞUBAT 2006

FOREWORD

I would like to thank my supervisor Prof. Dr. Nafiz Aydın HIZAL for his guidance during this thesis.

December, 2005 Mustafa UYSAL

TABLE OF CONTENTS

ABBREVIATIONS

AIC : Adaptive inverse control

FIR : Finite impulse response

IIR : Infinite impulse response

LMS : Least mean square

MSE : Mean square error

tr : Trace

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF SYMBOLS

d_k : Desired response

E [] : Expected value

M(z) : Reference model transfer function

n_k : Disturbance

P : Input vector cross-correlation matrix

P(z) : Plant transfer function

R : Input vector auto-correlation matrix

s : Laplace operator

Ts : Sampling time

u_k : Plant input signal

$\mathbf{W}_k$: Weight vector at k^{th} iteration

$\mathbf{W}^*$: Optimum weight vector

$\mathbf{X}_k$: input vector at k^{th} iteration

y : Filter output

z : Discrete time operator

z^{-1} : unit delay in discrete time

ξ : Mean square error

Δ : Gradient, inverse modeling delay

μ : Convergence factor

ε : Error

$\hat{\nabla}$: Gradient estimate

δ_k : Dither signal

σ : A very small number to prevent division by zero

τ : Time constant

λ : Eigenvalue

UYARLAMALI TERS KONTROL

ÖZET

Uyarlamalı ters kontrol sistemleri, kontrol edilecek sistem hakkında tam bilgi sahibi olunmadığı ya da bu sistemin dinamiklerinin zamanla yavaşça değişim gösterdiği durumlarda kullanılması gerekli olabilecek açık ve kapalı çevrimli olabilen kontrol sistemleridir. Dinamikleri hakkında tam bilgiye sahip olunamayan sistemin kontrol edilebilmesi, sistem tanımlanması yöntemine dayanır.

Uyarlamalı FIR filtreler kullanılarak bilinmeyen sistemin direk modeli ya da ters modeli elde edilebilir. Genellikle bu sistemlerde hem direk model hem de ters model kontrol çevrimi sırasında elde edilir. Elde edilen ters model, kontrol organı olarak sisteme seri bağlanır ve böylelikle sistemin dinamiklerini iptal etmesi amaçlanır. Bu şekilde komut girişi ile sistemin cevabı arasındaki transfer fonksiyonu 1'e eşit olur ve sistem komut girişini izler. Birebir ters modelin oluştuğu durumlarda sistemin cevabının çok ani olacağı göz önünde bulundurularak, referans model tersi oluşturulması hedeflenir. Bu durumda kontrol organının transfer fonksiyonu sistemin transfer fonksiyonunu yine iptal eder, fakat sistemin cevabı referans modelin cevabı şeklinde olur. Bu şekilde istenen karakterde geçici rejim cevabı elde edilir.

Bu çalışmada öncelikle uyarlamalı ters kontrol sistemlerinin teorik esasları ele alınmıştır. Daha sonra sistemin performansını etkileyen parametreler modelleme süreçleri üzerinde incelenmiştir. Uyarlamalı ters kontrol sisteminin başarısı direk ve ters modelleme süreçlerinin başarısı ile doğru orantılıdır. Modeller bilinmeyen sisteme ne kadar yakınsarsa o derece hassas kontrol mümkün olacaktır.

Modelleme süreçleri üzerinde parametre seçimine karar verildikten sonra temel uyarlamalı ters kontrol şeması esas alınarak, dinamikleri zamanla yavaşça değişen kararlı bir sistem için kontrol sistemi benzetimi yapılmıştır. Benzetim uygulamaları Matlab programında hazırlanmıştır. Son olarak, kararsız davranış gösteren top-kiriş düzeneği üzerinde, uyarlamalı ters kontrol sisteminin, sisteme bozucu etkidiği durumlardaki performansı incelenmiştir.

ADAPTIVE INVERSE CONTROL

SUMMARY

Control of the plants whose dynamics are not known or slowly time variable needs a different approach than the conventional control methods. Controls of such plants are available with adaptive inverse control systems, which can work either open-loop or closed-loop. Controls of those plants are based on the system identification methods.

Direct or inverse models of the unknown plants are obtained by utilizing adaptive FIR filters. Usually both direct and inverse models are obtained within the control cycle. Inverse model of the unknown plant is used in the controller position to cancel the plant dynamics. Thus the transfer function between the input and the output signal is unity and the output follows the input signal just as is. If the controller is a perfect inverse then the system will response suddenly. In such cases a model reference inverse is obtained to smooth the transient response of the system. With model reference inverse, the controller cancels the plant dynamics but the response is the same as the reference model.

In this work theoretical background of adaptive inverse control is reviewed first. Then the parameters that have an effect on the performance of the system are investigated with adaptive modeling processes. Performance of the AIC systems is directly proportional with the success of the modeling process. The more the models are representing the unknown plant the efficient the AIC system is.

After examining the parameters, a control simulation based on the basic AIC scheme is applied on a time-varying system. Simulation applications are implemented within Matlab program. Finally, performance of the AIC system is examined on the unstable ball-beam setup in existence of disturbances.

1. INTRODUCTION

The principle concern of control theory is to keep the outputs of a dynamical system within desired limits. There are many methodologies developed to control system dynamics. Adaptive control is a branch of those methodologies with an adaptive viewpoint. Unlike the conventional controllers adaptive controllers modify its parameters in response to changes in the dynamics of the controlled system or the disturbances affecting the system. Conventional controllers work well when the plant dynamics are well known. A change in the dynamics of the system needs to readjust the parameters of the conventional controller. Therefore if there is no accurate information about the plant dynamics or those dynamics are slowly time-varying then adaptive controllers are taken into account.

Most of the controllers are based on feedback mechanisms. Utilization of feedback is different in adaptive inverse control concept. Feedback from the plant output is not directly fed to the controller input. Nevertheless it is not an open-loop control system. Controller is adapted with respect to the information from plant output and command input. The loop is closed through the adaptive process.

Adaptive inverse controllers utilize adaptive signal processing methods to perform adaptivity. Adaptive filters are used in a large number of applications such as channel equalization, interference (noise) cancellation and echo cancellation in digital communication systems. In adaptive inverse control they are used for identification of the plants.

In this work, structures and implementation of adaptive inverse controllers are reviewed and simulated with time-varying and disturbed plants. Matlab and its component Simulink are used for computer simulations. This work has seven chapters and two appendices. Next chapter is a review of adaptive filter theory. Structure of adaptive filters and fundamentals of algorithms for filter update are reviewed. Chapter 3 is about adaptive modeling. Usage of adaptive filters for obtaining direct models is explained with different schemes and direct modeling

examples are given. Chapter 4 shows the usage of adaptive filters for forming inverse models. Obtaining the inverses of minimum-phase and nonminimum-phase plants and reference model inverses are explained with examples. In chapter 5 direct and inverse models are combined to form an adaptive inverse control system. Possible schemes for dynamic control and disturbance canceling are explained. A basic AIC system is simulated as an example. Parameters that have an effect on AIC system are examined in chapter 6. Defining the filter length according to plant response and sampling time, effects of convergence factor and modeling signal characteristics and advantages of using normalized LMS are detailed in this chapter. Application of adaptive inverse control on a time-varying system and a disturbed system is explained on chapter 7. Simulink diagrams and Matlab program codes used in simulations are provided within appendices.

2. ADAPTIVE FILTERS

Adaptive inverse control is built upon the basics of adaptive filtering. This chapter introduces the fundamental concepts for adaptive inverse control. Adaptive filters are used in wide variety of signal processing and control applications for plant modeling and inverse plant modeling. Figure 2.1 represents a basic adaptive filter.

At every step of the way, adaptive filtering is present. It is important to think of the adaptive filter as a building block, having an input signal, having an output signal, and having a special input signal called the "error" which is used in the learning process. This building block can be combined with other building blocks to make adaptive inverse control systems [1].

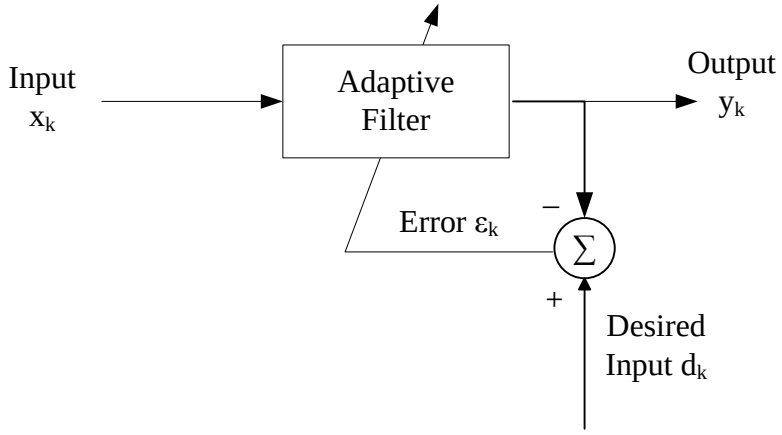


Figure 2.1 Basic representation of an adaptive filter

There are two types of linear adaptive filters: finite impulse response (FIR), and infinite impulse response (IIR). The impulse response of the FIR filter is non-zero for a finite period of time. However an IIR filter respond with non-zero values for an infinite period of time.

It is well known that any stable linear system may be approximated by a "sufficiently long" FIR filter. Therefore in this work we utilize FIR filters. All of the work presented in following chapters apply equally well to IIR filters, but their use was avoided due to the possibility of instability. FIR filters with finite weights are always stable. IIR filters are not [2].

2.1 Linear Combiner

Linear combiner is the start point for adaptive filtering. Because of its nonrecursive structure it is easy to understand and analyze.

The combiner is said to be linear because in the following analyzes, weights of the combiner are assumed to be fixed and under these assumptions the output of the combiner is linear combination of the input components.

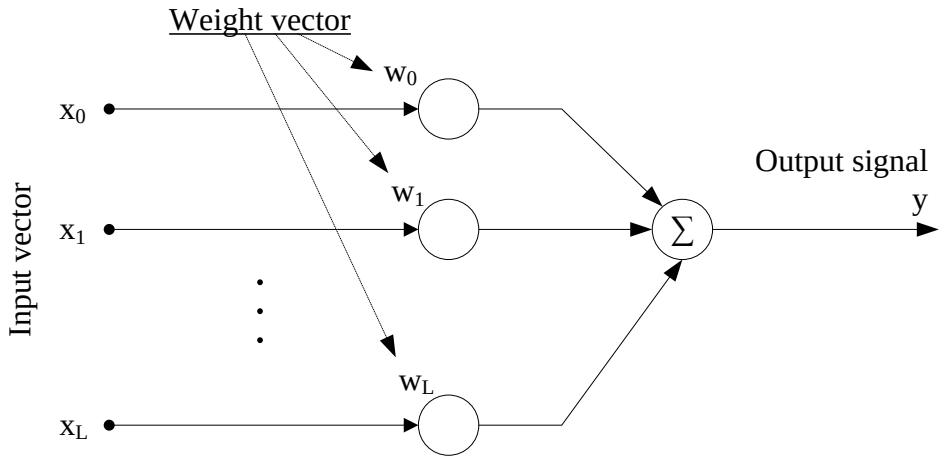


Figure 2.2 Linear Combiner

As depicted in figure 2.2, linear combiner comprises of an input signal vector, weights, a summing unit and an output signal.

Input signal vector;

$$\mathbf{X}_k = [x_{0k} \;\; x_{1k} \;\; \mathrm{L} \;\; x_{Lk}]^T \qquad \textbf{(2.1)}$$

Weight vector;

$$\mathbf{W}_k = [w_{0k} \;\; w_{1k} \; \mathrm{L} \;\; w_{Lk}]^T \tag{2.2}$$

The elements of the input signal vector are weighted and summed to form an output signal vector. For the present analysis the weights are assumed to be fixed. So it is possible to write;

$$y_k = \sum_{l=0}^{L} w_{lk} x_{lk} \tag{2.3}$$

$$y_k = \mathbf{X}_k^T \mathbf{W}_k = \mathbf{W}_k^T \mathbf{X}_k \tag{2.4}$$

At this point desired response is added to the linear combiner in order to develop adjustability. Weight adjustment is accomplished by comparing the output with the desired response to obtain an error signal and then adjusting the weight vector to minimize this signal.

$$\varepsilon_k = d_k - y_k = d_k - \mathbf{W}^T \mathbf{X}_k = d_k - \mathbf{X}_k^T \mathbf{W} \tag{2.5}$$

The new form of the combiner is depicted in figure 2.3 with the desired response. From now on, the combiner can be called adaptive linear combiner. The arrows on the weights represent adjustability.

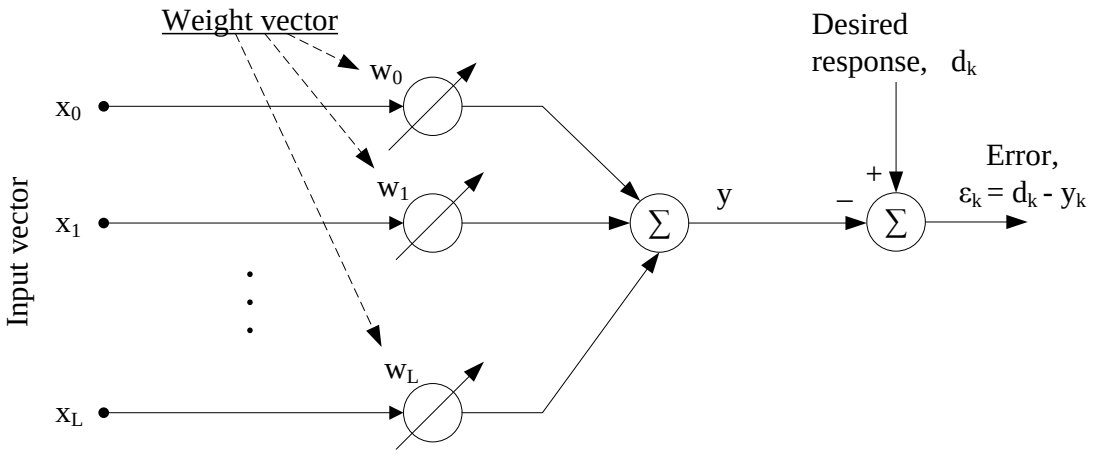


Figure 2. 3 Adaptive linear combiner

2.2 Performance Surface

The error signal was defined above. Now by squaring equation (2.5) the instantaneous squared error is obtained.

$$\varepsilon_k^2 = d_k^2 + \mathbf{W}^T \mathbf{X}_k \mathbf{X}_k^T \mathbf{W} - 2 d_k \mathbf{X}_k^T \mathbf{W} \qquad \textbf{(2.6)}$$

The expected value of ε_k^2 is defined as the mean square error.

$$\text{MSE} = \xi \; @E\left[\varepsilon_k^2\right] = E\left[d_k^2\right] + \mathbf{W}^T E\left[\mathbf{X}_k \mathbf{X}_k^T\right] \mathbf{W} - 2E\left[d_k \mathbf{X}_k^T\right] \mathbf{W} \qquad \textbf{(2.7)}$$

The input correlation matrix **R** is defined in equation (2.8);

$$\mathbf{R} \; @E\begin{bmatrix} x_{1k}x_{1k} & x_{1k}x_{2k} & \text{L} \\ x_{2k}x_{1k} & x_{2k}x_{2k} & \text{L} \\ \text{M} & \text{M} & x_{nk}x_{nk} \end{bmatrix} \qquad \textbf{(2.8)}$$

and the cross correlation vector **P** is defined in equation (2.9).

$$\mathbf{P} \; @E\begin{bmatrix} d_k x_{1k} \\ d_k x_{2k} \\ \text{M} \\ d_k x_{nk} \end{bmatrix} \qquad \textbf{(2.9)}$$

By writing MSE in terms of R and P equation (2.10) is obtained.

$$\xi = E\left[d_k^2\right] + \mathbf{W}^T \mathbf{R} \mathbf{W} - 2\mathbf{P}^T \mathbf{W} \qquad \textbf{(2.10)}$$

It is clear to see from equation (2.10) that MSE performance function is a quadratic function of the weights. It is a bowl-shaped surface in existence of two weights and this is depicted in figure 2.4. The point at the bottom of the bowl is projected onto the weight vector plane as W^*. That is the optimal weight vector or, point of minimum mean square error.

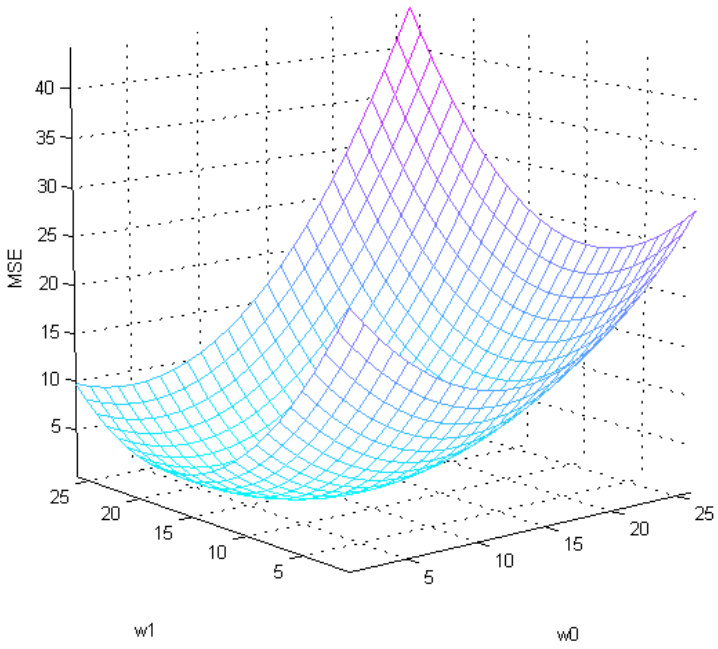


Figure 2. 4 Performance surface in existence of two weights

2.3 The Gradient and the Minimum MSE

The gradient at any point of the performance function is obtained by differentiating the MSE function in equation (2.10). Here ∇ stands for gradient.

$$\nabla @\frac{\partial \xi}{\partial \mathbf{W}} = \begin{Bmatrix} \frac{\partial E\left[\varepsilon_k^2\right]}{\partial w_1} \\ \text{M} \\ \frac{\partial E\left[\varepsilon_k^2\right]}{\partial w_n} \end{Bmatrix} = -2\mathbf{P} + 2\mathbf{RW} \qquad \textbf{(2.11)}$$

To find the optimal weight vector $\mathbf{W}^*$, the gradient is set to zero.

$$\nabla = 0 = 2\mathbf{RW}^* - 2\mathbf{P} \qquad \textbf{(2.12)}$$

Assuming that **R** is nonsingular;

$$\mathbf{W}^* = \mathbf{R}^{-1}\mathbf{P} \qquad \textbf{(2.13)}$$

Equation (2.13) is called the Wiener solution or the Wiener weight vector. This is the optimal solution for a FIR filter and usually this filter would be casual that do not have output signal unless there is input signal. The minimum MSE is now obtained by substituting $\mathbf{W}^*$ from equation (2.13) for **W** in equation (2.10).

$$\xi_{\min} = E\left[d_k^2\right] + \mathbf{W}^{*T}\mathbf{R}\mathbf{W}^* - 2\mathbf{P}^T\mathbf{W}^* \qquad \textbf{(2.14)}$$

$$\xi_{\min} = E\left[d_k^2\right] + \left[\mathbf{R}^{-1}\mathbf{P}\right]^T \mathbf{R}\mathbf{R}^{-1}\mathbf{P} - 2\mathbf{P}^T\mathbf{R}^{-1}\mathbf{P} \qquad \textbf{(2.15)}$$

Simplification of equation (2.15) by matrix manipulations yields us to equation (2.16)

$$\xi_{\min} = E\left[d_k^2\right] - \mathbf{P}^T\mathbf{W}^* \qquad \textbf{(2.16)}$$

By substituting equations (2.13) and (2.16) into equations (2.10) we obtain;

$$\xi = \xi_{\min} + (\mathbf{W} - \mathbf{W}^*)^T\mathbf{R}(\mathbf{W} - \mathbf{W}^*) \qquad \textbf{(2.17)}$$

2.4 The Method of Steepest Descent

The method of steepest descent uses gradients of the performance function in seeking its minimum. Each change in the weight vector is made proportional to the negative of the gradient vector.

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu\left(-\nabla_k\right) \qquad \textbf{(2.18)}$$

In equation(2.18), ∇_k stands for the gradient at the k. iteration and μ is called the convergence factor or step size. It is a scalar value and has crucial effects on the stability and the speed of adaptation. For stability of the equation (2.18) it is necessary that

$$\frac{1}{\lambda_{\max}} > \mu > 0 \qquad \textbf{(2.19)}$$

$\lambda_{\max}$ represents the largest eigenvalue of **R**.

2.5 The Least Mean Squares (LMS) Algorithm

The LMS algorithm is an implementation of steepest descent method. Instead of using the actual gradient, a simple estimate of the gradient is used. There is no squaring, averaging or differentiation in the algorithm so it is relatively simple and efficient.

The actual gradient

$$\nabla = \frac{\partial E\left[\varepsilon_k^2\right]}{\partial \mathbf{W}} \tag{2.20}$$

is replaced by the estimate

$$\hat{\nabla}_k = \begin{Bmatrix} \frac{\partial \varepsilon_k^2}{\partial w_1} \\ \text{M} \\ \frac{\partial \varepsilon_k^2}{\partial w_n} \end{Bmatrix} = 2\varepsilon_k \begin{Bmatrix} \frac{\partial \varepsilon_k}{\partial w_1} \\ \text{M} \\ \frac{\partial \varepsilon_k}{\partial w_n} \end{Bmatrix} = -2\varepsilon_k \mathbf{X}_k \tag{2.21}$$

By substituting the gradient estimate in equation(2.18) the equation for LMS algorithm is obtained.

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu\left(-\hat{\nabla}_k\right) \tag{2.22}$$

$$\mathbf{W}_{k+1} = \mathbf{W}_k + 2\mu\varepsilon_k \mathbf{X}_k \tag{2.23}$$

In equation(2.19) the interval for μ was defined but here, a stronger condition for convergence is needed.

Stability of the LMS algorithm is guaranteed if the convergence constant μ is selected within the range $(1/tr\mathbf{R}) > \mu > 0$ [3].

Normalized LMS algorithm is a variant of the LMS algorithm with much faster convergence in many cases. Convergence factor μ is normalized by the energy of the input signal vector.

$$\mu_{NLMS} = \frac{1}{2}\frac{1}{\mathbf{X}_k^T\mathbf{X}_k + \sigma}\mu \qquad \textbf{(2.24)}$$

In equation (2.24) σ is a very small number added for preventing division by zero if $\mathbf{X}_k^T\mathbf{X}_k$ is very small.

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \frac{\mu}{\sigma + \mathbf{X}^T\mathbf{X}}\varepsilon_k\mathbf{X}_k \qquad \textbf{(2.25)}$$

This form of LMS algorithm is independent of signal scaling. As the input signal power changes, the algorithm adjusts the convergence factor to maintain an appropriate value. Thus the step size changes with time. As a result, the normalized algorithm converges more quickly in many cases. For input signals that change slowly over time, the normalized LMS can represent a more efficient LMS approach.

Using gradient estimate in the LMS algorithm causes noise in the weight vector. Thus noisy adaptation leads to an MSE larger than the optimal value. Misadjustment is defined to quantify the increase in the MSE.

$$M\ @\frac{\text{average excess MSE}}{\xi_{\min}} \qquad \textbf{(2.26)}$$

It is desirable to keep M as small as possible. A value of $M = 10$ percent means that the adaptive system has an MSE only 10 percent greater than $\xi_{\min}$. Years of experience with adaptive filters convinces one that a 10 percent misadjustment is satisfactory for many engineering designs. Operation with 10 percent misadjustment can generally be achieved with an adaptive settling time equal to 10 times the memory time span of the adaptive filter. Adapting faster will cause more misadjustment. Adapting slower will result in less misadjustment [1].

3. ADAPTIVE MODELING

Adaptive plant modeling, also named as adaptive system identification, is an integral part of an adaptive control process. The basic structure of adaptive plant modeling is illustrated in figure 3.1.

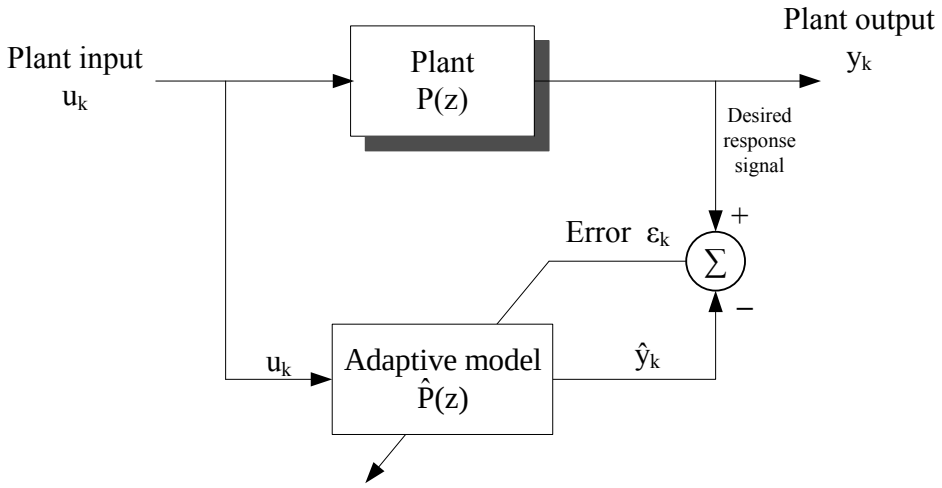


Figure 3. 1 Basic structure of adaptive plant modeling

In many cases, the plant to be controlled may be unknown and possibly time variable. In order to apply adaptive modeling the plant must be stable. For present purposes the plant is assumed to stable and linear-time invariant. Modeling process works in discrete time, in figure 3.1 all systems and signals are considered to be sampled. Both the plant and the adaptive filter receive the same input signal. The output of the plant is the desired response for the adaptive filter. The discrete time impulse response of the plant is formed thorough the filter by varying the weights of the linear combiner.

After convergence the weights contain the identification information about the plant dynamics in the form of an impulse response shape [3].

In an adaptive modeling process, plant disturbance and plant output sensor noise can be handled as an additive disturbance at the plant output. This situation is depicted in figure 3.2. n_k is the discrete-time additive noise and the overall plant output is z_k.

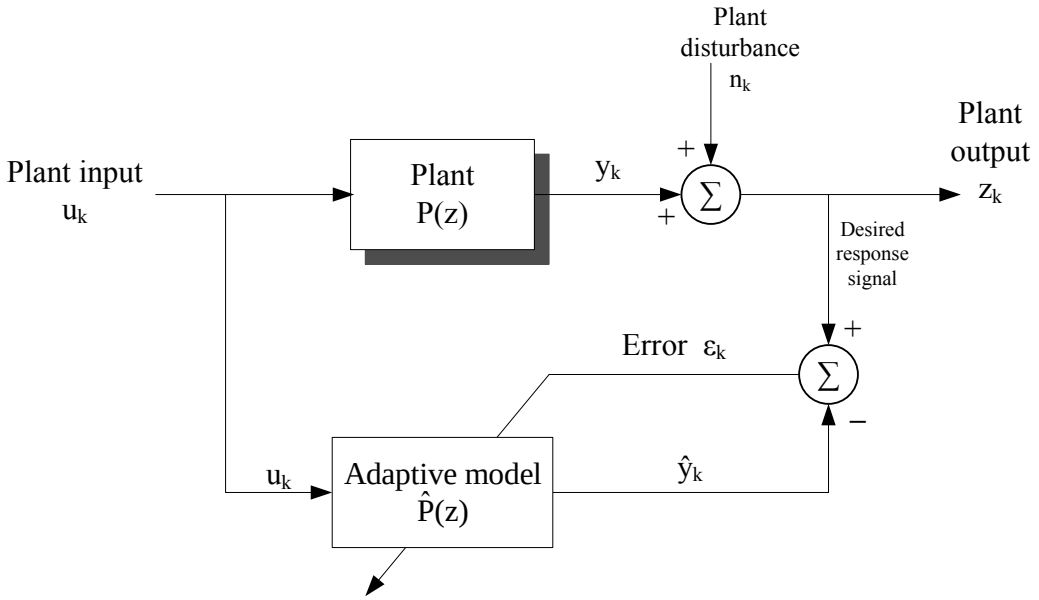


Figure 3. 2 Adaptive modeling of a noisy plant

Assuming the input signal excites all the plant modes, it is statistically stationary and not correlated with the disturbance n_k, the adaptive algorithm will develop a transfer function equal to that of the plant. Although the desired response for the adaptive process is the disturbed plant output z_k, it will give the same Wiener solution as if it was trained with y_k.

Estimated adaptive models will be very close representations of the actual plants. However there will be differences between the actual plant $P(z)$ and the estimated model $\hat{P}(z)$. These differences are called mismatch. There are three sources of mismatch.

In practice the plant to be modeled will have an infinite impulse response (IIR). Modeling the IIR plant with a FIR filter will result in mismatch. In order to overcome this issue, delay line length must be long enough so that the model's impulse response duration can cover the most significant part (practical memory time) of the impulse response of the system to be modeled. If the delay line length is

selected so long, the unnecessary weights at the end will tend to become zero. Also selecting a long line will increase misadjustment. The noise in the model weights increase by the number of the weights. One can find the satisfactory values by trial and error through the simulations.

To achieve a close match between the adaptive model and the unknown plant over a specified range of frequencies, the plant input u_k needs to have spectral energy over this range of frequencies. If the plant input has uniform spectral density over the frequencies of interest, then error tolerance will be uniformly tightly held over this frequency range. In many cases, however, the plant input u_k fails to have adequate spectral density at all frequencies where good fit is required. The result is mismatch, development of a difference between $\hat{P}$ and P [1].

A non-stationary input signal will result in same problem. Systems switching between constant levels and holding the levels for a long time represents this kind of situations.

Dither signals added to the plant inputs are used to circumvent these difficulties. In figure 3.3, one of the dither scheme A is depicted.

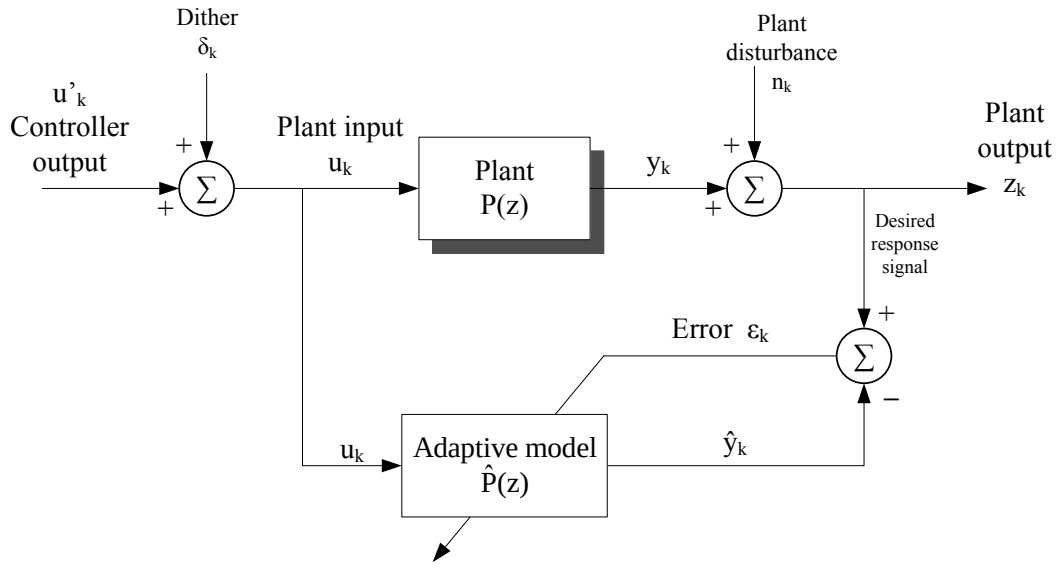


Figure 3. 3 Dither scheme A

Dither scheme A is a straightforward way of plant modeling. The dither δ_k is simply added to the controller output to form the plant input. Hence a desired spectral character for u_k is obtained. This scheme works well if the controller output is a stationary stochastic process. Dither scheme A comes with the drawback of having an increased minimum mean square error.

For the cases of u_k being non-stationary, dither scheme C is depicted in figure 3.4

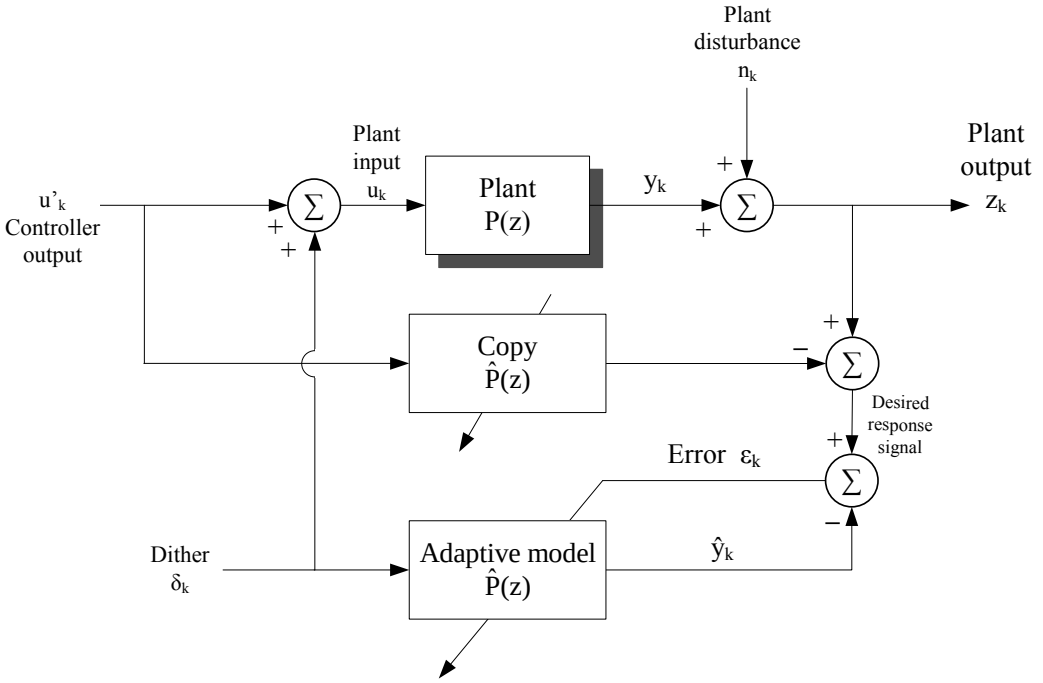


Figure 3. 4 Dither scheme C

This is a more complex layout, but it eliminates the drawbacks of scheme A. The non-stationary controller output does not pass through adaptive filter; an exact digital copy is used instead. Output of the exact digital copy is used for obtaining the desired response for the actual adaptive filter.

Selecting white noise as the dither signal is adequate for modeling signal characteristics; however the power level of this signal must be well optimized. Low level dither power will slow down the adaptive process. High levels of power will make the adaptation faster but it comes with increasing noise in the weights.

As an example of direct modeling of the plant given in equation 3.1, step responses of both actual and modeled plants are shown on figure 3.5

$$P_1(z) = \frac{1+2z^{-1}}{1-z^{-1}+\frac{3}{4}z^{-2}} \tag{3.1}$$

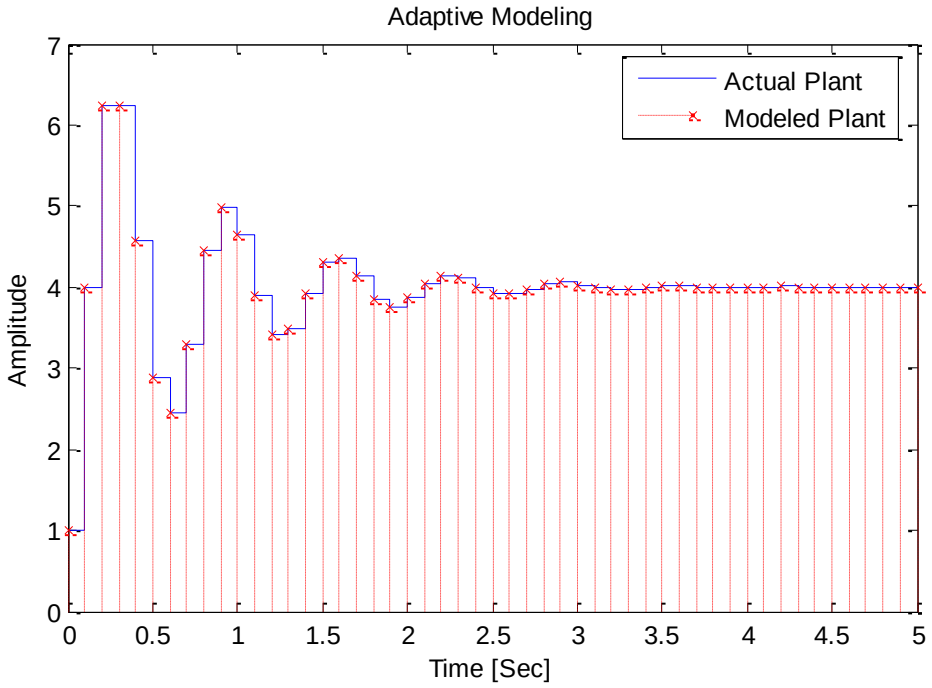


Figure 3. 5 Direct modeling result

Plant modeled with 50 weights, convergence factor is set to 0.1 and normalized LMS algorithm is used. With these values direct model converged within 3000 iteration values and a perfect fit is obtained.

To see the adaptive modeling performance under disturbance random noise is added to the plant output. To make the modeling signal correlated with the noise, random number generator seeds are set to a equal value. When the disturbance was correlated with the modeling signal it caused bias in the direct model without noise in the weights. The biased solution can be observed from figure 3.6. With the uncorrelated disturbance adaptive model converged to the actual plant but the results are noisy. Squared errors for both correlated and uncorrelated disturbances are plotted on figure 3.7.

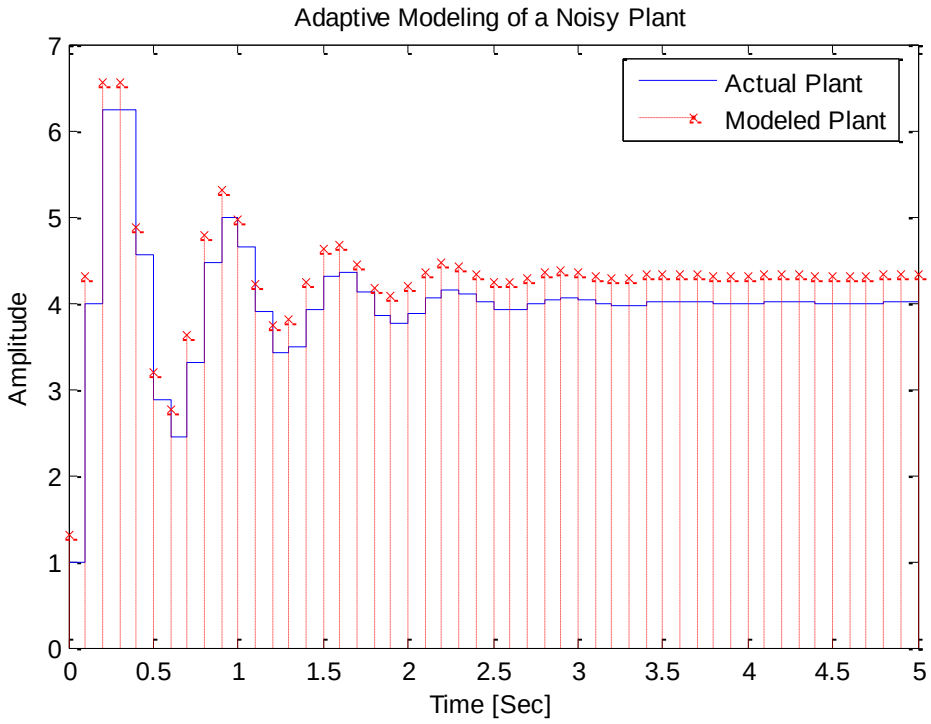


Figure 3. 6 Added disturbance is correlated with the modeling signal, converged model is biased.

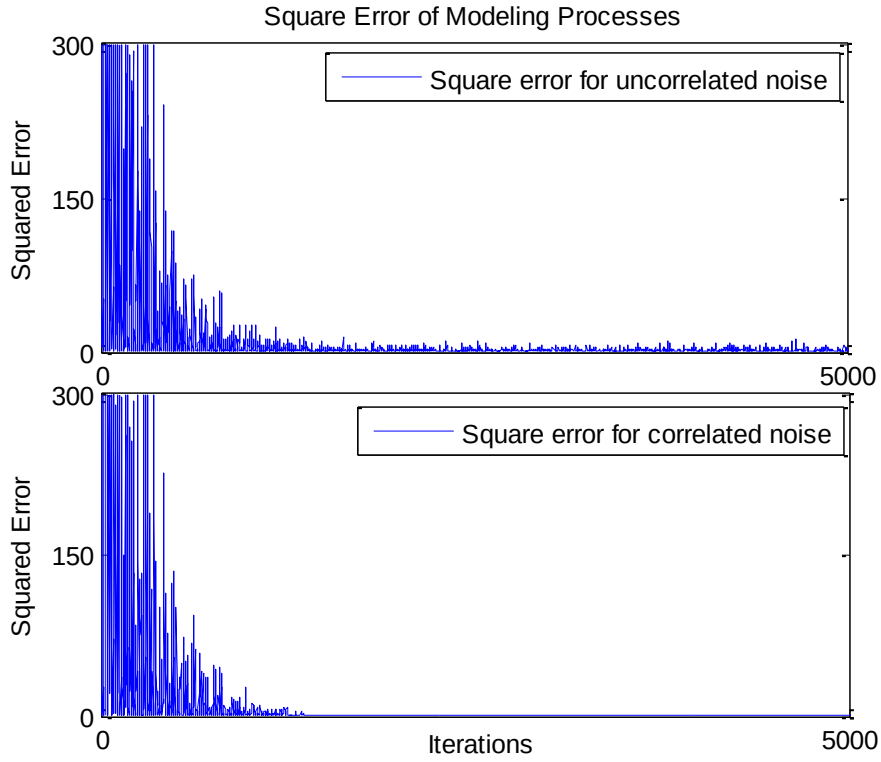


Figure 3. 7 Direct modeling errors

4. ADAPTIVE INVERSE MODELING

Plant inverses are used as controllers in adaptive inverse control. To obtain inverse model in place of the direct model, the roles of the plant input and plant output are interchanged on the adaptive modeler. The plant input is the desired response and the plant output is the input to the adaptive inverse modeler.

Adaptive inverse modeling is only applicable for stable plants. If the plant to be modeled is unstable, it must be stabilized with conventional feedback.

The plant generally has zeros and poles thus the inverse of it should have also. If the plant has all of its zeros inside the unit circle in the z-plane, then it is called a minimum-phase plant. If any of the zeros is outside the unit circle, then it is called a nonminimum-phase plant. The inverse of a nonminimum-phase plant will have poles outside the unit circle which makes it unstable.

Inverse plant modeling scheme for minimum-phase plants is depicted in figure 4.1. The adaptive filter is connected cascade with the plant to be inverse modeled. As assumed before that the plant $P(z)$ is minimum-phase, it should have a perfect inverse $C(z) = 1/P(z)$. $C(z)$ is both stable and casual.

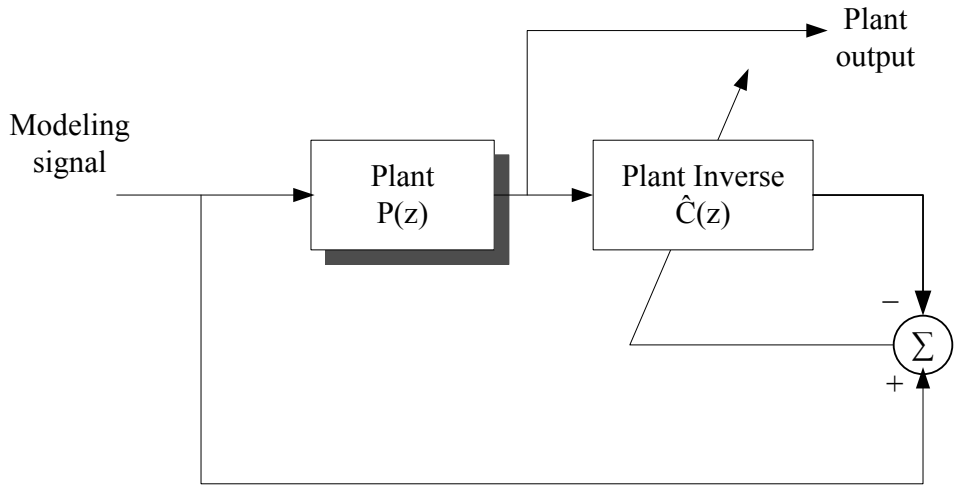


Figure 4. 1 Inverse modeling of a minimum-phase plant

Adaptive algorithm would provide an inverse $\hat{C}(z)$ which closely approximates $C(z)$ when it has sufficient number of degrees of freedom.

The technique depicted in figure 4.1 will not work if the plant is nonminimum-phase or it has transport delay. In such cases delaying the desired response through the inverse modeling process will make the adaptive filter capable of forming a working controller. Modeling scheme for delayed plant inverse is depicted in figure 4.2.

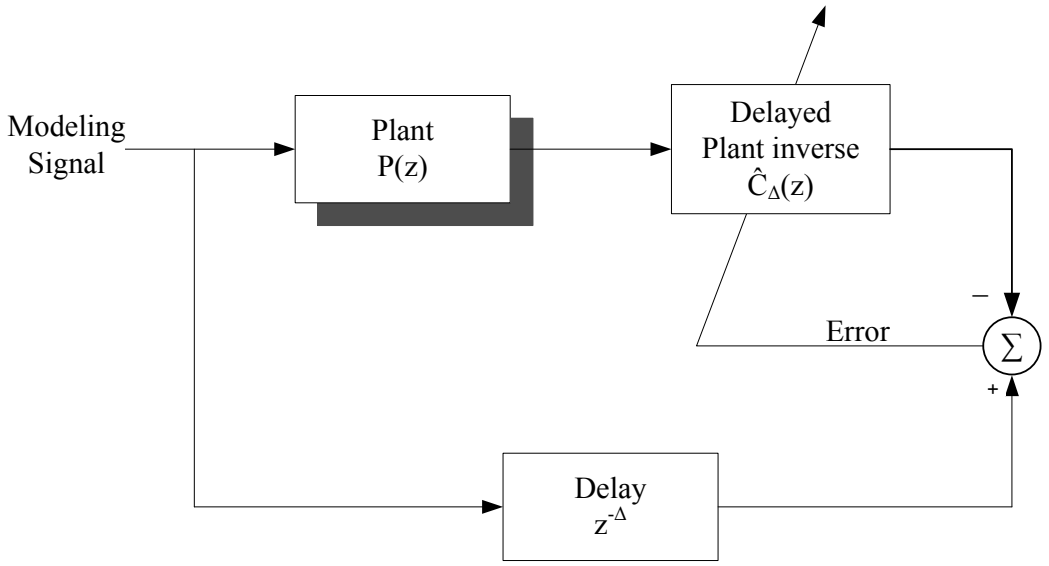


Figure 4. 2 Forming a delayed plant inverse

Here Δ stands for the delay amount. Selecting larger values of Δ would result in more perfect inverses. But the delay in the overall control system would be greater if the inverse filter were used as a controller.

For any minimum-phase plant, $\Delta = 0$ would suffice, except when the plant has more poles than zeros, then $\Delta = 1$ would suffice. Because when the analog plant is discretized, more poles than zeros causes the discrete impulse response to begin after a delay of one sample period [1].

For FIR filters, increasing Δ beyond the point of any reasonable need could cause the impulse response to be pushed out of the time window.

By replacing the delay block in figure 4.2 with a reference block, model-reference inverses can be obtained. Such a scheme is depicted in figure 4.3.

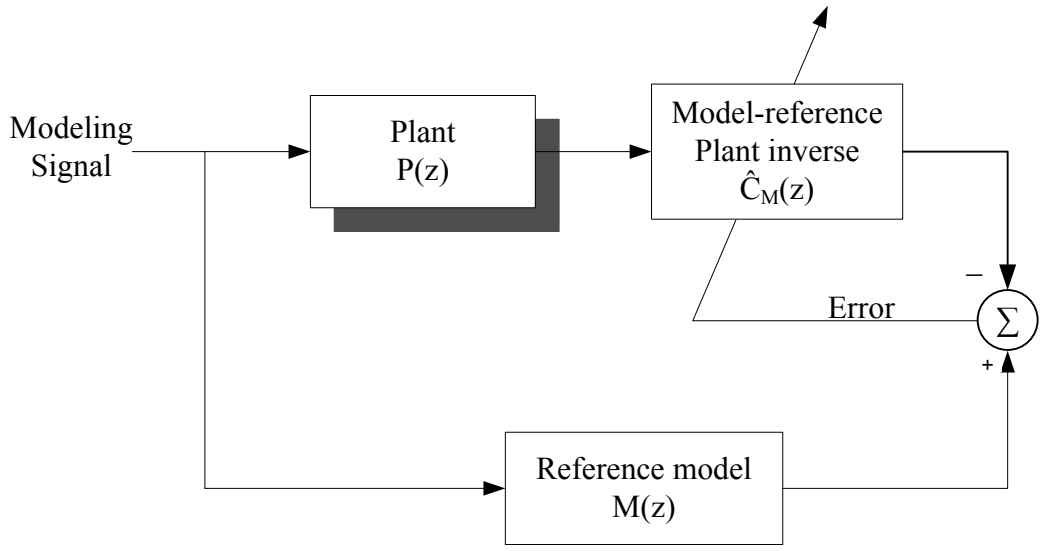


Figure 4. 3 Obtaining model-reference plant inverse

The goal of this process is to obtain a controller $\hat{C}_M(z)$ that, when used to drive the plant, would result in a control system whose overall transfer function would closely match the transfer function $M(z)$ of a given reference model.

Reference-models should reflect the dynamics that is desired at the controlled plant output. By using reference-models, smooth transient responses could be achieved in the cases of perfect inverses. Because perfect inverses, when used as controllers, would result in sudden responses of the controlled system, which may be sometimes unwanted.

In chapter three, it was mentioned that plant disturbance does not affect the Wiener solution when the plant is directly modeled. But in the case of inverse modeling, previously introduced modeling schemes will not work if plant disturbance exists. Such a scheme is illustrated in figure 4.4 which prevents the formation of a proper inverse. Proper methods for inverse modeling of a plant with disturbance are depicted in figure 4.5 and 4.6, which illustrates online and offline processes respectively

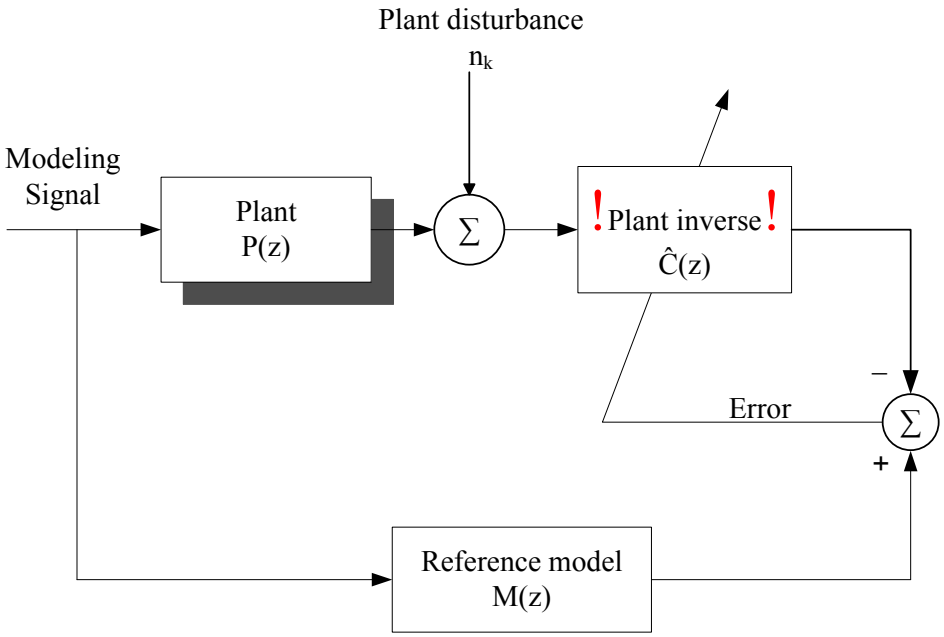


Figure 4. 4 An incorrect method for inverse modeling of a plant with disturbance

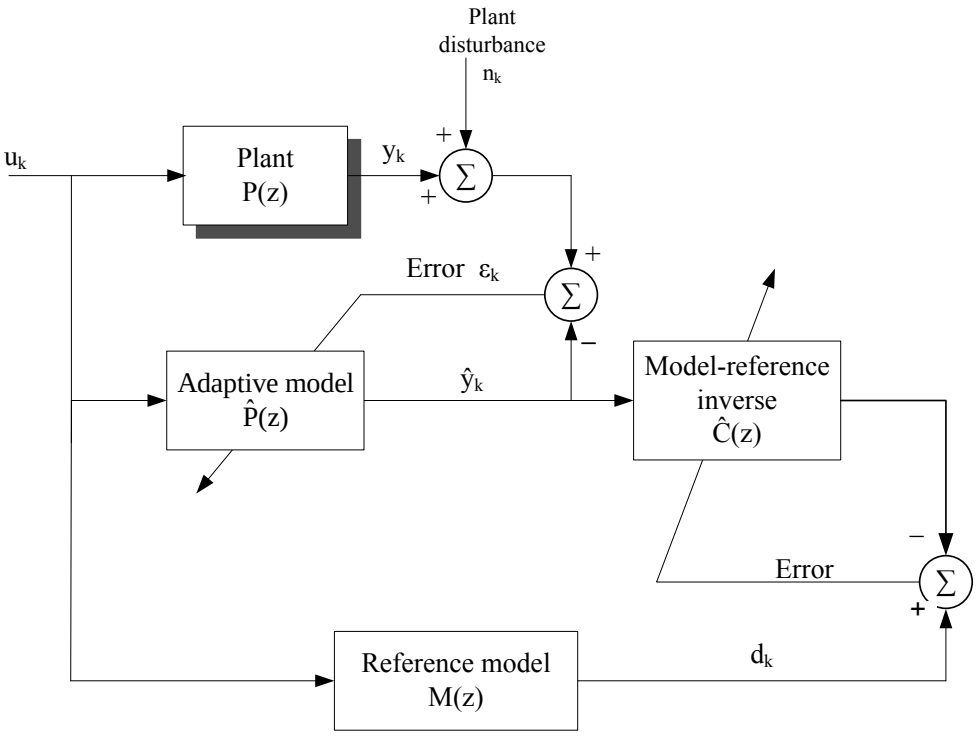


Figure 4. 5 A proper method for online inverse modeling of a plant with disturbance

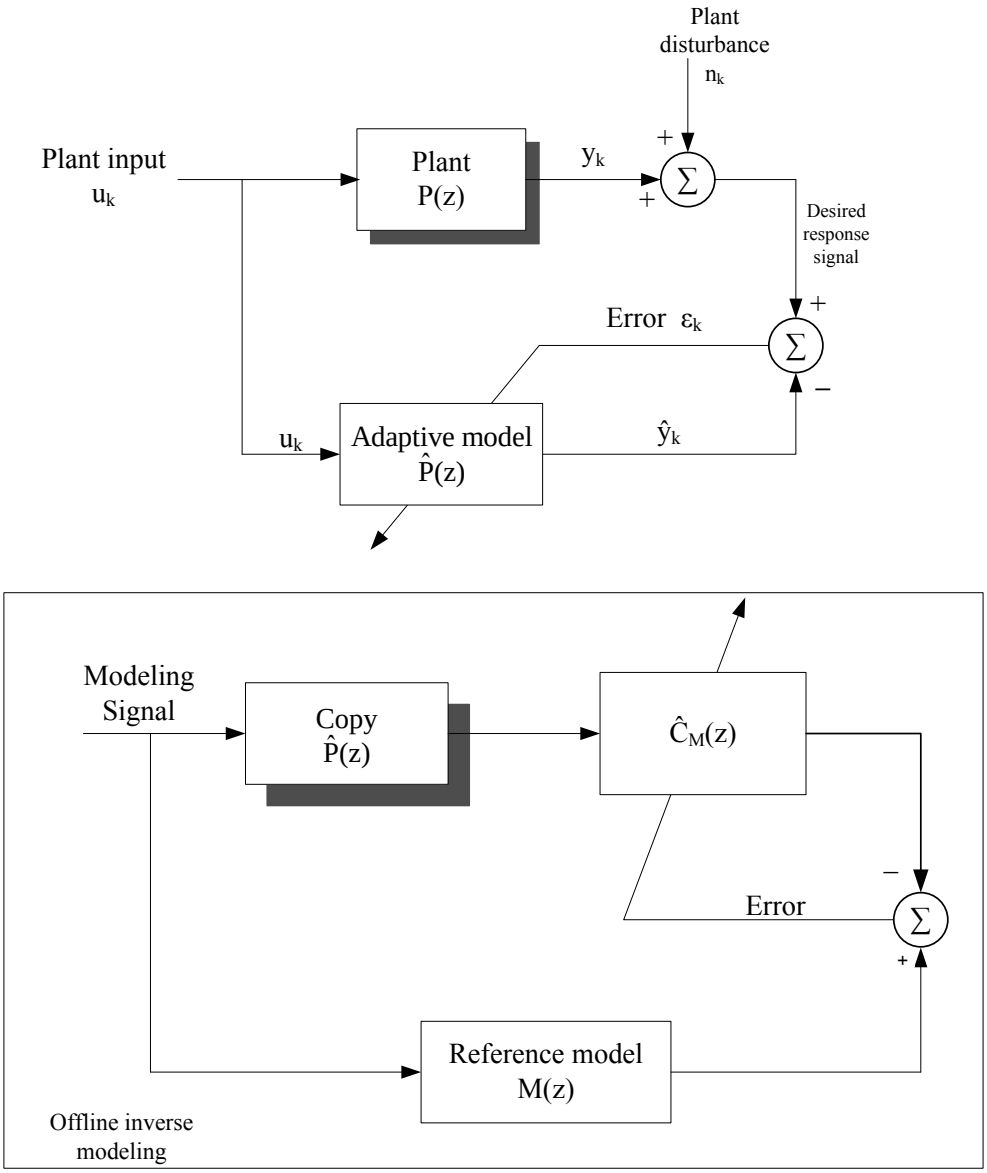


Figure 4. 6 An offline process for inverse modeling of a plant with disturbance

In figure 4.5 direct model of the plant $P(z)$ is used. Instead of finding a model-reference inverse of $P(z)$, the model-reference is taken from $\hat{P}(z)$ which has the same dynamic response as $P(z)$ but is free of disturbance. There are two adaptive processes working in cascade. Second adaptive process for obtaining $\hat{C}(z)$ will always be lagging behind the first one for obtaining $\hat{P}(z)$. This lag is prevented in offline process depicted in figure 4.6. Direct model of the plant $\hat{P}(z)$ is obtained first and an exact digital copy of it is used in an offline process to obtain $\hat{C}(z)$. This is much faster than online process.

The modeling signal has significant effect over the control system. Here $\hat{C}(z)$ is restricted to be causal and FIR. Consequently there will not be enough weights and degrees of freedom in $\hat{C}(z)$ to perfectly match $C(z)$. Under these conditions the spectral shape of the modeling signal could have considerable influence on the frequency response of $\hat{C}(z)$. In general the frequency response curves of $M(z)$ and $\hat{P}(z)\cdot\hat{C}(z)$ will be different.

Optimizing $\hat{C}(z)$ with a white modeling signal will cause transfer function differences to be weighted equally at all frequencies causing the area of the difference of the two frequency response curves to be minimized. Using a non-white modeling signal causes frequency response differences to be weighted more heavily at frequencies where the modeling signal has higher power density [1].

To test the schemes introduced in this chapter the systems given in equation 3.1 and 4.1 are used. System of P_1 given in equation 3.1 is a stable but nonminumum-phase plant. System of P_2 given in equation 4.1 is a stable and minimum-phase plant.

$$P_2(z) = \frac{1+\frac{1}{2}z^{-1}}{1-z^{-1}+\frac{3}{4}z^{-2}} \qquad \textbf{(4.1)}$$

Reference model given in equation 4.2 is used to test model reference inverses.

$$M(z) = \frac{0.25z^{-1}}{(1-0.5z^{-1})^2} \qquad \textbf{(4.2)}$$

To see the success of inverse modeling process, the inverse model built with using the weights in a digital filter form and a unit step input is applied to the convolution of the inverse and the actual plant. In case of a perfect inverse the response would be 1 with no transient response. This indeed would be very difficult. In figure 4.7 deconvoluted plant's response is plotted. It has oscillatory transient response and steady state error. A reference model is added to the process and response of the deconvoluted plant has been greatly enhanced. The result is plotted on figure 4.8.

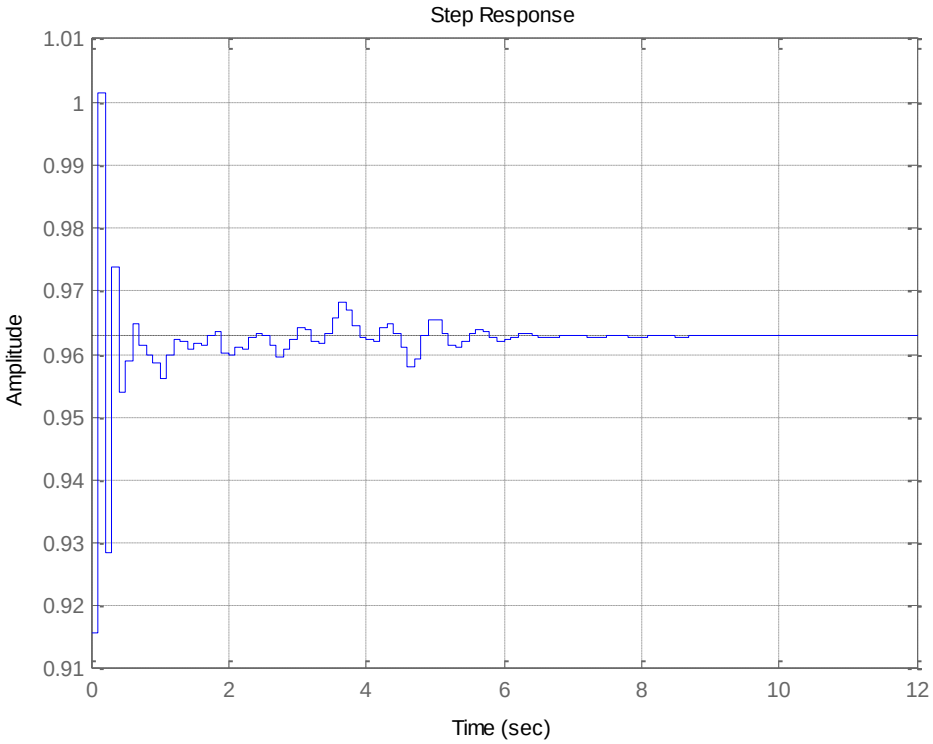


Figure 4. 7 Deconvolution of plant dynamics without reference model

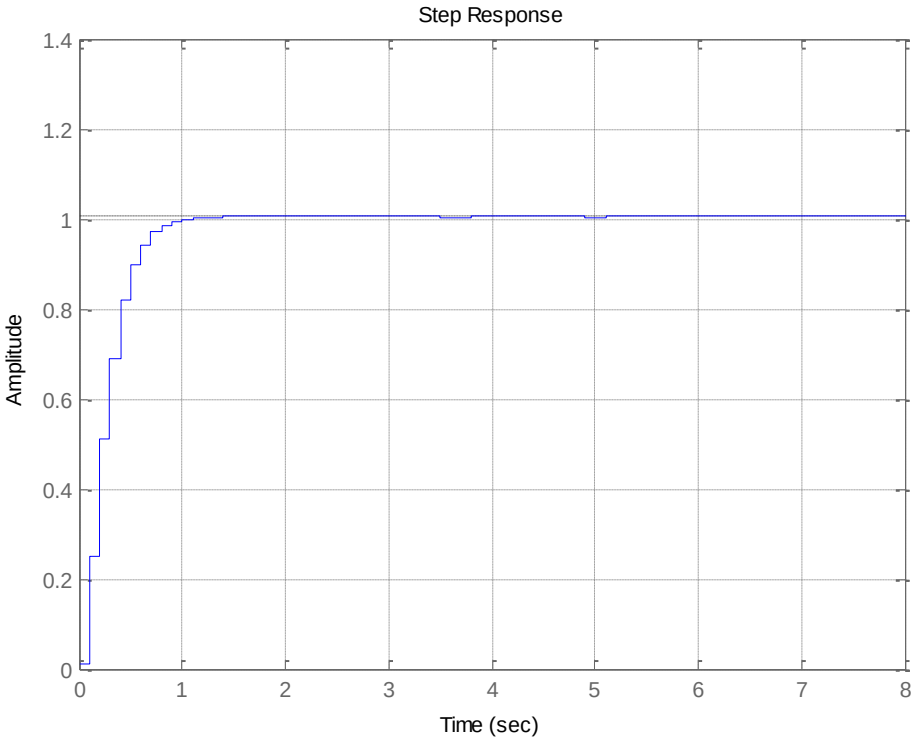


Figure 4. 8 Deconvolution of plant dynamics with reference model

To obtain an inverse model of the nonminumum-phase plant P_1, the desired response is delayed 3 samples. Its Deconvolution of the plant is plotted on figure 4.9. The delay can be easily seen from this response.

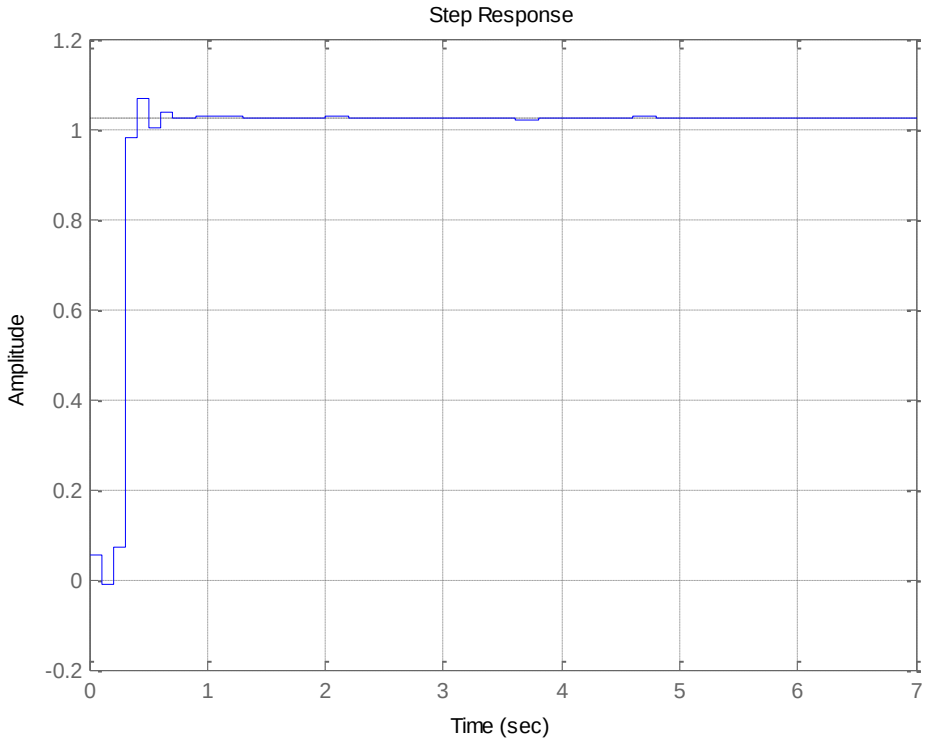


Figure 4. 9 Deconvolution of plant dynamics with delayed plant inverse.

5. ADAPTIVE INVERSE CONTROL

Previous chapters introduced us the elementary parts of the adaptive inverse control. These components take place in the system according to the desired specifications to form up an adaptive inverse control system. The main objective is how to lineup those elements to do a successful control task. The lineups of the blocks changes by the plant characteristics, signals fed to the system and the desired output of the overall control task.

The fundamental idea behind inverse control is to cancel the plant dynamics by using the inverse of it as a controller. This is depicted in figure 5.1 basically.

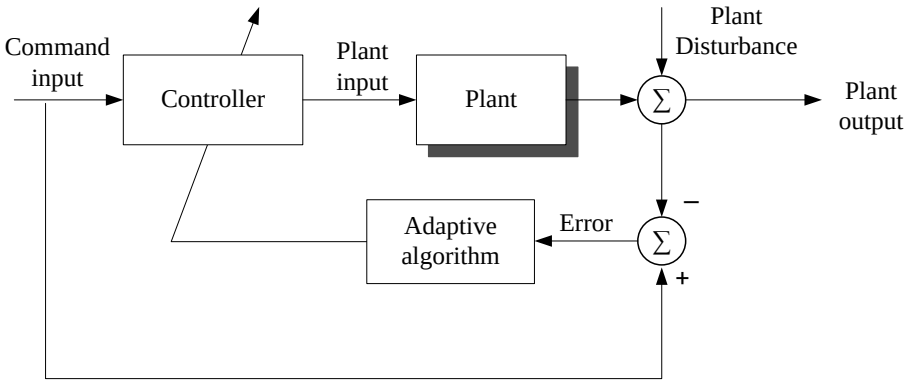


Figure 5. 1 Basic idea of adaptive inverse control

Figure 5.1 introduces the simplest form of inverse control which is applicable with different adaptive algorithms such as differential steepest descent (DSD) and linear random search (LRS), but LMS algorithm cannot be used to adapt the weights of the controller of figure 5.1. LMS is preferred because it is much faster than the others.

LMS needs its input from the plant output and an error signal referred to plant input to form a plant inverse. Figure 5.2 depicts the desired scheme for LMS to work. This scheme utilizes LMS for developing an inverse controller but it will not function when the plant disturbance has high level.

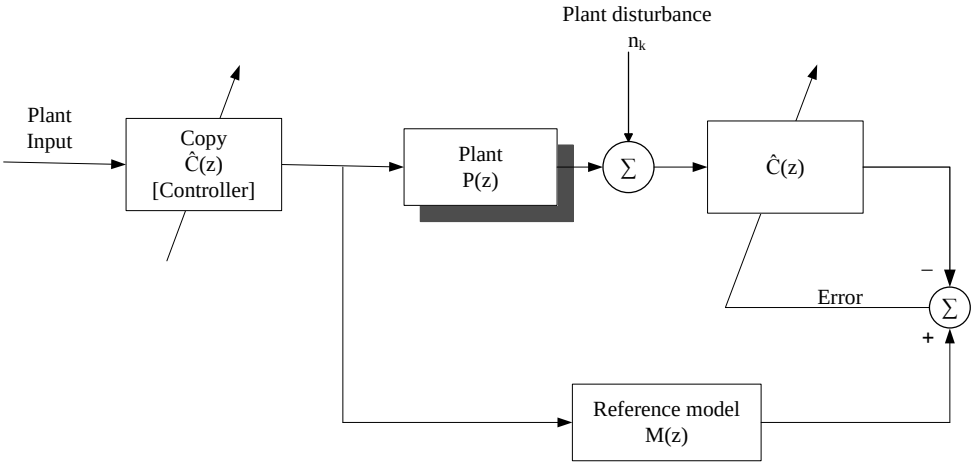


Figure 5. 2 Appropriate AIC system that works with LMS at low disturbance levels

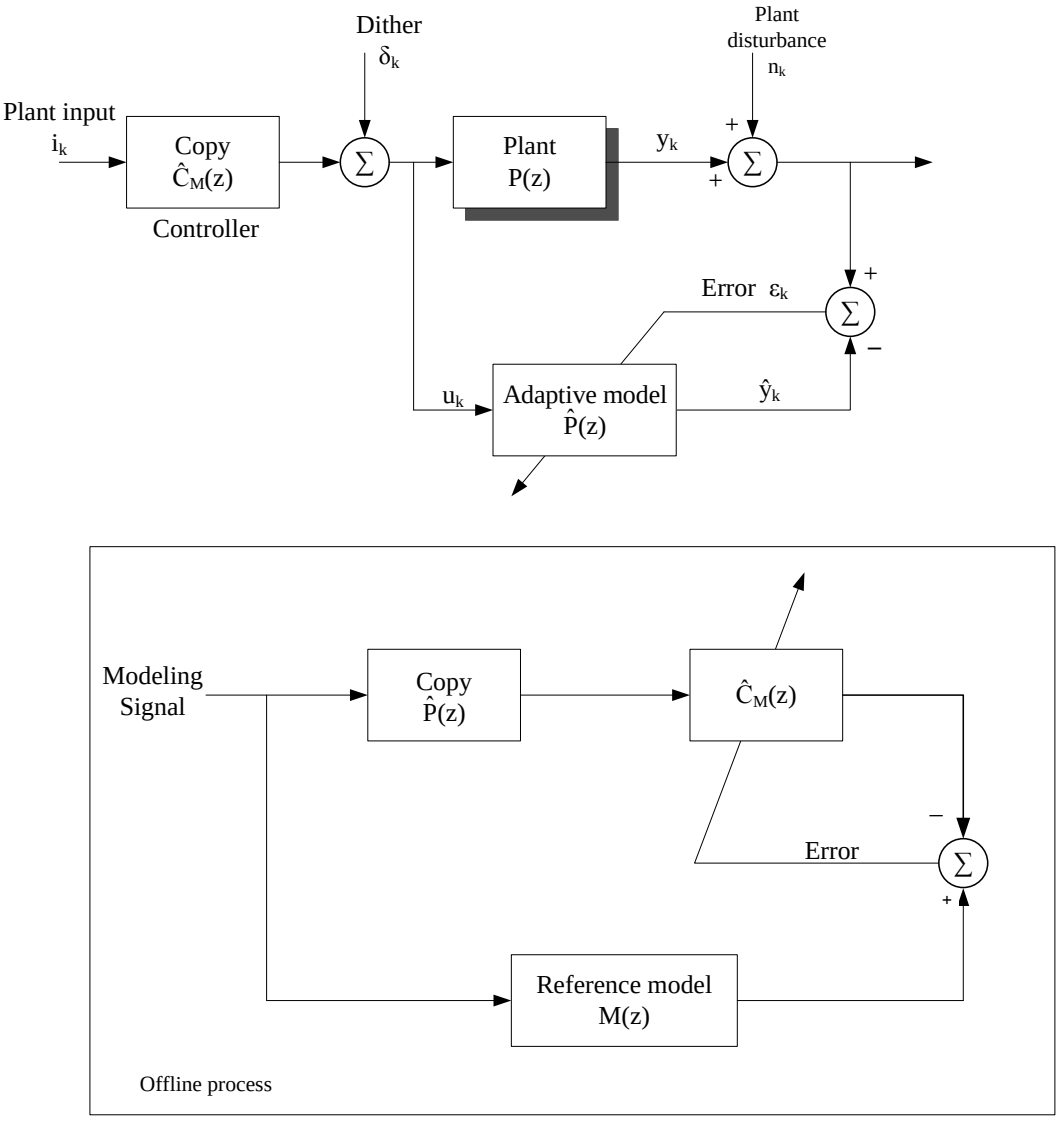


Figure 5. 3 An AIC system with offline inverse modeling for controlling plants with disturbance

The system of figure 5.2 works well as long as there is no plant disturbance. If plant disturbance is present, its effect is to bias the Wiener solution so that $\hat{C}(z)$ will no be a proper controller. The disturbance that appears at the plant output adds a component to the covariance of the input signal of the adaptive inverse model, directly affecting the Wiener solution for $\hat{C}(z)$ [1].

To overcome such problems, the scheme of figure 5.3 can be utilized. The plant model is formed and a digital copy of it is used to form the inverse model in an offline process. This technique was previously introduced in chapter 4. The idea is that $\hat{P}(z)$ has the same dynamic response as $P(z)$ but without disturbance. In direct modeling process plant disturbance dos not affect the Wiener solution.

Schemes introduced above cannot provide precise control in existence of plant disturbances. Their objective is to cancel the plant dynamics by obtaining a proper inverse but does nothing about canceling the disturbance. Closed-loop adaptive inverse controllers are used to cancel the disturbances. An adaptive inverse control scheme with a disturbance canceling feedback is introduced in figure 5.4.

Direct model is generated in an online process. As this model will be free of disturbances, output from a direct model copy is subtracted from the plant output to obtain an estimate of the disturbance. An inverse model is obtained from the direct model copy in an offline process. This inverse is used as the controller in feed forward and as the disturbance canceller block in the feedback. Disturbance estimate is passed through the disturbance canceling block and subtracted from the plant input. Disturbance canceling block is not activated before direct model and inverse model formed.

This scheme is utilized in chapter seven to investigate ball and beam experiment under the effects of disturbances.

An AIC simulation for the system given in equation 4.1 is made using the scheme which is depicted in figure 5.3. Plant given in equation 4.1 is used as reference model. Command tracking with and without reference model is given in figures 5.5 and 5.6 respectively. System iterated 20000 steps with added dither signal. If dither signal is omitted direct model never converges, thus the controller cannot function.

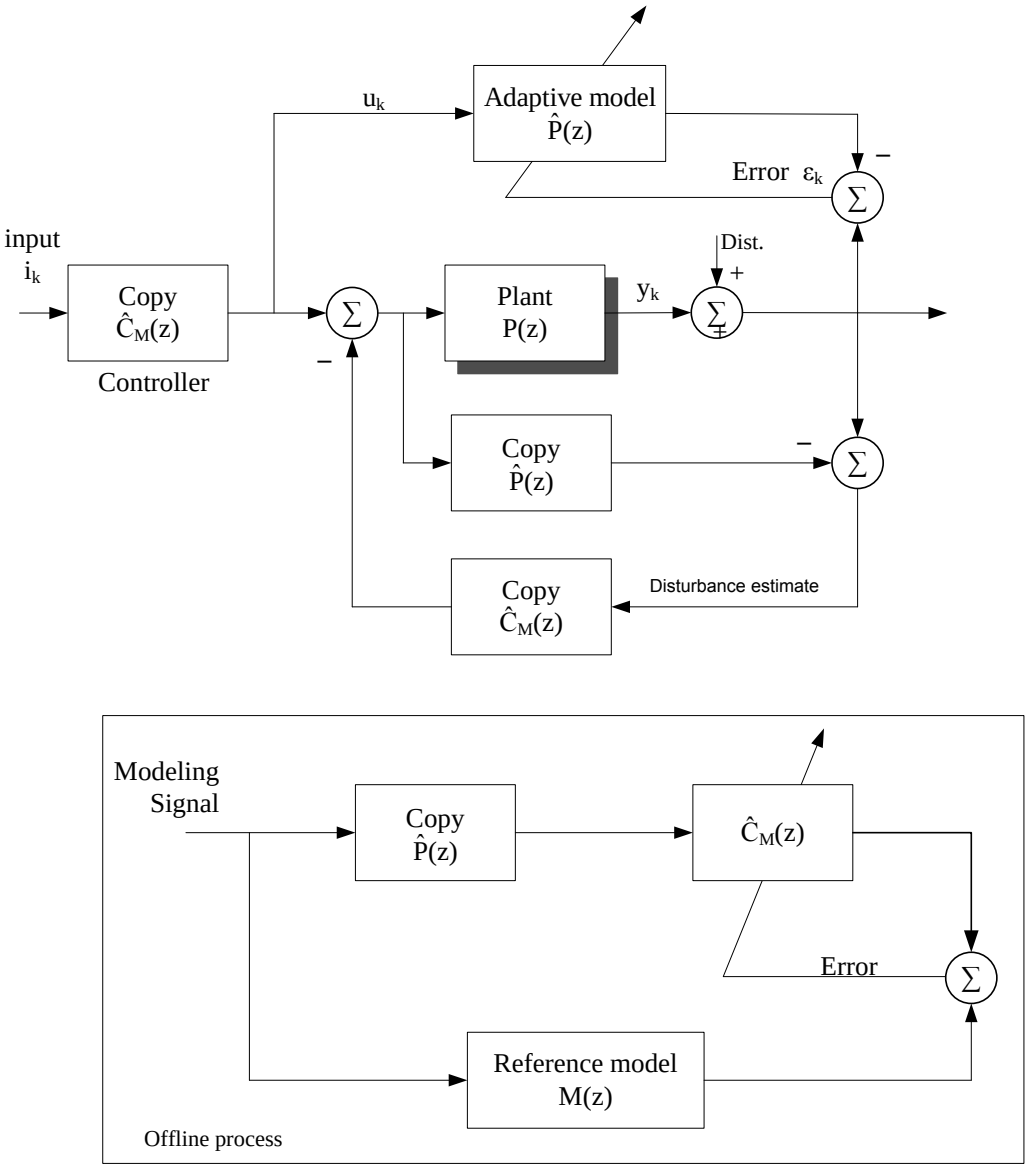


Figure 5. 4 Adaptive inverse control with disturbance canceling block

In another simulation only first 10 samples of the input signal are dithered. This helped the system to form the direct model and so the controller is functioning. But the result is not as good as dithered system. This can be seen on figure 5.7. In the other simulations with dither signal, adding dither signal is stopped at last steps to see command tracking performance clearly. For the simulations an alternative way to excite all the system modes is to make the controller input signal in different sections. Early sections of the signal would be rich in frequency content and the last sections would be step or square input, thus system's response can be clearly examined.

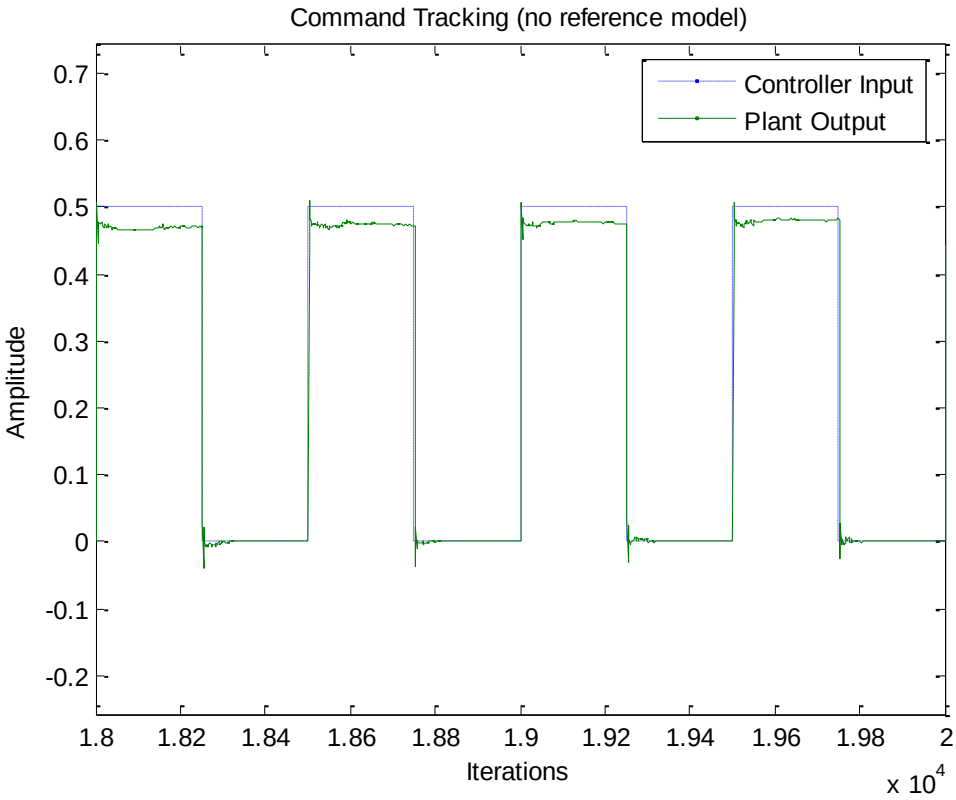


Figure 5. 5 Implementation of figure 5.3 without reference model

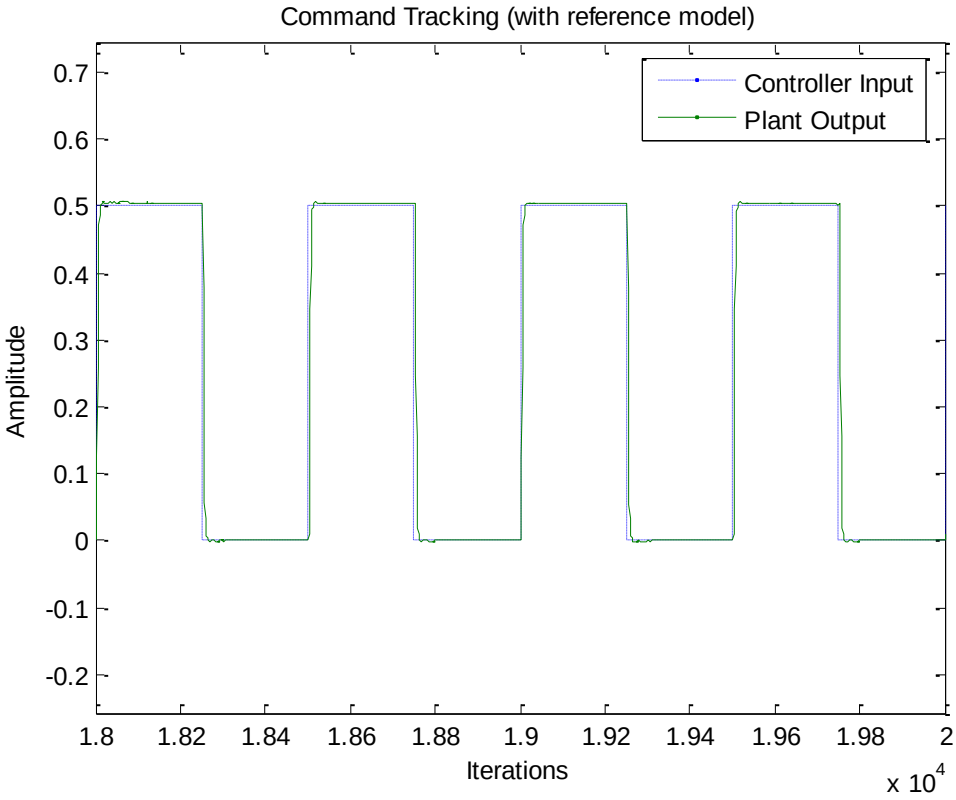


Figure 5. 6 Implementation of figure 5.3 with reference model

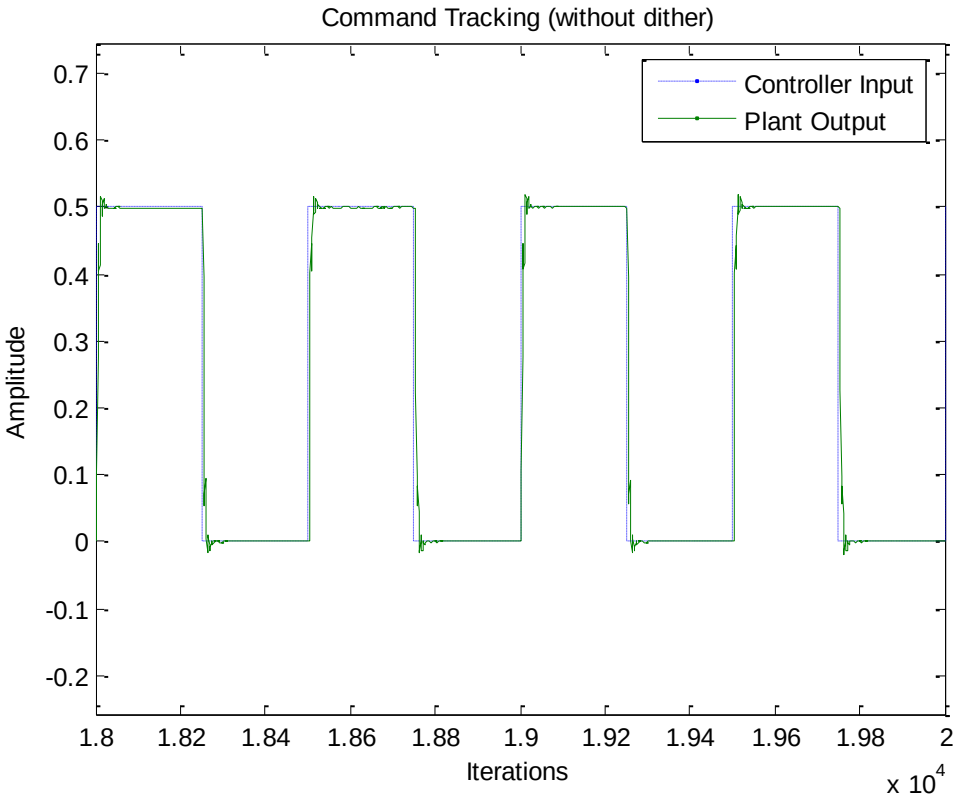


Figure 5. 7 Implementation of figure 5.3 (Only first 10 samples are dithered)

6. PARAMETERS OF ADAPTIVE INVERSE CONTROL SYSTEMS

Up to this chapter, usage of adaptive filters in direct modeling, inverse modeling and inverse control were examined with simulations but the parameters affecting the system are not taken into account. The parameters of adaptive filters and the signals in these systems have important role in application as well as the system itself. Playing around the parameters may greatly enhance the performance of the AIC system. Throughout this chapter, the effects of convergence factor, number of the filter weights and the modeling signal will be investigated using the conventional LMS and normalized LMS algorithm. A comparison of conventional LMS and normalized LMS algorithm is pointed out by this chapter

Effects of the parameters will be investigated over direct and inverse modeling processes. Direct and inverse modeling processes always take place in AIC systems. Examining the parameters such as the convergence factor, weight vector length or modeling signal characteristics on modeling processes makes determination of these parameters easier for the control system. An AIC system will not function properly without a successful modeling. If the modeling process is successful with a set of parameters than this parameters can be used in the AIC system. The better the model is the efficient the AIC system is.

Parameters are examined on the system whose transfer function is given is equation (4.1). It is given in digital filter form. It is a stable and minimum-phase plant and its impulse response is plotted on figure 6.1.

Settling time of the system is approximately 3 seconds. For a sampling period of 0.1 seconds, setting the adaptive filter weights to 30 would suffice for accurate modeling of the system

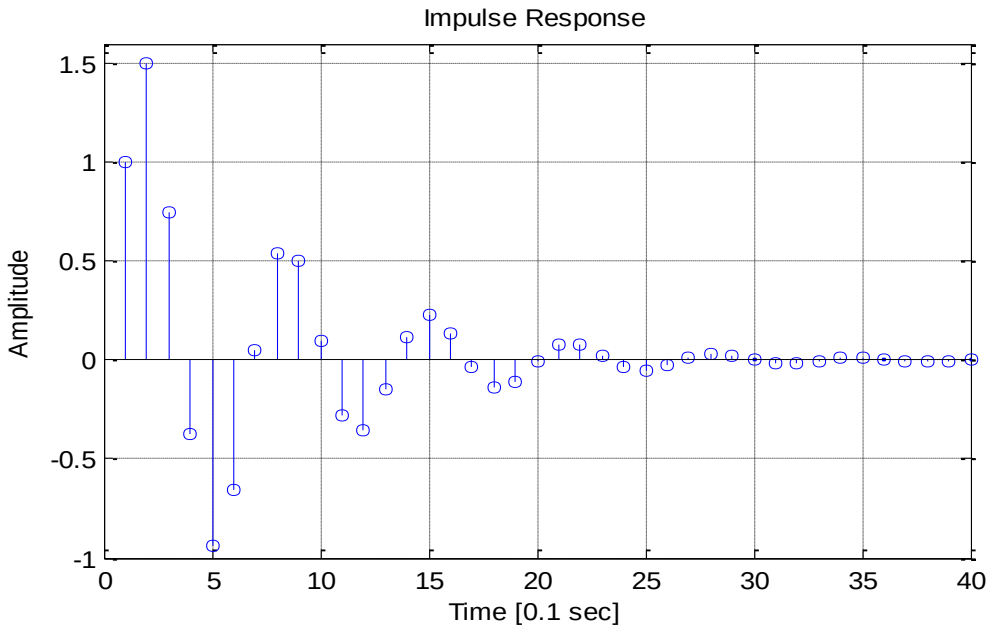


Figure 6. 1 Impulse response of P (z)

A good fit of the plant is plotted on figure 6.2. For this simulation convergence factor was set to 0.001. Modeling signal was a band limited white noise with power of 0.1 and sampling period was 0.1. A definite effect of weight number selection can be easily seen at first sight that the modeled systems response does not contain any information after 3 seconds. The sampling period was chosen to be 0.1 seconds, so with 30 weights, memory time of 3 seconds can be modeled.

$$\text{Memory time} = \text{Sampling Period } x \text{ Number of Weights}$$

With the same parameters used in direct modeling, an inverse modeling process has been run. Although 10000 iterations is well enough for direct model to converge, it is not adequate for inverse modeling process to form a good fit. This situation is illustrated in figure 6.3 by plotting the squared error for both direct and inverse modeling processes. With this set of parameters inverse modeling process needs to be iterated more to form a better model. This shows that direct and inverse modeling processes are two distinct processes and their implementation must be handled separately.

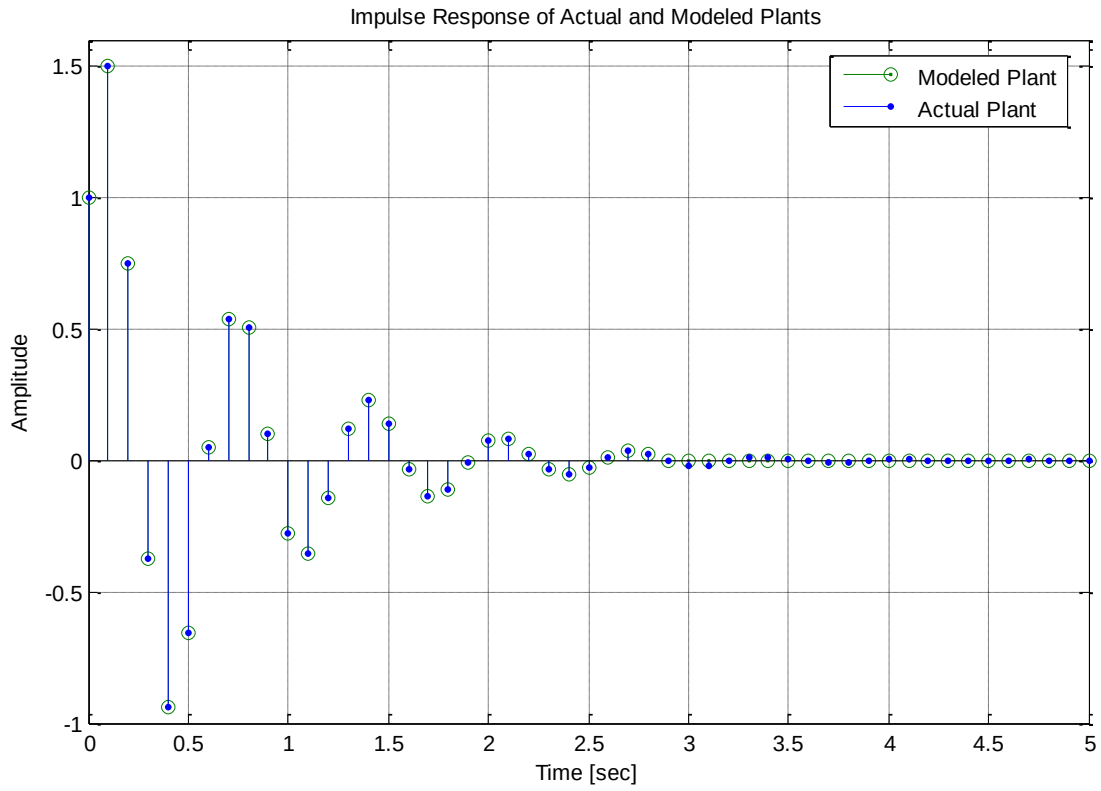


Figure 6. 2 Impulse response of actual and modeled plants

.

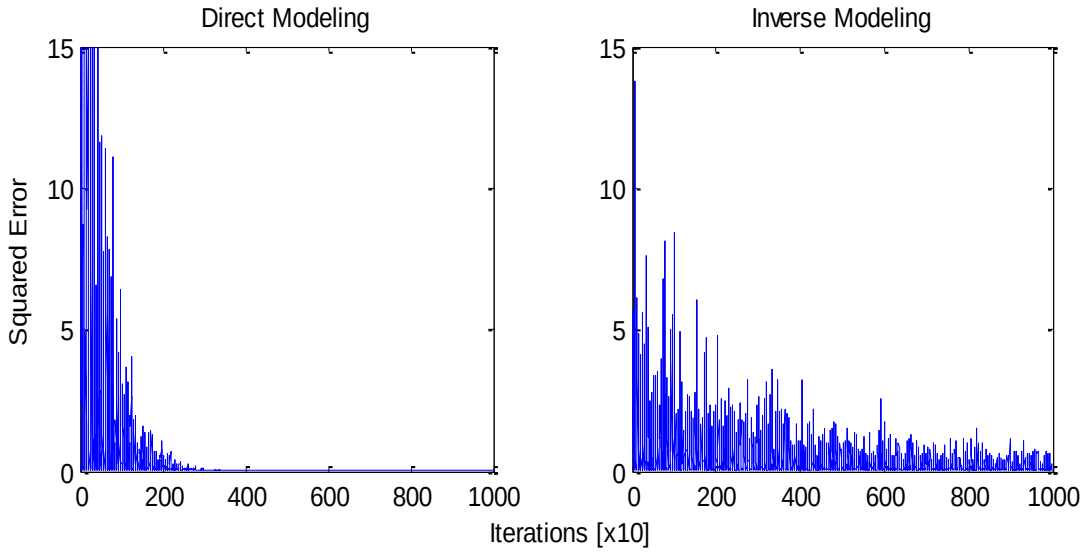


Figure 6. 3 Squared error for direct and inverse modeling processes

6.1 Effects of Convergence Factor on Modeling Processes

To investigate the effects of convergence factor μ, all of the other conditions are fixed and the simulations were run with changing values of μ. Observing the change of weights during simulation is a good way for understanding the simulation convergence time and the noise in the weights. In this study first, middle and the last weights are observed to see the effects of μ on weights.

In figure 6.4 the change of weights is illustrated with the convergence factor being 0.001. A vertical line at 4000^{th} iteration is plotted on this figure. This is approximately the time when weights converged to their final value.

Variations of the weights are calculated as a measure of noise. Variation value is used to compare noise levels for different values of μ. Results for $\mu = 0.005$ and $\mu = 0.0005$ are plotted on figures 6.6 and 6.7 respectively. Variations for $\mu = 0.005, 0.001, 0.0005$ are listed in table 6.1. Figure 6.4 shows the increase in the weight noise with increasing μ values

Table 6. 1 Variation values for weights after convergence

	Variations (as a measure of weight noise)		
μ	First weight	Middle weight	Last weight
0.005	0.00000452	0.00000325	0.00000329
0.001	0.00000371	0.00000141	0.00000148
0.0005	0.00000150	0.00000003	0.00000005

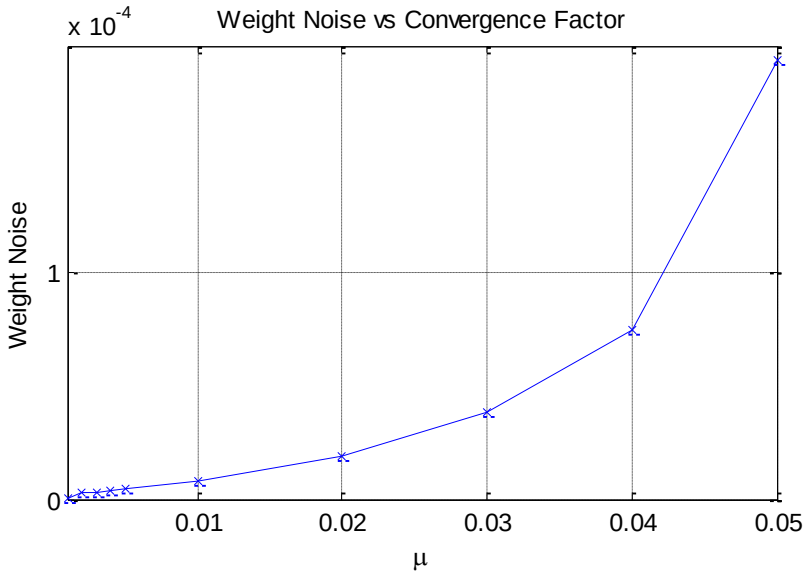


Figure 6. 4 Effects of μ on weight noise

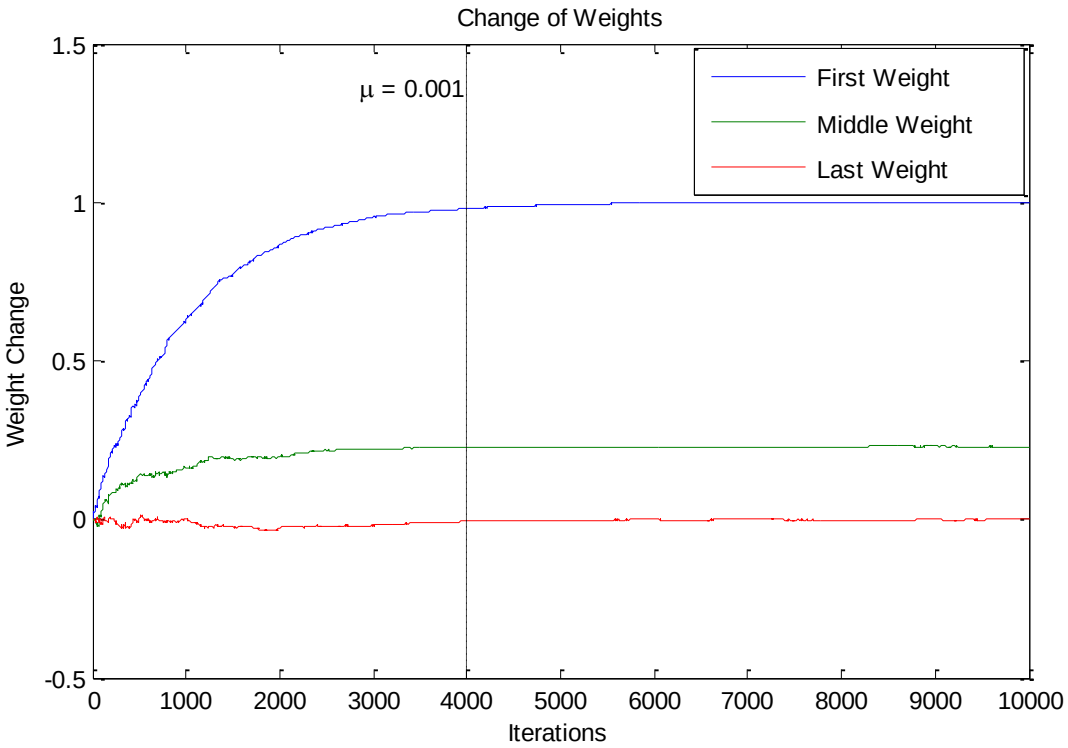


Figure 6. 5 Change of weights for μ=0.001

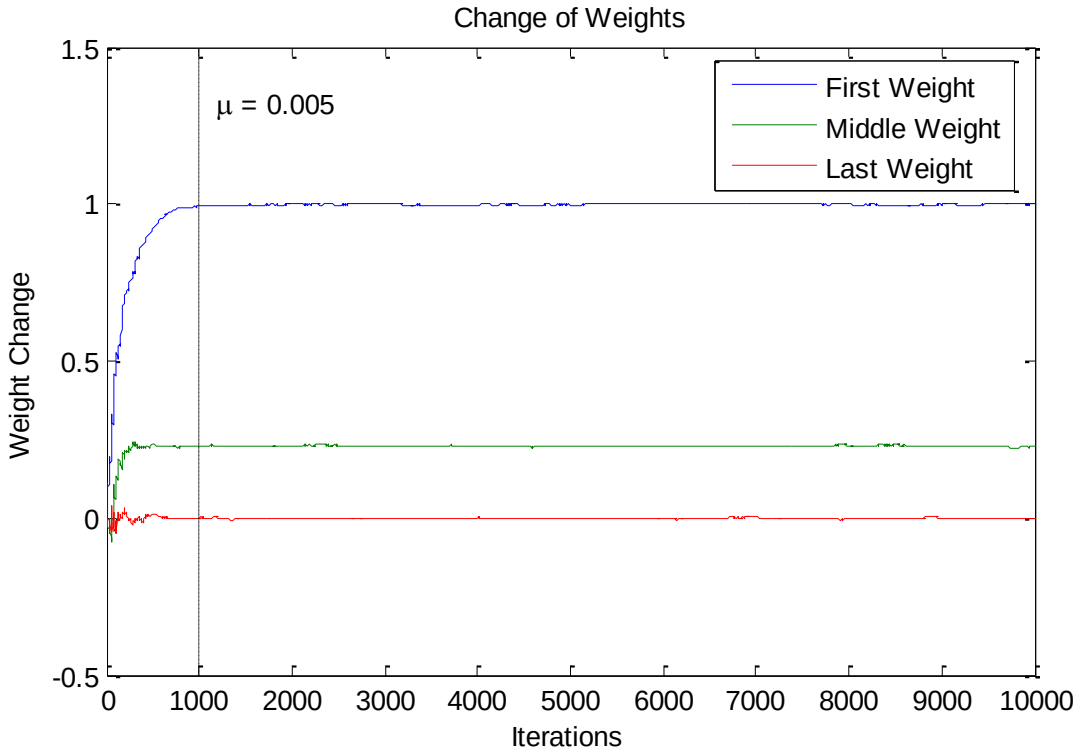


Figure 6. 6 Change of weights for μ=0.005

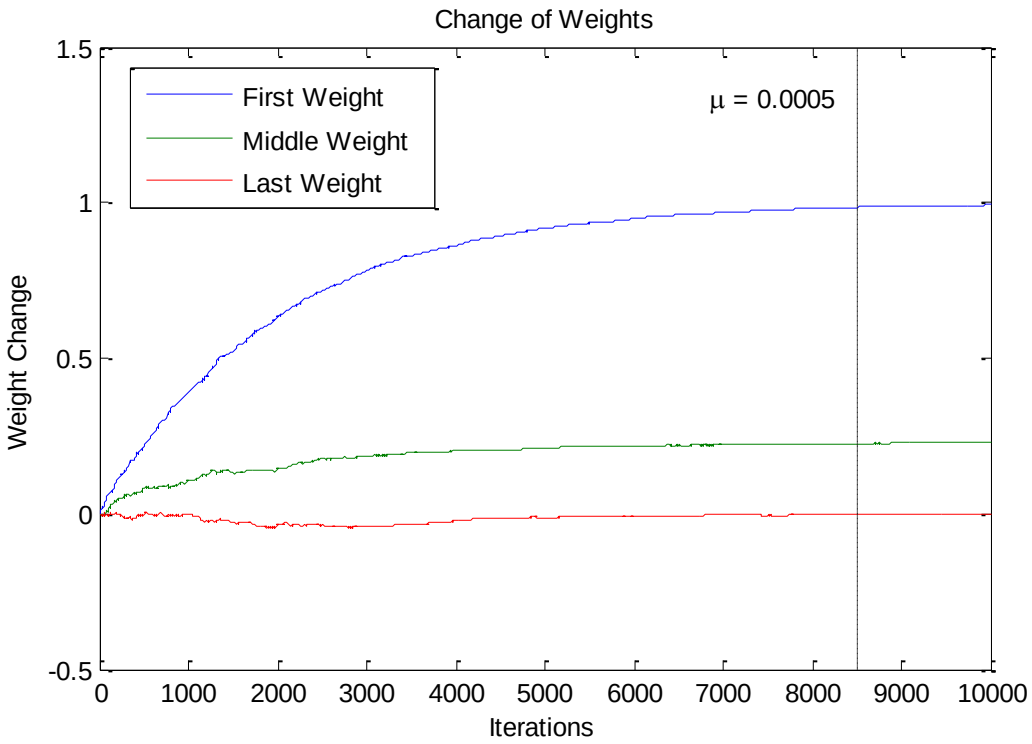


Figure 6. 7 Change of weights for μ=0.0005

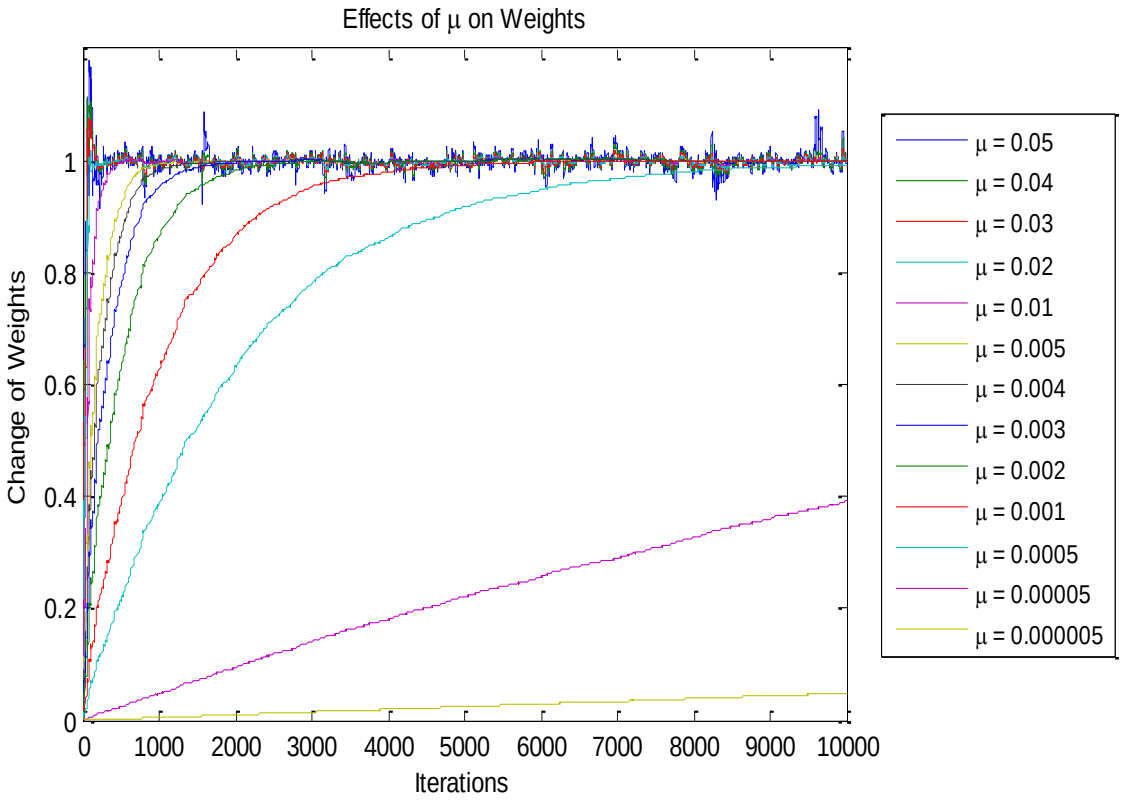


Figure 6. 8 Effects of μ on weights

Finally, to see the effects clearer, direct modeling process was run with 13 different values of μ from 0.5 to 0.000005. Convergence of the first weight in the system is plotted on figure 6.8 for these values of μ. Values bigger that 0.05 makes the process unstable, weights never converge and values smaller than 0.0005 causes the process to converge very slowly, it needs many more iterations to gets its final value.

As a result of these plots, it is obvious that larger values of μ causes the process converge faster but result in noisy weights, smaller values of μ yields a slower process but more steady weights.

6.2 Effects of Weight Vector Length on Modeling Processes

Obtaining a meaningful plant model with adaptive modeling will be impossible unless the FIR filter can cover the most important part of the systems impulse response. This can be achieved if the product of $wn \times Ts$ is bigger than system's settling time, where wn is weight number and Ts is the sampling time of the system.

For a fixed value of sampling period, selecting too many weights causes the last weights to become zero. This situation approves the sufficiency of the weights. For a fixed value of μ choosing a very long weight vector prevents convergence. Convergence factor needs to be decreased in this case. Plant given in equation (4.1) is directly modeled with 60 weights and results are plotted on figure 6.9. Impulse response of the modeled system is a perfect fit of the actual system. As seen on the graphs last 20 weights tends to zero, at the end of the simulation. So it can be said that last 20 weights of the filter is unnecessary for this plant. Omitting unnecessary enables working with larger μ values. For 50 weights $\mu = 0.001$ would suffice but 250 weights requires $\mu = 0.0005$.

Same system modeled with 15 weights and the results are depicted in figure 6.10 and 6.11. Modeled plant does not fit the actual plant and weights are very noisy. This is an expected result because with the sampling period of 0.1 seconds, only 1.5 seconds of the plant's response can be handled with 15 weights.

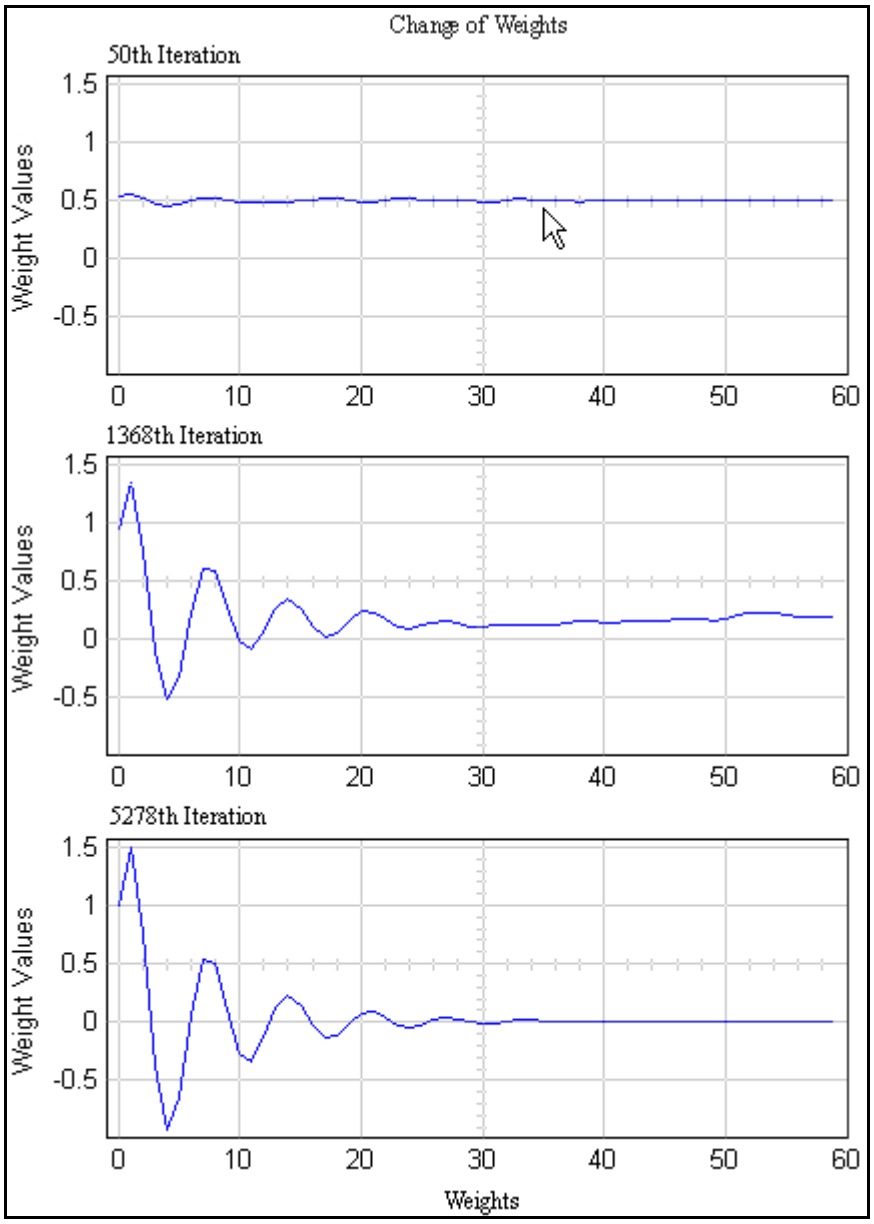


Figure 6. 9 Selecting too many weights

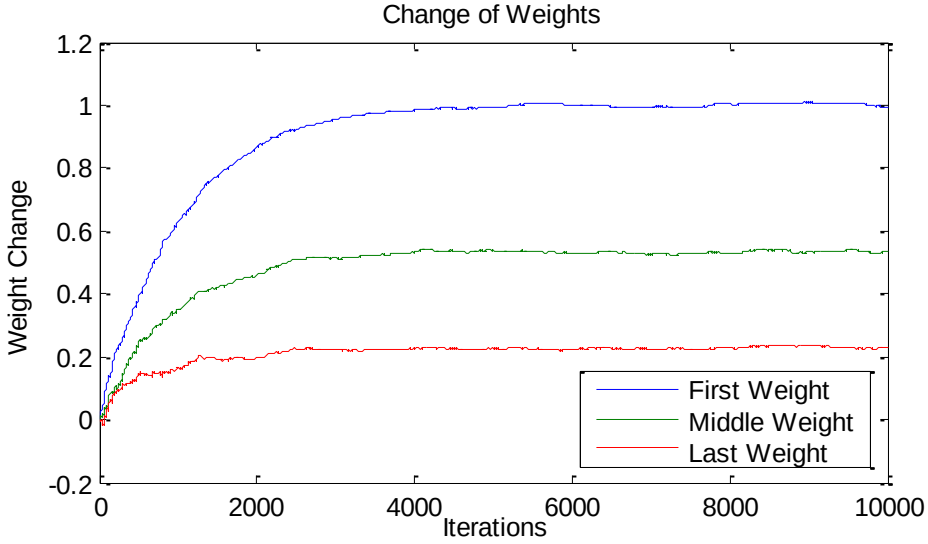


Figure 6. 10 Selecting insufficient weights

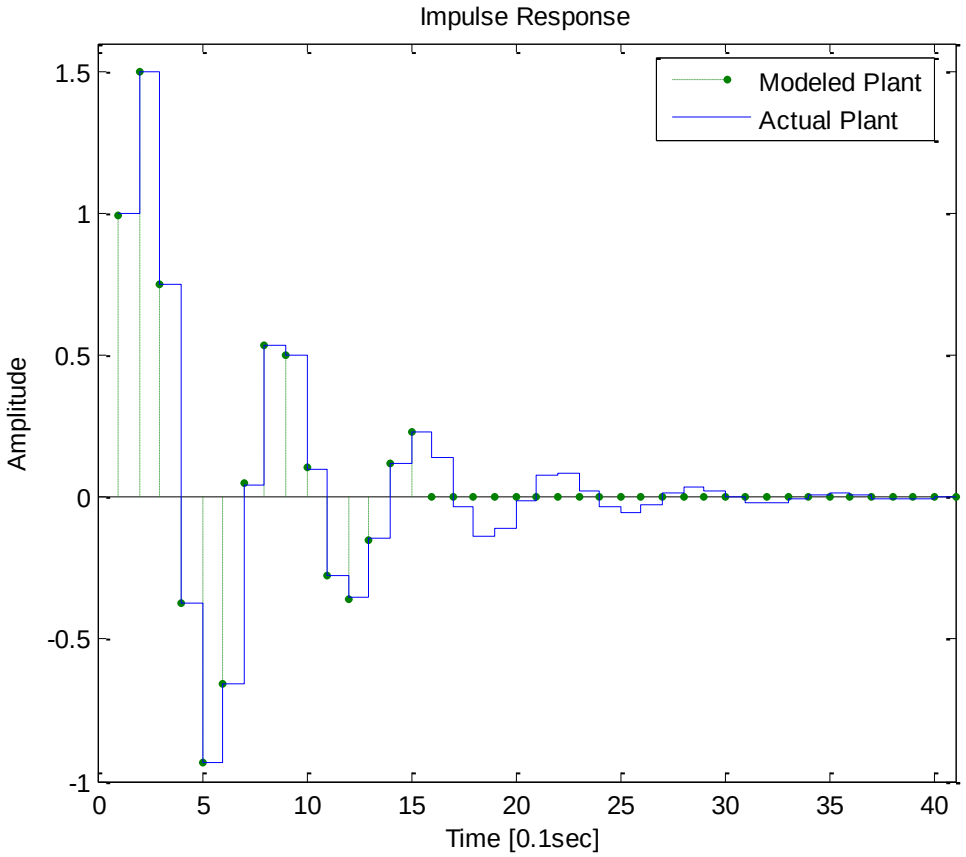


Figure 6. 11 Selecting insufficient weights

6.3 Effects of Modeling Signal Characteristics

Statistical properties of the modeling signal have significant role over the success of the modeling process. Two main criteria are the frequency content and the power of the modeling signal.

Simulations had been run with modeling signals which were poor in frequency content and the results were disastrous. Signals that contain frequencies around and less than the bandwidth of the plant are never enough for a modeling process. To ensure adequacy of frequency content of the signal it must be much greater than the fastest dynamics of the system. In literature it is advised to be 30 times greater than the bandwidth of the plant. This is approved with various simulations that it is quite enough.

Throughout the simulations it was seen that the LMS algorithm is highly sensitive to the power of the input signal. For the system given in equation 4.2 LMS with μ=0.001 is stable unless the power of the input signal is below 39. After this value

LMS goes unstable and never converges. Figure 6.12 shows two step responses of models, one modeled with a signal of power 38 and the other 39. Although a signal of power 38 can model accurately signal of power 39 can not. This situation can be alleviated with decreasing the value of convergence factor. Step response of model generated with same signals but with smaller μ are plotted on figure 6.13

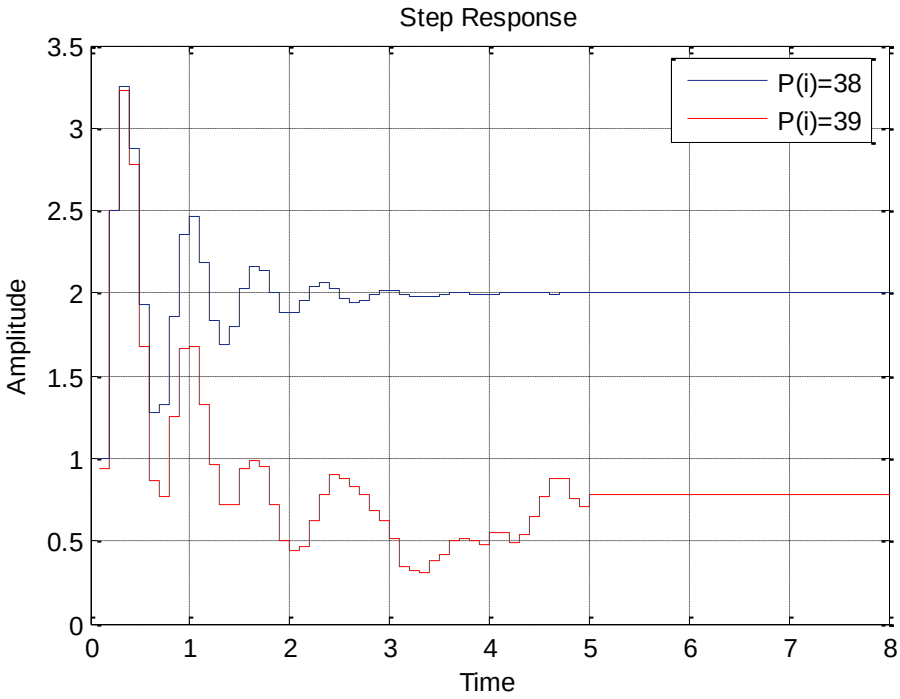


Figure 6. 12 Modeling with μ=0.001

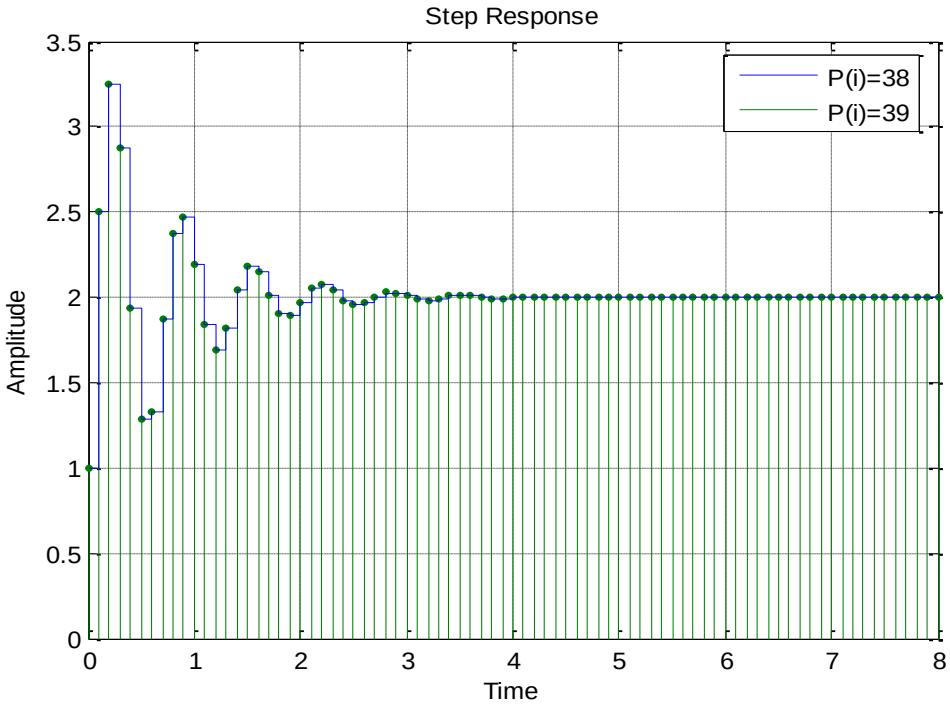


Figure 6. 13 Modeling with μ=0.0005

As a result it can be said that power of the input signal have a great effect on selecting the convergence factor and thus on the speed of the adaptation. As the power of input signal increased the convergence factor should be decreased to avoid instability of the LMS algorithm. According to the results obtained in section 6.1 this causes slow adaptation.

6.4 Using Normalized LMS in Modeling Processes

In section 6.3 it is shown that convergence factor needs to be scaled by the power of the input signal. This may be a problem if there is no prior knowledge about the input signal power.

Using normalized LMS omits such problems. As the convergence factor is scaled by the input signal power during the process, any change in the power will not cause the LMS to be instable.

Comparison results of these two algorithms are given in figures 6.13 and 6.14. Although conventional LMS is not working with μ=0.001 after passing the power 38, normalized LMS works very well even at power 50 and with much greater μ. It is also seen that normalized LMS converges faster as a result of greater μ values.

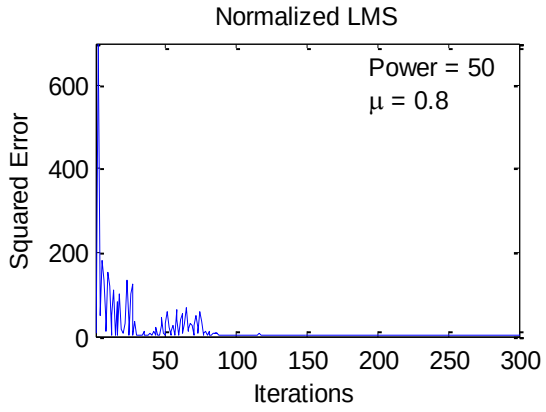


Figure 6. 14 Using Normalized LMS

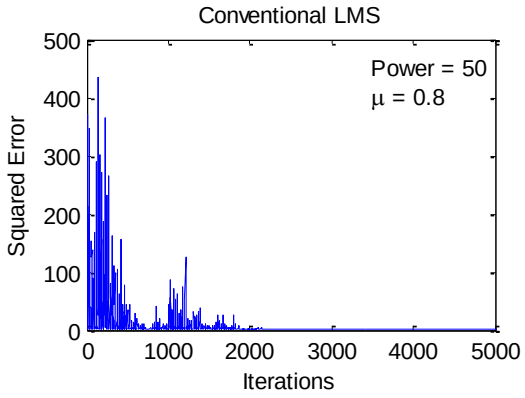


Figure 6. 15 Using Conventional LMS

7. ADAPTIVE INVERSE CONTROL APPLICATIONS

Control of time-varying plants and plants with disturbances are two favorite application areas of adaptive inverse control. In the first section of this chapter performance of AIC system is examined on a plant with time-invariant and time-varying cases. In the second section ball-beam experiment is examined with various disturbance effects.

7.1 AIC of Time-varying Plants

One of the most charming application areas of adaptive control is time varying systems. In this section both time-invariant and time-varying cases of the system given in equation 7.1 is examined with adaptive inverse control scheme given in figure 5.2.

$$\frac{y_p(s)}{u_p(s)} = \frac{2}{(s+1)} \frac{229}{(s^2 + as + 229)} \qquad \textbf{(7.1)}$$

Here $a = 30$ for time-invariant case and $a = 30 + 5\sin(2t)$ for time-varying case. It reflects the change of the damping ratio. The output of this plant was required to follow the output of the reference model in equation 7.2 [6].

$$\frac{y_m(s)}{u_m(s)} = \frac{1}{(1+\frac{s}{3})} \qquad \textbf{(7.2)}$$

A controller for this system was designed with MRAC method. Figure 7.1 shows plant and reference model outputs from the previous work [6].

Step and impulse responses of both time-invariant and time-varying cases of the plant are plotted on figure 7.2 and 7.3. System is stable and settling time is approximately 5 seconds.

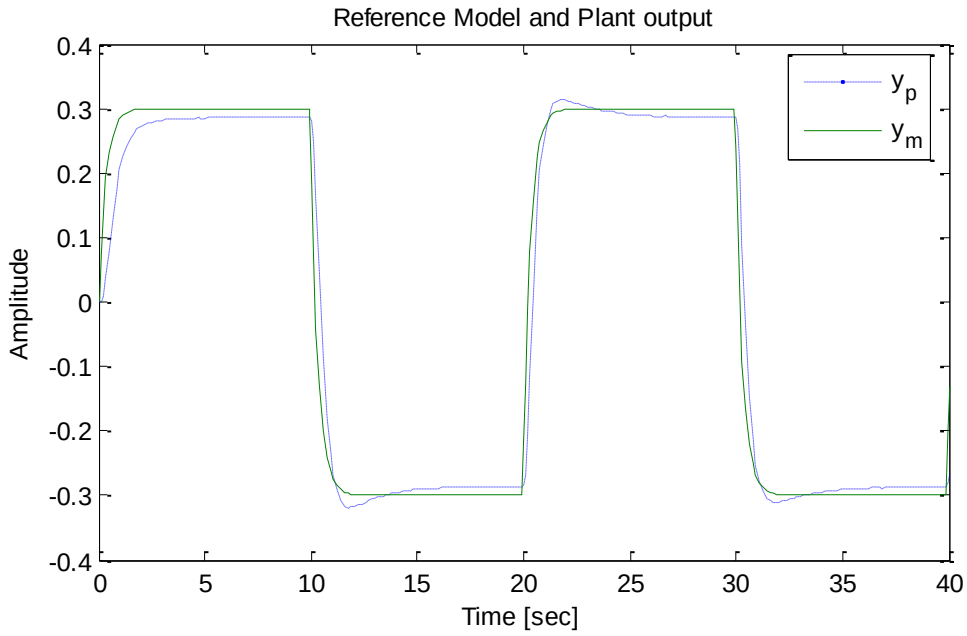


Figure 7. 1 MRAC output from ref [6]

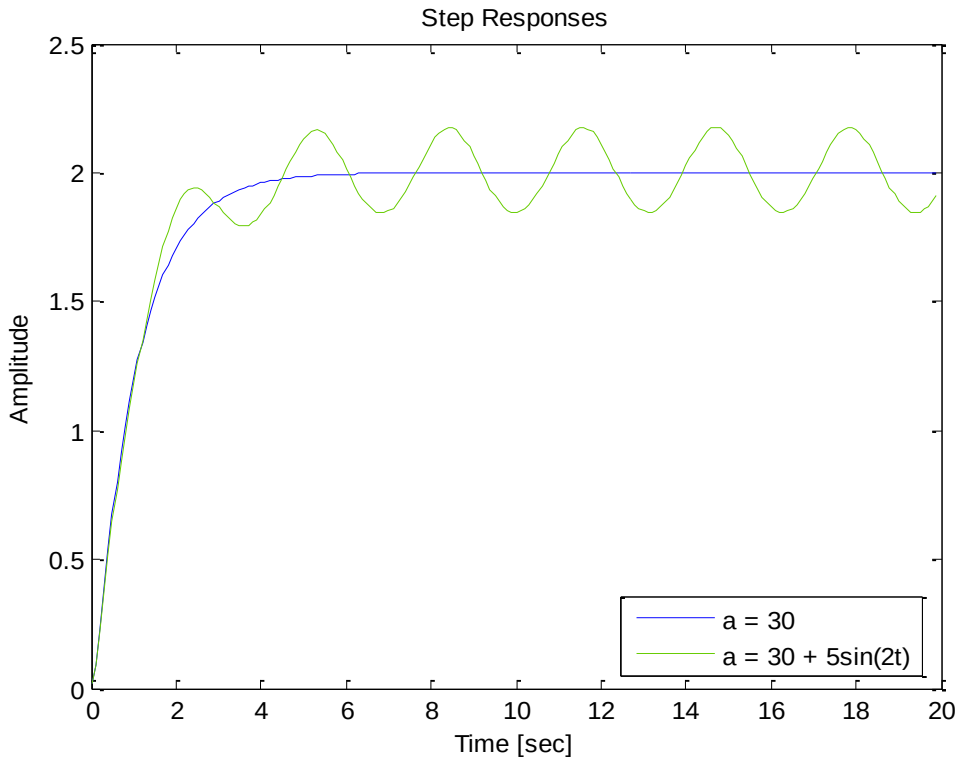


Figure 7. 2 Step responses of time invariant and time varying cases

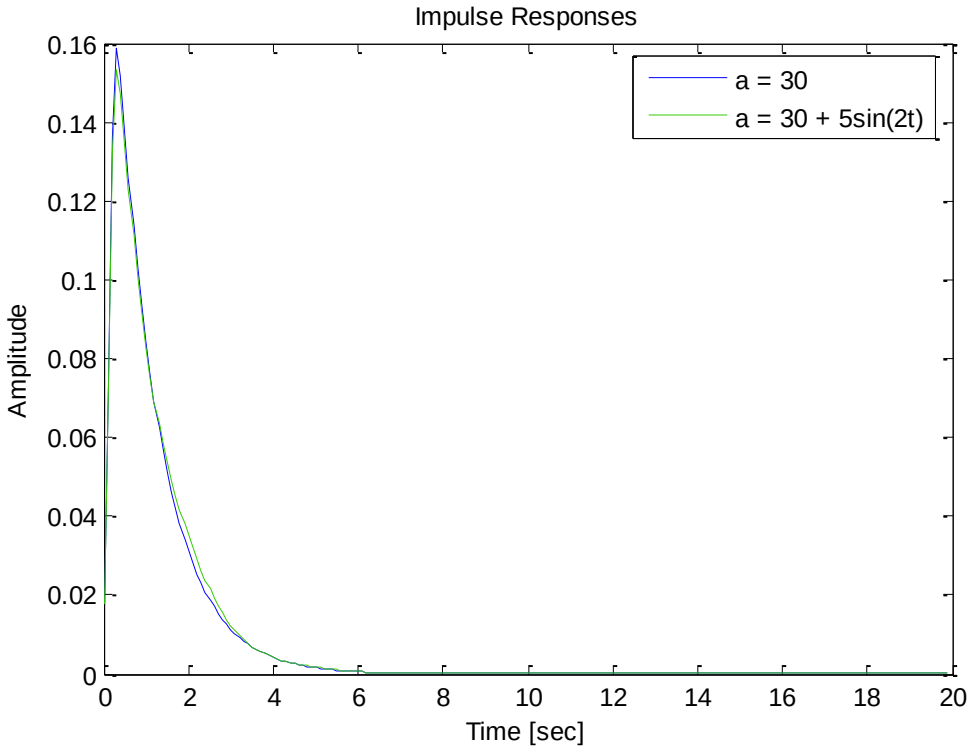


Figure 7. 3 Impulse responses of time invariant and time varying cases

Sampling time of the overall process is set to 0.1 seconds. For handling the memory time of the system FIR filter length is set to 50.

The selected parameters are checked with modeling simulations first. And it is seen that adding a unit delay to the desired response greatly enhances the inverse modeling process.

If the analog to digital transformation is done within the loop it causes the simulation to run very slow. To make the simulation of time-varying case faster, the system given in s-domain is converted to z-domain by bilinear transformation method.

The method is also called the Tustin transformation, or trapezoid approximation. The transformation equation is:

$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right) \tag{7.3}$$

The bilinear method is the most commonly used method for convert controllers into the discrete time domain [8]. In equation 7.3 T is the sampling time which is taken 0.1 seconds in this transformation. By substituting equation 7.3 in equation 7.1 and doing algebraic simplifications, the discrete form of time-varying plant is obtained. Equation 7.4 shows the discrete form the time-varying plant. Coefficients of the filter

are normalized by the value of denominator's first coefficient within the program, this is necessary to avoid overflow error in Matlab.

$$\frac{y_p(z)}{u_p(z)} = \frac{458z^3 + 1374z^2 + 1374z + 458}{(420a + 13209)z^3 - (380a + 19133)z^2 + (19707 - 420a)z + (380a - 11951)} \quad \textbf{(7.4)}$$

Weight adjustment of the controller is done with normalized LMS and μ is set to 0.1. A total iteration step is 10000 for all simulations. Within these conditions very good control over the system is achieved. System's response to a square-wave input has shown no overshoot and entered the tolerance band within 3 seconds and did not exceed it anymore. Steady state error of the system is also zero. Figure 7.4 shows the response of the time-invariant plant to square wave input.

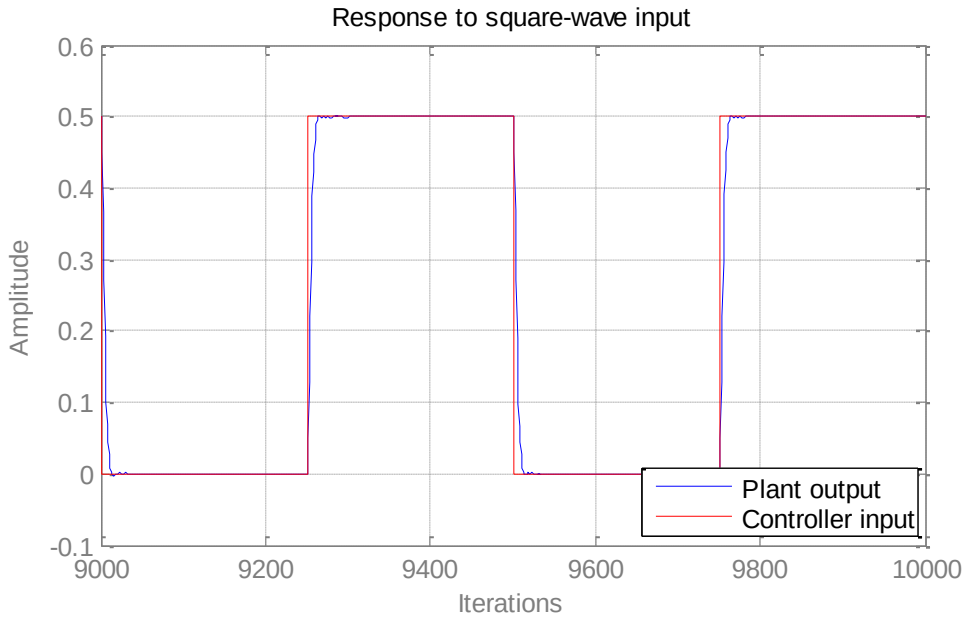


Figure 7. 4 Control of time-invariant plant

Control system error of time-invariant plant is plotted in figure 7.5. It shows peaks when the signal state chances. Zero steady state error can be seen here.

Parameter a in the system 7.4 is subjected to a linear change from 30 to 33 during the simulation. Systems response to square-wave input is shown on figure 7.6. There are overshoots in the response and the settling time is increased. System is still under control and the steady state error is zero. Increased control system error can be seen in figure 7.7.

Figure 7. 5 Overall control system error of time-invariant plant

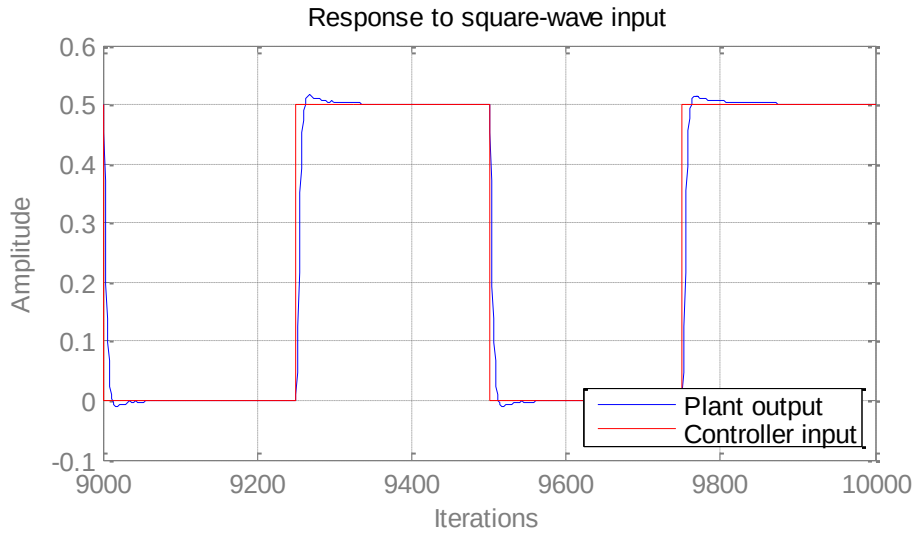


Figure 7. 6 Control of time-varying plant, parameter a changes 30 to 33

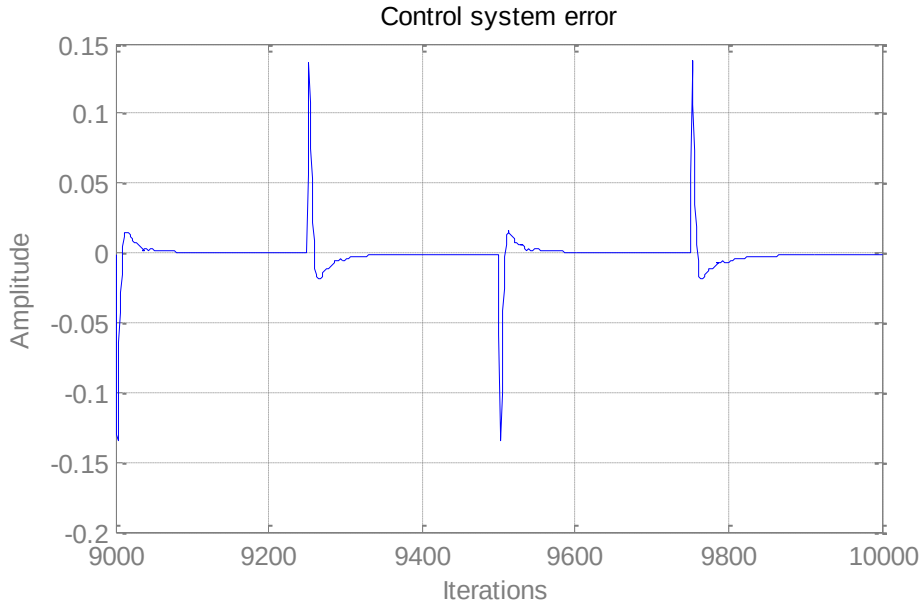


Figure 7. 7 Overall control system error of time-invariant plant

Increasing the changing range causes the AIC system to fail. When parameter a is subjected to change from 30 to 34 during the simulation, performance of the AIC system greatly decreased. Figure 7.8 shows how the system fails to control the plant.

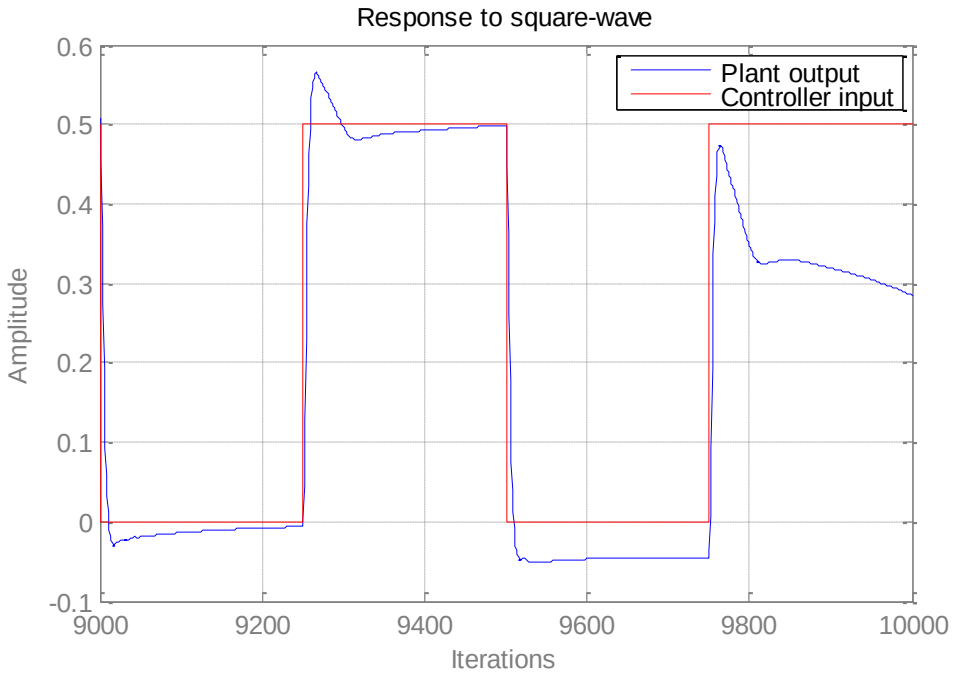


Figure 7. 8 Control system failure in case of fast changing dynamics

Control system's response to step input is plotted on figure 7.9. Here the plant is subjected to sinusoidal changes with frequency of 5 rad/s. The response is oscillatory and these oscillations never die out with parameter configurations. Lowering the frequency increases the control performance.

Simulations with different time-varying modes of the plant have shown that oscillatory changing plant dynamics can be controlled within a frequency range that the AIC system may adapt itself. Increase in the amplitude and the frequency of the oscillations increases the control system error. Amplitudes of change in the parameter a is same in both simulations which resulted figure 7.9 and 7.10. But with the lower frequency simulation, amplitude of the error decreased. This shows the relation between frequency and control performance.

Simulations have shown that AIC system is capable of controlling slowly changing plants. If the change is fast then AIC fails to control the plant.

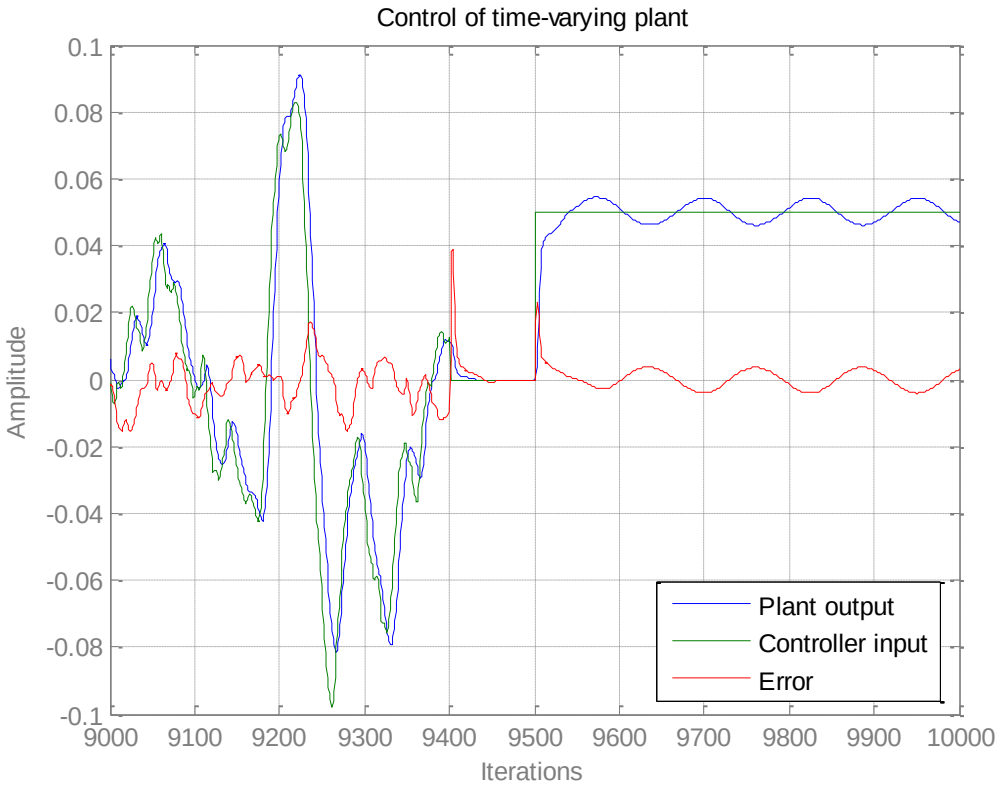


Figure 7. 9 Control of time-varying plant, f = 5 rad / s

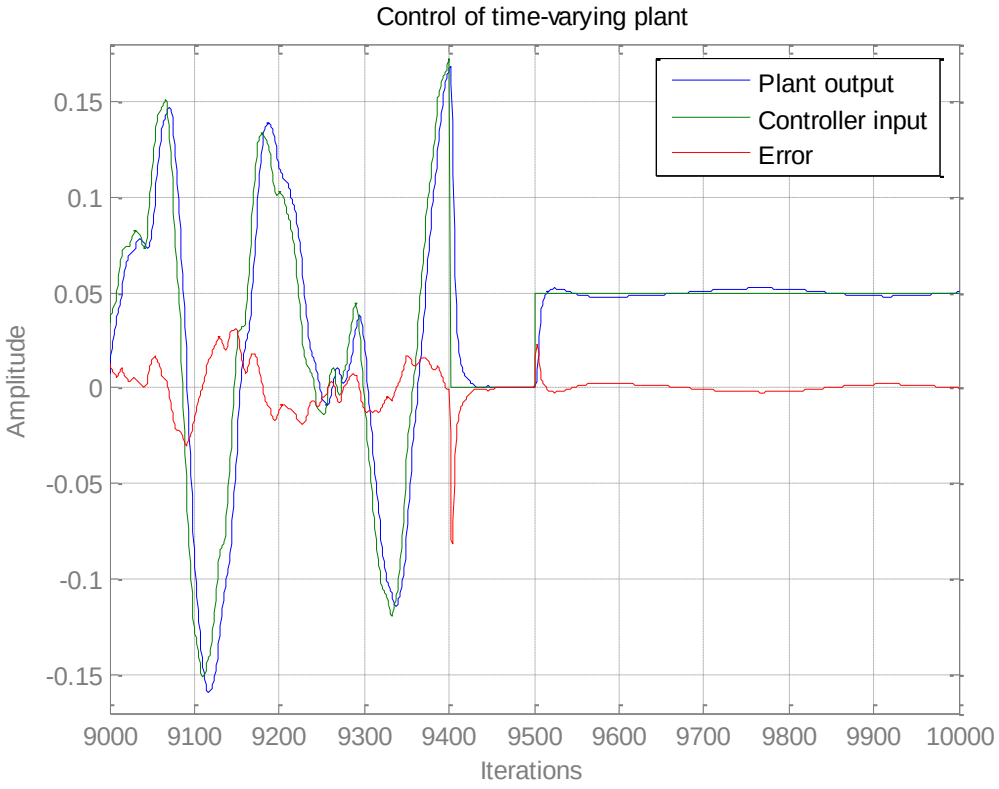


Figure 7. 10 Tracking performance of time-varying plant, f = 2 rad / s

7.2 AIC of Unstable Ball on a Beam in Existence of Disturbances

In this section ball and beam experiment[1] is examined in existence of disturbances. Illustration of the experiment system is depicted on figure 7.11. A ball is placed on a beam with 1 degree of freedom; it can roll along the length of the beam. The angle of the beam α is controlled with the rotation of the servo gear which is connected to the beam with a lever arm. Any change in the angle α causes the ball to roll along the beam. Position r is the controlled variable in this system. Disturbance canceling capabilities of adaptive inverse controller is tested with different disturbance types such as step, ramp and random.

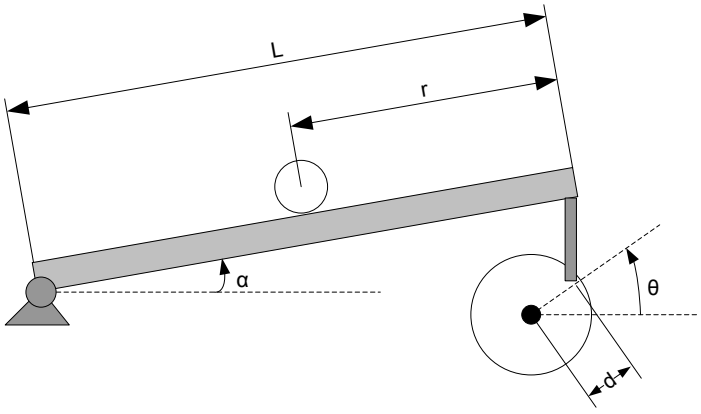


Figure 7. 11 Ball and beam system

Transfer function from the gear angle θ to the ball position r is given in equation 7.5. Constants and variables for this system are given as:

M, mass of the ball = 0.11 kg
R, radius of the ball = 0.015 m
d, lever arm offset = 0.03 m
g, gravitational acceleration = 9.8 ms^{-2}
L, length of the beam = 1.0 m
J, ball's moment of inertia = 9.99e-6 kgm^2
r, ball position coordinate
α, beam angle coordinate
θ, servo gear angle

[1] All of the information about the ball and beam experiment is taken from the website: http://www.engin.umich.edu/group/ctm/examples/ball/ball.html

$$\frac{R(s)}{\Theta(s)} = -\frac{mgd}{L\left(\frac{J}{R^2}+m\right)}\frac{1}{s^2} \qquad \textbf{(7.5)}$$

By substituting the given numerical values in equation 7.5, equation 7.6 is obtained.

$$\frac{R(s)}{\Theta(s)} = \frac{0.21}{s^2} \qquad \textbf{(7.6)}$$

System of 7.6 has two poles at the origin which makes it unstable. In order to apply adaptive inverse control it must be stabilized first. System is stabilized with a PD controller whose proportional gain is 20 and derivative gain is 10. The stabilized system's transfer function is given in equation 7.7.

$$\frac{2.1s+4.2}{s^2+2.1s+4.2} \qquad \textbf{(7.7)}$$

Settling time of the stabilized plant is approximately 4 seconds. Therefore with sampling time of 0.1 seconds, setting the weight vector length to 50 will be enough to cover the memory time of the plant. Normalized LMS algorithm is used for weight update, thus system converged within 3000 iterations. Total iteration number is selected 10000 for all simulations. Convergence factors are set to 0.01 during the first half of the simulations and then decreased to 0.001 to get smoother responses. A second order reference model with a settling time less than 3 seconds is used to smooth the transient response of the AIC system.

Simulation program for this application is based on the disturbance canceling scheme which is given in figure 5.4.

Control of the ball position is plotted on figure 7.12. There is no disturbance yet. AIC system is working very well; control error is approximately 2%.

At the 7500^{th} iteration system is disturbed with a 0.2 m step input. When the disturbance canceller block is turned off the system has lost its stability against the disturbance. Figure 7.13 shows the system's performance when the disturbance canceller is off. The effect of disturbance canceller block is seen on figure 7.14.

System is subjected to a continuous ramp disturbance starting from 7500^{th} iteration. Figure 7.15 depicts systems response against the disturbance. Performance is slightly decreased compared to the step disturbance.

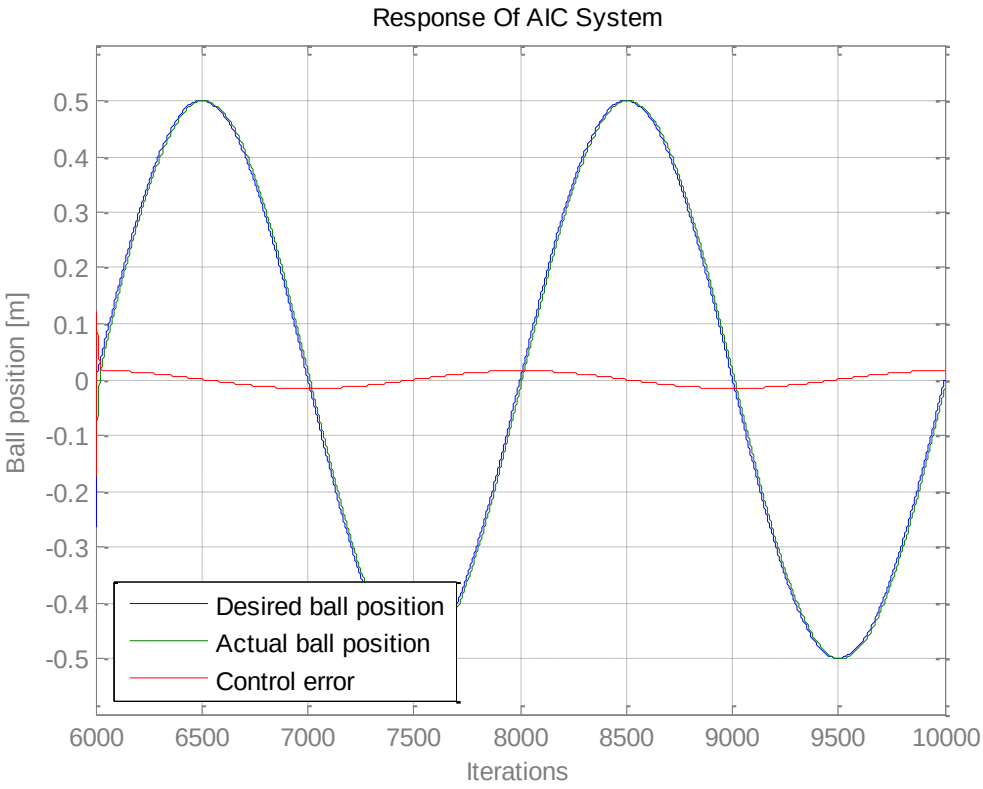


Figure 7. 12 Controlling ball position, no disturbance added

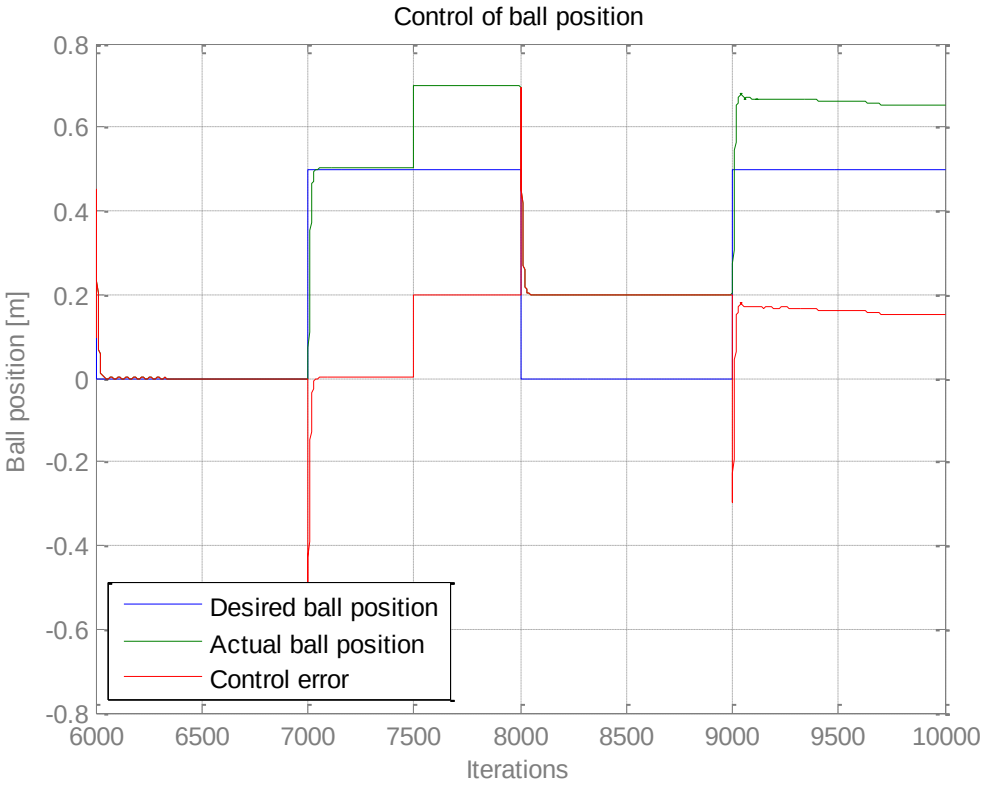


Figure 7. 13 Step disturbance (without disturbance canceller block turned on)

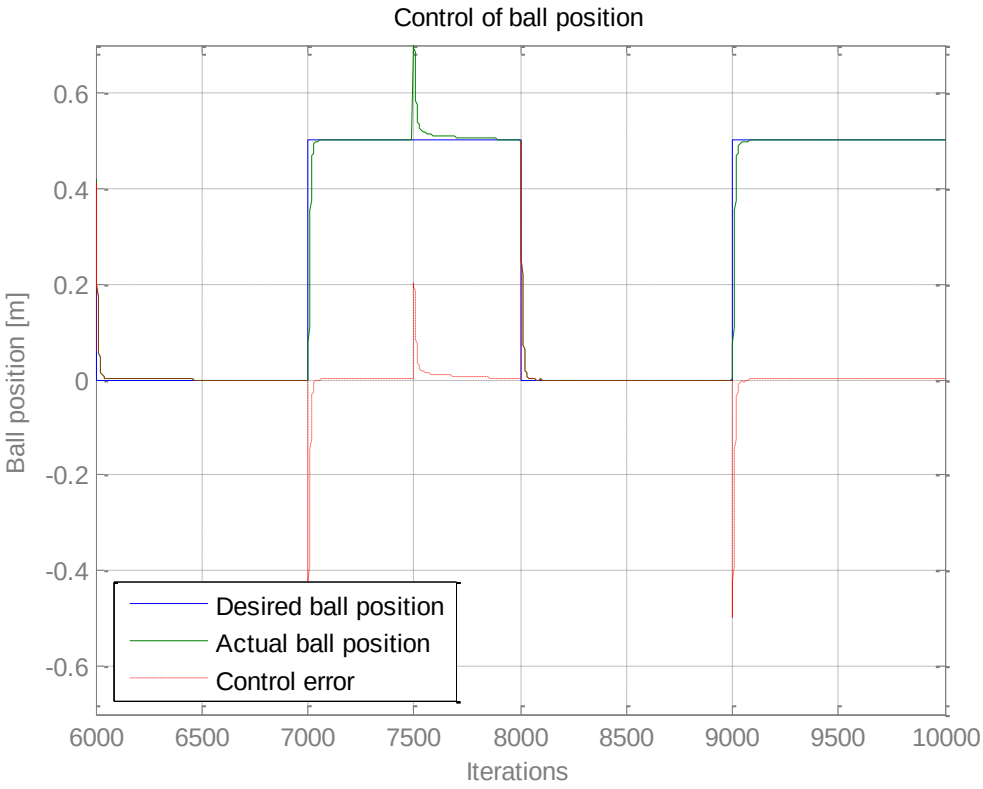


Figure 7. 14 Step disturbance (with disturbance canceller block turned on)

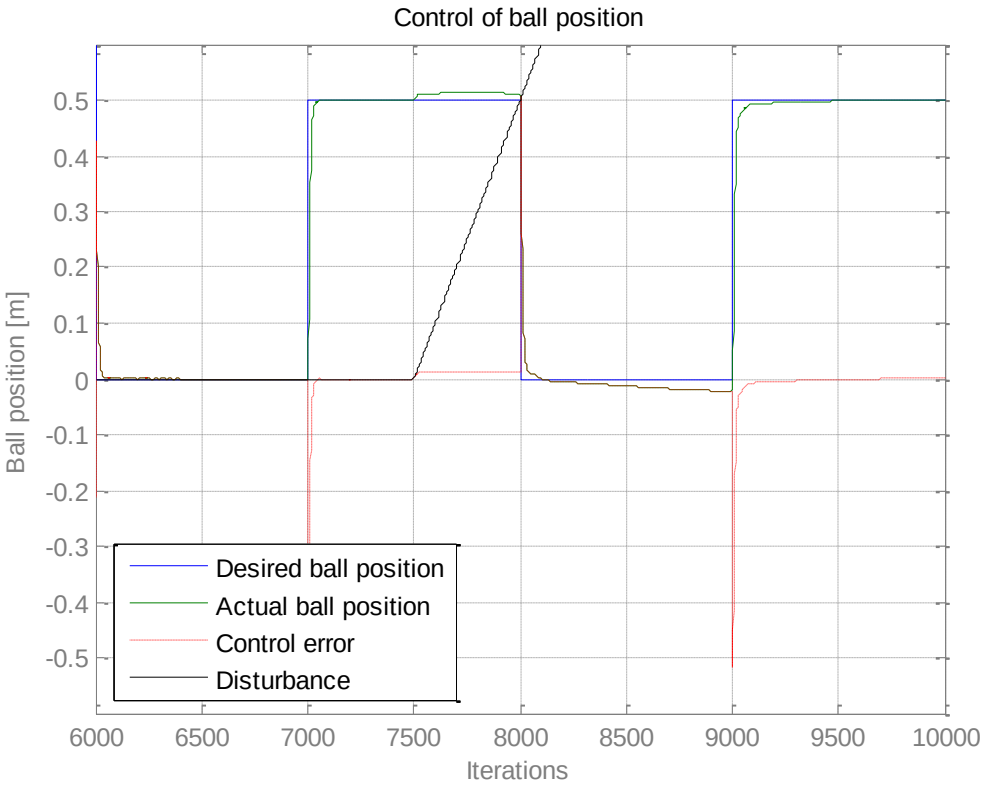


Figure 7. 15 Ramp disturbance (with disturbance canceller block turned on)

When random disturbance is added to the system, performance of the AIC decreases but dynamic control is still available. In figure 7.16 shows the system's performance against random disturbances. The disturbed output of the system is directly proportional with amplitude and the frequency of the noise added as disturbance.

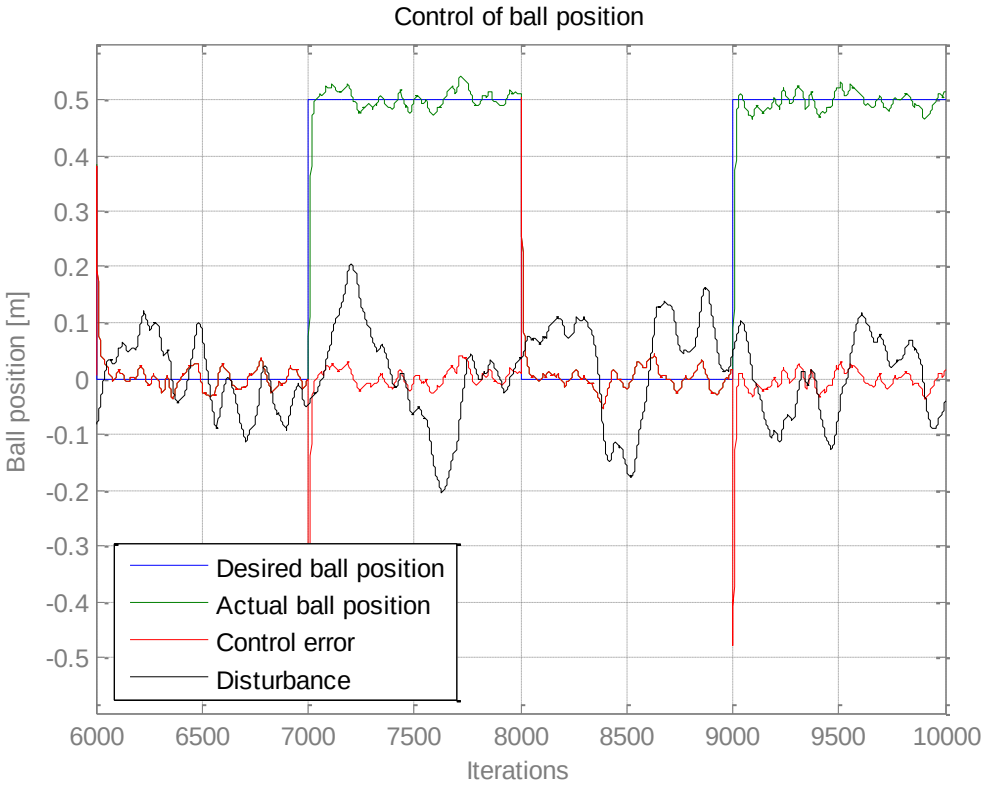


Figure 7. 16 Plant subjected to random disturbance

As seen from these results AIC system is capable of dynamic control even with disturbed plants. In the case of a system which is unknown or slowly time-varying, using an adaptive inverse controller will be acceptable even in existence of disturbances. The simulations made here are within limited iteration numbers. Increasing the iteration number will make more correct direct and inverse models thus the performance of the AIC system will increase over time.

It is always possible to get a better performance from the AIC system with changing its parameters. A more precise capturing of inputs and outputs is possible with decreasing the sampling time of the system. This will cause to work with much larger weight vector length and smaller convergence factors which will slow down the convergence time. Faster sampling rates also adds extra calculation load to the simulation hardware.

8. CONCLUSION

In this study adaptive inverse control concept is examined and applied on sample systems. Theoretical backgrounds are reviewed in first 5 chapters and most of the schemes are tested with simulations. Before applying adaptive filters on control schemes parameters of the adaptive filters are investigated in chapter 6. Criteria obtained for choosing system parameters are used to setup the AIC systems. Performance of the AIC system on time-varying plants and disturbed plants are investigated on chapter 7.

Parameter selection for AIC systems is the one of the most significant part of setting up the system. Wrong parameters make a perfect control scheme useless. There are two considerations for choosing the FIR filter length; settling time of the plant and the sampling rate of the overall control system. Product of filter length and sampling time must involve the systems settling time. This provides the FIR filter to behave more likely as the system. If the filter length is selected too long, time constant of the adaptive system total system error increases.

Convergence factor μ controls the stability and speed of the adaptation. As μ increases system converges faster but result in noisy output. Decreasing μ makes output smoother but system converges slowly. Converge factor is highly sensitive to the power of the input signal. Even a unit increment of input signal power may cause the system to lose its stability. As the power of the input signal increases it is needed to use smaller μ values. Using normalized LMS remedies this problem. In this algorithm μ is scaled by the power of the input signal. Another advantage of normalized LMS is faster convergence of the models. Compared to the conventional LMS it is 2 or 3 times faster.

As the adaptive algorithm uses the plant's output to adapt filter coefficients, all of the plant information must be acquired by exciting all the modes of the plant. This is possible with signals which are rich in frequency. When the signals of poor frequency content are used as modeling signals, the filter fails to converge to the

optimum solution. If the input signals are stationary in practice, then adding dither signals to the actual input helps to excite the system. Frequency content about 10 times of the bandwidth of the plant is sufficient in most cases.

Simulations have shown that small variations of the plant dynamics can be handled by AIC system. If the rate of change is high then AIC system fails to follow the command input. Within the limited range of changes AIC system is successful like the case that system is time invariant.

Adaptive inverse control scheme with disturbance canceling feedback shows good performance on canceling step and ramp type disturbances. Random disturbances can not be eliminated totally but dynamic control is still available with oscillations which reflect the structure of the noise.

REFERENCES

[1] **Widrow, B. and Walach, E.,** 1997. Adaptive Inverse Control, Prentice Hall, Inc. New Jersey.

[2] **Plett, G. L.,** 1998. Adaptive Inverse Control of Plants with Disturbance, *PhD Thesis*, Stanford, California.

[3] **Hızal, N. A.,** 1999. Improved Adaptive Model Control, *ARI*, **51**, 181-190.

[4] **Haykin, S.,** 1996. Adaptive Filter Theory, Prentice Hall, Inc. New Jersey.

[5] **Stearns, S. D. and David, R. A.,** 1996. Signal Processing Algorithms in Matlab. Prentice Hall, New Jersey.

[6] **Kaufman, H., Barkana, I. and Sobel, K.,** 1998. Direct Adaptive Control Algorithms Theory and Applications, Springer-Verlag, Inc. New York.

[7] **Widrow, B.,** 1971. Adaptive Model Control Applied to Real Time Blood-Pressure Regulation, *J. of Pattern Recognition and Machine Learning*, Plenum Pres,.310-324

[8] **Adam, T., Dadvandipour, S. and Futas, J.,** 2003. Influence of Discretization Method on the Digital Control System Performance, *Acta Montanistica Slovaca*, **8,** 197-200

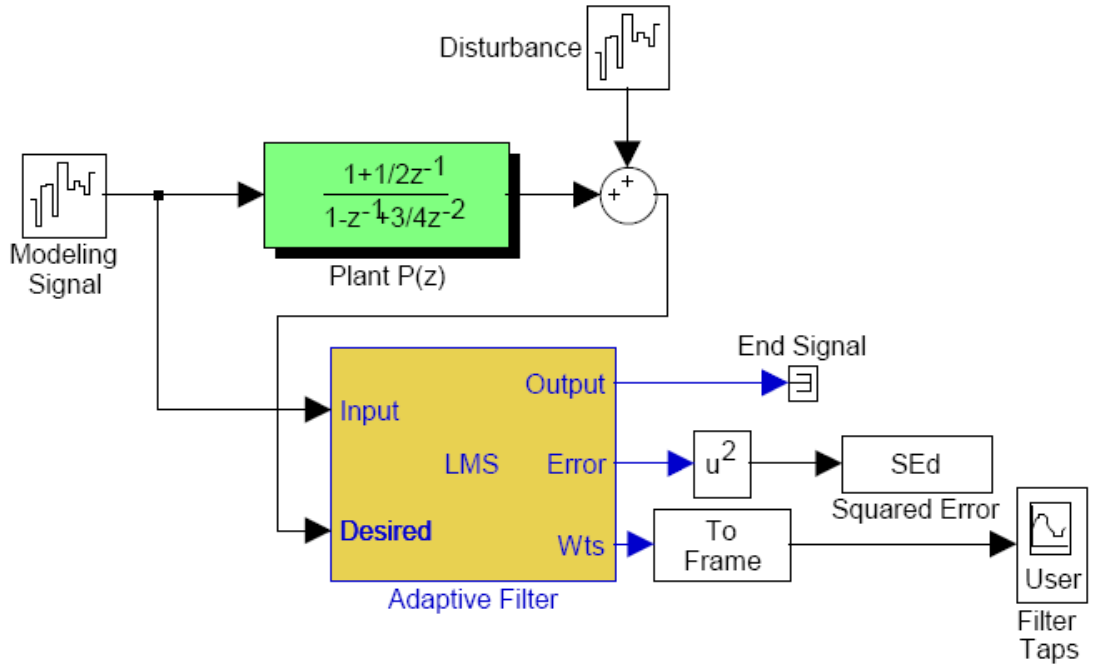
[9] **Farhang-Boroujeny, B.,** 1998. Adaptive Filters Theory and Applications, John Wiley & Sons, Ltd. Chichester.

[10] **Lyons, R. G.,** 2001. Understanding Digital Signal Processing, Prentice Hall, Inc. New Jersey.

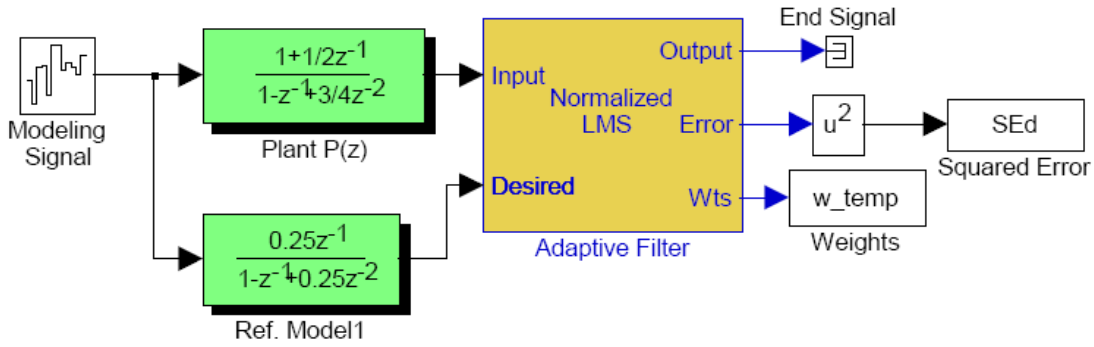
[11] **Glentis, G. O., Berberidis, K. and Theodoridis, S.,** 1999. Efficient Least Squares Algorithms for FIR Transversal Filtering, *IEEE Signal Processing Magazine*, **16,** 13-41.

APPENDIX – A

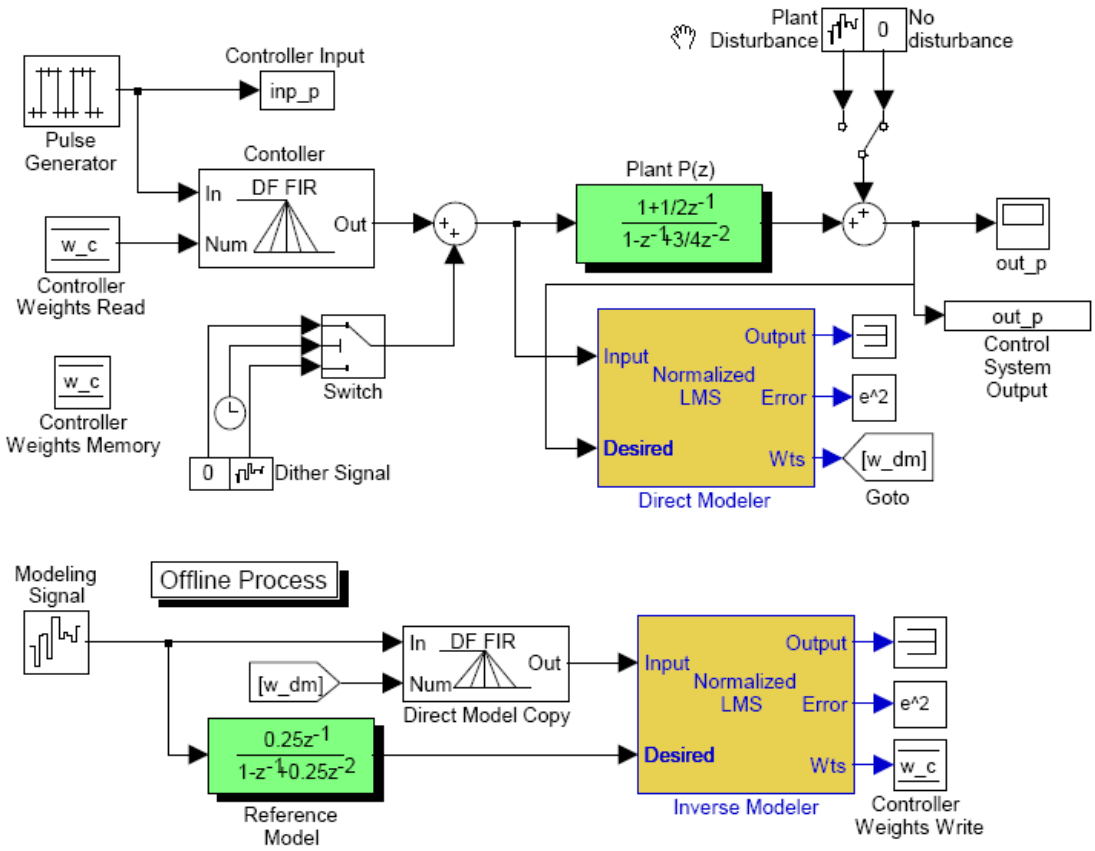
A.1 Simulink Diagram for Adaptive Direct Modeling



A.2 Simulink Diagram for Adaptive Inverse Modeling



A.3 Simulink Diagram for Adaptive Inverse Control Based on Figure 5.3



APPENDIX – B

B.1 Matlab Program for Adaptive Direct Modeling

```
clc;clear;
%Plant definition in digital filter form
Ts = .1; %Sampling time
%z=tf('z',Ts);
%Pz = (1 - 2 * z^-1)/(1 - z^-1 + .75 * z^-2);
%clear z; %Delete unnecessary variables from memory
numdf = [1 .5]; %     digital filter constants in ascending
dendf = [1 -1 .75]; % powers of z^-1
Pz=filt(numdf,dendf,Ts);
N = 2000; %number of iterations
wn_dm = 50; %direct model weight number
mu = 0.1; % convergence factor
inp_dm = zeros(1,wn_dm); %Initialize direct model input
out_dm = zeros(1,N); %Initialize direct model output
w_dm = zeros(1,wn_dm); %Initialize weights
sigma = 0; % Very small number for preventing
         % division by zero in normalized LMS
nLMS = 1; % Switch for normalized LMS
I = 0.5 * filtrand(N,.1,100); % modeling signal
disp 'Simulating Plant...';
[out_p t] = lsim(Pz,I); % Plant simulation
disp 'Adaptation Started...';
for k=1:N
   inp_dm = [I(k) inp_dm(1:wn_dm-1)]; %Direct model input
   out_dm(k) = inp_dm * w_dm'; %Direct model output
   e_dm(k) = out_p(k) - out_dm(k); %Error
   if nLMS == 1
      %Using Normalized LMS
      w_dm = w_dm + ((inp_dm / (sigma + (inp_dm * inp_dm'))) * ...
         mu * e_dm(k)); %Weight update
   elseif nLMS == 0
      w_dm = w_dm + 2 * mu * inp_dm * e_dm(k); %Weight update
   end
end
disp 'Adaptation Finished'
Pdm = filt(w_dm,[1],Ts); %Rebuild direct model in digital filter form
SE = e_dm.^2; %Square error
MSE = (abs(e_dm) / length(e_dm)).^2;%Mean square error
step(Pz,Pdm);
```

B.2 Matlab Program for Adaptive Inverse Modeling

```
clc;clear;
%Plant definition in digital filter form
Ts = .1; %Sampling time
%z=tf('z',Ts);
%Pz = (1 - 2 * z^-1)/(1 - z^-1 + .75 * z^-2);
%clear z; %Delete unnecessary variables from memory
numdf = [1 .5]; %digital filter constants
dendf = [1 -1 .75];%in ascending powers of z^-1
Pz=filt(numdf,dendf,Ts);
N = 50000; %number of iterations
wn_inv = 25; %inverse model weight number
mu = 0.001; % convergence factor
inp_inv = zeros(1,wn_inv); %Initialize inverse model input
out_inv = zeros(1,N); %Initialize inverse model output
w_inv = zeros(1,wn_inv); %Initialize weights
I = 0.5 * filtrand(N,.1,100); % modeling signal
[out_p t] = lsim(Pz,I); % Plant simulation
disp 'Adaptation Started...';
for k=1:N
   inp_inv = [out_p(k) inp_inv(1:wn_inv-1)]; %Inverse model input
   out_inv(k) = inp_inv * w_inv'; %Inverse model output
   e_inv(k) = I(k) - out_inv(k); %Error
   w_inv = w_inv + 2 * mu * inp_inv * e_inv(k); %Weight update
end
clc;disp 'Adaptation Finished'
Pinv = filt(w_inv,[1],Ts); %Build inverse model tf
step(Pinv*Pz);%Check whether deconvoluting successfully
```

B.3 Matlab Program for Adaptive Inverse Control of Time-varying Plant

```
%%Adaptive inverse control of a time-varying plant
%Based on Figure 5.2
clc;clear;
tv = 1; % 0:time invariant, 1:time varying(linear), 2:sinusoidal
c_inp = 1;%Command input 0:step, 1:square-wave ,2;sinus wave
nLMS = 1; %Normalized LMS Switch [1:On 0:Off]
if nLMS == 1;sigma = 1e-6;end%very small number, prevents division by zero
N = 10000;%number of iterations
wn_inv = 50; %inverse model weight number
mu_inv = .1; % inverse model convergence factor
Ts = 0.1;
t = 0:Ts:((N-1)*Ts); %time vector
delay = 1;
switch c_inp %Generate controller input
   case 0 %step
      I1 = 0.1 * filtrand(7500,.1,50);
      I2 = 0.5 * filtrand(2500,.1,.5,[100 500 2]);
      I = [I1 I2];
```

```
    case 1 %square-wave
        I = 0.5 * filtrand(N,.1,50,[N/2 0 0]);
        [u tu] = gensig('square',500,N-1,1);
        I = I + 0.5 * u';
    case 2 %sinus-wave
        I = 0.5 * filtrand(N,.1,50,[N/2 0 0]);
        [u tu] = gensig('sin',500,N-1,1);
        I = I + 0.5 * u';
end
%Plant
switch tv
    case 0 %time-invariant
        a = 30;
    case 1
        a = 30 + t * 0.003;%time-varying(linearly)
    case 2
        a = 30 + .5 * sin(2 * t);%time-varying(sinusoidal)
    otherwise
        disp('Error')
end
numd =[458 1374 1374 458] / 25809;
dend = [420*30+13209 -380*30-19133 19707-420*30 380*30-11951] / 25809;
order_p_1 = length(dend);
order_p = order_p_1 - 1;
sim_p = zeros(1,order_p_1);%Initialize plant simulation vector
out_p = zeros(1,N); %initialize plant output

%Reference Model
numr = 1; denr = [1/3 1];
[numdr dendr] = c2dm(numr,denr,Ts,'tustin');
order_r_1 = length(dendr);
order_r = order_r_1 - 1;
sim_r = zeros(1,order_r_1);%Initialize reference model simulation vector
out_r = zeros(1,N); %initialize reference model output
sim_r2 = zeros(1,order_r_1);
out_r2 = zeros(1,N);

%---
inp_inv = zeros(1,wn_inv); %Initialize Inverse Model Input
out_inv = zeros(1,N); %Initialize Inverse Model Output
w_inv = 0.1 * ones(1,wn_inv); %Weight vector
%---
inp_c = 0.1 * ones(1,wn_inv); %initialize controler input
%h = waitbar(0,'Please wait...');
for k = 1:N
    %waitbar(k/N)
    if tv == 1 | tv == 2%vary plant
        dend = [420*a(k)+13209 -380*a(k)-19133 19707-420*a(k) ...
            380*a(k)-11951] / 25809;
    end
```

```
    %Controler
    inp_c = [I(k) inp_c(1:wn_inv - 1)];
    out_c = inp_c * w_inv';
    %Plant
    inp_p(k) = out_c;
    sim_p(1) = inp_p(k) - dend(2:order_p_1) * sim_p(2:order_p_1)';
    out_p(k) = sim_p * numd';
    sim_p = [0 sim_p(1:order_p)];
    %Reference Model (to form ref. model inverse)
    inp_r(k) = out_c; %same input with plant
    sim_r(1) = inp_r(k) - dendr(2:order_r_1) * sim_r(2:order_r_1)';
    out_r(k) = sim_r * numdr';
    sim_r = [0 sim_r(1:order_r)];
    %Reference Model (to examine model following performance)
    inp_r2(k) = I(k); %same input with controller
    sim_r2(1) = inp_r2(k) - dendr(2:order_r_1) * sim_r2(2:order_r_1)';
    out_r2(k) = sim_r2 * numdr';
    sim_r2 = [0 sim_r2(1:order_r)];

    %Inverse Model
    inp_inv = [out_p(k) inp_inv(1:wn_inv - 1)];
    out_inv(k) = inp_inv * w_inv';
    %Inverse Model Weight Update
    if k > delay
       e_inv(k) = out_r(k - delay) - out_inv(k); %error
       if nLMS == 1
          w_inv = w_inv + ((inp_inv / (sigma + (inp_inv * ...
             inp_inv'))) * mu_inv * e_inv(k)); %nLMS
       elseif nLMS == 0
          w_inv = w_inv + 2 * mu_inv * inp_inv * e_inv(k); %LMS
       end
    end
end
SEi = e_inv.^2; %Squared error
%close(h);
clear order_p order_p_1 order_r order_r_1 out_c numr num denr den ...
    tv wn_inv sim_r sim_p nLMS mu_inv h dend numd dendr numdr k ...
    delay sigma
disp('finished')
e_overall = (out_r2 - out_p);
plot([9000:10000],out_p(9000:10000),[9000:10000],I(9000:10000));
%axis([500 N*Ts -1 1]);grid
%hold on;plot([1 N*Ts],[0.98 0.98],'r');plot([1 N*Ts],[1.02 1.02],'r')
figure;plot([9000:10000],e_overall(9000:10000))
```

B.4 Matlab Program for Adaptive Inverse Control of Plants with Disturbances

```
%%AIC with disturbance canceling
%Based on figure 5.4
clc;clear;
dist_type = 2; %0:no dist., 1:step, 2:random, 3:ramp
start_dc = 4000;%iteration to start disturbance canceling
add_dist = 4100;%iteration to add disturbance
%Adjustable Parameters
N = 10000;%iteration number
wn_dm = 50;%direct model weight number
wn_inv = 50;%inverse model weight number
mu_dm = 0.01;%direct model convergence factor
mu_inv = 0.01;%inverse model convergence factor
delay = 0;%inverse modeling delay
%SIGNALS
I = 0.5 * filtrand(N,.1,5,[4000 0 1]);%excitation signal
[u t] = gensig('square',2000,N-1,1);%command input
I = I + 0.5 * u';
I_off = 0.5 * filtrand(N,.1,50);%offline process modeling signal
noise = filtrand(N,.1,.2);%random noise
%PLANT --ball and beam experiment
Ts = 0.1;%sampling time
num=[2.1 4.2];
den=[1 2.1 4.2];
[numd dend]=c2dm(num,den,Ts,'tustin');
order_p = length(dend) - 1;
%REF. MODEL (settling time < 3 s, no overshoot)
num_r=[0 0 4];
den_r=[1 4 4];
[numd_r dend_r]=c2dm(num_r,den_r,Ts,'tustin');
order_r = length(dend_r) - 1;
%initialize variables
out_p = zeros(1,N);%disturbed plant output
sim_p = zeros(1,N);%plant simulation output
out_dm = zeros(1,N);%direct model output
out_dmc_off = zeros(1,N);%direct model copy output(offline process)
out_dmc_on = zeros(1,N);%direct model copy output(online process)
out_inv = zeros(1,N);%inverse model output
out_c = zeros(1,N);%controller output
out_dc = 0;%disturbance canceller output
out_r = zeros(1,N);%reference model output
inp_dm = zeros(1,wn_dm+1);%direct model input
inp_dmc_off = zeros(1,wn_dm+1);%direct model copy input(offline)
inp_dmc_on = zeros(1,wn_dm+1);%direct model copy input(offline)
inp_inv = zeros(1,wn_inv+1);%inverse model input
inp_c = zeros(1,wn_inv+1);%controller input
inp_dc = zeros(1,wn_inv+1);%disturbance canceller input
sim_p_dummy = zeros(1,order_p+1);%plant simulation dummy variable
sim_r_dummy = zeros(1,order_r+1);%ref. model simulation dummy variable
```

```
w_dm = (dimpulse(numd,dend,wn_dm+1))';%direct model weights
w_inv = .03 * ones(1,wn_inv+1);%inverse model weights
dist = zeros(1,N);%disturbance
dly = zeros(1,delay+1);%inverse modeling delay vector
sigma = 1e-6;%very small number, prevents division by zero
%START ITERATION
disp('Please wait...')
for k=1:N
   if k >= 5000
      mu_dm = 0.001;
      mu_inv = 0.001;
   end
   %CONTROLLER
   inp_c=[I(k) inp_c(1:wn_inv)];
   out_c(k)=inp_c*w_inv';
   %PLANT
   inp_p(k)=out_c(k)-out_dc;
   sim_p_dummy(1)=inp_p(k)-dend(2:order_p+1)*sim_p_dummy(2:order_p+1)';
   sim_p(k)=sim_p_dummy*numd';
   sim_p_dummy=[0 sim_p_dummy(1:order_p)];
   %Add Disturbance
   if k >= add_dist
      if dist_type == 0%no dist.
         dist(k) = 0;
      elseif dist_type == 1%step
         dist(k) = 0.2;
      elseif dist_type == 2
         dist(k) = noise(k);%random
      elseif dist_type == 3
         dist(k) = (t(k) * 0.001)-(add_dist*0.001);%ramp
      end
   end
   out_p(k) = sim_p(k) + dist(k); %disturbed plant output
   %DIRECT MODEL
   inp_dm_1=out_c(k);
   inp_dm=[inp_dm_1 inp_dm(1:wn_dm)];
   out_dm(k)=inp_dm*w_dm';
   e_dm(k)=out_p(k)-out_dm(k);
   w_dm=w_dm+((inp_dm/(sigma + (inp_dm*inp_dm'))) * mu_dm * ...
      e_dm(k)); %nLMS
   %DIRECT MODEL COPY
   inp_dmc_on_1=out_c(k)-out_dc;
   inp_dmc_on=[inp_dmc_on_1 inp_dmc_on(1:wn_dm)];
   out_dmc_on(k)=inp_dmc_on*w_dm';
   %DIST CAN.
   if k>=start_dc
      inp_dc_1=out_p(k)-out_dmc_on(k);
      inp_dc=[inp_dc_1 inp_dc(1:wn_inv)];
      out_dc=inp_dc*w_inv';
   end
```

```
    %%OFFLINE PROCESS
    %DIRECT MODEL
    inp_dmc_off_1=I_off(k);
    inp_dmc_off=[inp_dmc_off_1 inp_dmc_off(1:wn_dm)];
    out_dmc_off(k)=inp_dmc_off*w_dm';
    %REF. MODEL
    inp_r(k)=I_off(k);
    sim_r_dummy(1)=inp_r(k)-dend_r(2:order_r+1)*sim_r_dummy(2:order_r+1)';
    out_r(k)=sim_r_dummy*numd_r';
    sim_r_dummy=[0 sim_r_dummy(1:order_r)];
    %Inverse modeling DELAY
        dly(1) = out_r(k); %set current as 1st delay vector element
        out_r_delayed(k) = dly(delay + 1);%get previous for current iteration
        dly = [0 dly(1:delay)];%shift delay vector
    %INVERSE MODEL
    inp_inv = [out_dmc_off(k) inp_inv(1:wn_inv)];
    out_inv(k) = inp_inv * w_inv';
    e_inv(k) = out_r_delayed(k) - out_inv(k);
    w_inv=w_inv+((inp_inv/(sigma + (inp_inv*inp_inv'))) * mu_inv * ...
      e_inv(k)); %nLMS
    error(k) = I(k) - out_p(k);
end
%END ITERATION
clear delay den den_r dend dend_r dist_type dly inp_c inp_dc inp_dc_1 ...
      inp_dm inp_dm_1 inp_dmc_off inp_dmc_off_1 inp_dmc_on ...
      inp_dmc_on_1 inp_inv inp_p inp_r k mu_dm mu_inv nLMS num ...
      num_r numd numd_r order_p order_r out_c out_dc out_dm ...
      out_dmc_off out_dmc_on out_inv out_r sigma out_r_delayed ...
      sim_p_dummy sim_r_dummy wn_dm wn_inv start_dc w_dm w_inv
disp('Simulation Finished')
figure;
plot(1:N,I(1:N),1:N,out_p(1:N),1:N,-error(1:N));axis([6000 10000 -.6 .6]);
title('Control of ball position');
legend('Desired ball position','Actual ball position','Control error')
xlabel('Iterations');ylabel('Ball position [m]');
grid on;
```

B.5 Matlab Code of Filtrand Sub-program

```
function y=filtrand(m,Ts,Wn,st)
% y=filtrand(m,Ts,Wn,st)
% Filtered random signal.
% "randn" is passed through lpfilt with
%   natural (corner) frequency Wn.
% The length of the row vector y is specified as m.
% If st is used as a nonzero value, a step function is placed
%   at the end, the total length still being m:
%   st=[length of zeros, length of ones, step size]
%   Step size is optional (Default: 1).
%   st can be a scalar, in which case st is taken as [st st 1].
```

```
%   If st is a four vector, the first element specifies multiple
%   steps with period given by it. E.g.: [1000 50 50 1] places
%   steps at 1000 before, at 2000 before etc. from the end, in
%   addition to the one at the end. For equal intervals and also
%   to start away from a step, make m a multiple of st(1) of the
%   four-vector st.
% To increase the frequency content, increase Wn with
%   a fixed Ts. "randn" can have frequencies upto the
%   Nyquist frequency  pi/Ts, so, beyond Wn = about 2*pi/Ts,
%   randn remains practically unfiltered. It is the product
%   Wn*Ts that determines the apparent roughness or smoothness:
%   Wn*Ts = 2*pi*(Ts/Tn) = pi*(wn/w_nyquist).
%   See lpfilt.
% N.A.Hızal.
stpr=0;
if nargin==4
  if length(st)==1
     st=[st st 1];
  elseif length(st)==2
     st=[st 1];
  elseif length(st)==4
     stpr=st(1); st(1)=[];
  end
  yst=[zeros(1,st(1)) st(3)*ones(1,st(2))];
else
  st=[0 0]; yst=[];
end
x=randn(1,m-st(1)-st(2));
y=lpfilt(x,Ts,Wn);
y=[y yst];
if stpr
 k=1;
 l=m-st(1)-st(2);
 while 1
   if l-k*stpr<0  break, end
   y(l-k*stpr+1:l-k*stpr+st(1))=zeros(1,st(1));
   y(l-k*stpr+st(1)+1:l-k*stpr+st(1)+st(2))=st(3)*ones(1,st(2));
   k=k+1;
 end
end
```

B.6 Matlab Code of Lpfilt Sub-program

```
function y=lpfilt(u,Ts,Wn)
% Low-pass filter for input vector u
% Second order filter (zeta = .7071 and user-defined Wn)
% N.A.Hizal.
zeta=.7071;numa=1;dena=[1/Wn/Wn 2*zeta/Wn 1];
[num den]=c2dm(numa,dena,Ts,'tustin');
y=filter(num,den,u);
```

CURRICULUM VITAE

Mustafa UYSAL was born in Kdz. Ereğli in 1981. He graduated from Kdz. Ereğli Anatolian High School in 1999 and took Bachelor's degree from Kocaeli University in Mechanical Engineering. He has been graduate student in İstanbul Technical University since 2003.