İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

CONSTRAINED ADAPTIVE INVERSE CONTROL WITH DISTURBANCES

M.Sc. Thesis by

Deniz ER, B.Sc.

Department : Mechanical Engineering

Programme: System Dynamics and Control

Supervisor : Prof. Dr. N. Aydın Hızal

JUNE 2007

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

CONSTRAINED ADAPTIVE INVERSE CONTROL WITH DISTURBANCES

M.Sc. Thesis by

Deniz ER, B.Sc.

503041602

Date of submission : **7 May 2007**

Date of defence examination: **4 June 2007**

Supervisor (Chairman): **Prof. Dr. N. Aydın Hızal**

Members of the Examining Committee: **Prof.Dr. Serhat ŞEKER**

Assoc. Prof. Dr. Şeniz Ertuğrul

JUNE 2007

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GÜRÜLTLÜ ALTINDA SINIRLI UYARLAMALI TERS KONTROL

Yüksek Lisans Tezi

Deniz ER,
503041602

Anabilim Dalı: Makina Mühendisliği

Programı: Sistem Dinamiği ve Kontrol

Tez Danışmanı: Prof. Dr. N. Aydın HIZAL

HAZİRAN 2007

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Prof. Dr. N. Aydın Hızal. He has provided technical support and encouragement. I also would like to express my gratitude Assoc. Prof. Şeniz Ertuğrul, who introduces me with Neural Networks. I also would like to express gratitude the bosses of my company SONAR AR-GE, Mr. Z. Ali Eser, Mr. Serhat Saka, Mr. Taşkın Baylan, and my colleague Mr. Emrah Eryılmaz. I would also express my indebts my parents, that they always encourage me, during my life. My engineering began at Yeditepe University. The friends I knew from Yeditepe Alper Tozan and Eren Emir, thank you for your supports.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ..... **III**

TABLE OF CONTENTS ..... **IV**

ABBREVIATION ..... **VII**

LIST OF FIGURES ..... **VIII**

LIST OF SYMBOLS ..... **X**

ÖZET ..... **XIII**

ABSTRACT ..... **XIV**

1 INTRODUCTION **1**

2 ADAPTIVE INVERSE CONTROL **2**

3 LINEAR ADAPTIVE FILTERS **5**

3.1 Adapting Linear Filters **6**

3.2 Optimal Solution for Linear Adaptive Filters **9**

4 NON- LINEAR ADAPTIVE FILTERS **11**

4.1 Adapting Nonlinear Filters **13**

4.1.1 Adapting a Feedforward Neural Network 13

4.1.2 Adapting an Externally-Recurrent Neural Network 16

4.2 Optimal Solution for Non Linear Adaptive Filters **18**

5 SYSTEM IDENTIFICATION **21**

5.1 Identification of Linear Systems 21

5.2 Identification of Nonlinear Systems 23

5.3 Simulation Examples 24

5.3.1 Linear Plant 24

5.3.2 Nonlinear Plant 35

6 ADAPTIVE FEEDFORWARD CONTROL 42

6.1 Introduction 42

6.2 Constrained Controller via BPTM Algorithm 43

6.2.1 Linear FIR Plant Model, Linear FIR Controller 46

6.2.2 Nonlinear NARX Plant Model, Nonlinear NARX Controller 49

6.3 Simulation Examples 50

6.3.1 Linear Plant 51

6.3.2 Nonlinear Plant 53

6.3.3 Summary 54

7 DISTURBANCE CANCELLING 56

7.1 Introduction 56

7.1.1 Structure of the Disturbance Canceller 56

7.2 Synthesis of the Disturbance Canceller via the BPTM Algorithm 59

7.2.1 Training X In-Place 59

7.3 Simulation Examples 60

7.3.1 Linear Plant 60

7.3.2 Nonlinear Plant 62

7.3.3 Summary 63

8 SUMMARY 64

8.1 Constrained Adaptive Feedforward Control 64

8.2 Disturbance Cancelling 65

BIBLIOGRAPHY 66

APPENDIX A **71**

Performance Surface **71**

The Gradient and Minimum MSE **73**

The Method of Steepest Descent **74**

The Least Mean Squares (LMS) Algorithm **74**

APPENDIX B **76**

CURRICULUM VITAE **106**

ABBREVIATION

ARMA :AutoRegressive Moving Average.

FIR : Finite impulse response.

i.i.d. : Independent and identically distributed (random variables).

IIR : Infinite impulse response.

MSE : Mean Squared Error.

NARX : Nonlinear AutoRegressive filter with eXogeneous input.

Backprop :Adapts feedforward neural networks (also known as "Backpropagation").

BPTT : BackPropagation Through Time: Adapts recurrent neural networks.

BPTM : BackPropagation Through (Plant) Model:

LMS : Least Mean Square: Adapts FIR linear filters.

RTRL : Real Time Recurrent Learning: Adapts recurrent neural networks in real time.

LIST OF FIGURES

Figure 2.1: Basic Concept of Adaptive Inverse Control 2
Figure 2.2: Model Reference Adaptive Inverse Control 3
Figure 2.3 : Noise and Disturbance Cancelling 3
Figure 3.1 : Symbolic Representation of Basic Adaptive Filter 5
Figure 3.2 : IIR Adaptive Filter 6
Figure 3.3: Modelling an unknown system by a discrete adaptive filter 6
Figure 4.1 : Modelling an Unknown System with Neural Network 11
Figure 4.2 : Neuron 12
Figure 4.3 : Feedforward Neural Network 13
Figure 4.4: Externally Recurrent Neural Network 17
Figure 5.1 : Adaptive System Identification 22
Figure 5.2 : Parallel and Series-Parallel Connection Identification 23
Figure 5.3 : Tank Temperature Control 25
Figure 5.4 : Bode Diagram of Linear Plant 27
Figure 5.5 : Pole Zero Map of Linear Plant 28
Figure 5.6 : Impulse Response of Linear Plant 28
Figure 5.7: 20 Tab FIR Filter 28
Figure 5.8: Linear Plant Identification with FIR Filter, μ=0.1 29
Figure 5.9: Linear Plant Identification with FIR Filter, μ=0.01 29
Figure 5.10 : Linear Plant Identification with FIR Filter, μ=0.8 30
Figure 5.11 : Feedforward Neural Network 30
Figure 5.12 : Linear Plant Identification with Feedforward NN, with μ=0.1 linear activation function and 200 time steps 31
Figure 5.13 : Feedforward Neural Network 31
Figure 5.14 : Linear Plant Identification with Feedforward NN, with μ=0.01 Using Linear Activation Function for 200 Time Steps 32
Figure 5.15: Feedforward Neural Network 32
Figure 5.16: Linear Plant Identification with Feedforward NN, with μ=0.1 Using Nonlinear Activation Function for 200 time steps 33

Figure 5.17: Externally Recurrent Neural Network 33
Figure 5.18 : Linear Plant Identification with Externally-Recurrent NN with Using Linear Activation Function for 200 Time Steps 34
Figure 5.19 : Linear Plant Identification with Externally-Recurrent NN with μ=0.1 Using Nonlinear Activation Function for 200 Time Steps 34
Figure 5.20 : Illustration of Heading Angle 36
Figure 5.21 : Block Diagram of Ship Yaw Dynamics from Wheel Angle δ_w to the Heading Angle ψ 37
Figure 5.22 : Step Response of Ship Yaw Dynamics with Stabilized and Non-Stabilized Plant 38
Figure 5.23: Stabilized Plant 38
Figure 5.24 : Disturbance Signal 39
Figure 5.25 : Training and Validation Results of Plant Modelling with Feedforward Neural Network 40
Figure 5.26 : Training and Validation Results of Plant Modelling with Feedforward Neural Network with more inputs 40
Figure 5.27 Training Results and Validation Results of Plant Modelling with Externally Recurrent Neural Network 41
Figure 6.1 : Adaptive Inverse Control 42
Figure 6.2 : Adaptive Inverse Control 43
Figure 6.3 : Structure Diagram Illustrating BPTM Method 44
Figure 6.4 : Constrained Control Effort 51
Figure 6.5 : Unconstrained Control Effort 52
Figure 6.6 : Modelling and System Error Values of Constrained Control 52
Figure 6.7 : Modelling and System Error Values of Unconstrained Control 52
Figure 6.8 Constrained Control of Plant with FF NN 53
Figure 6.9 Constrained Control of Plant with FF NN 54
Figure 6.10: Constrained Control of Plant under Disturbance Effect 54
Figure 7.1 : Input Output Timing of a Discrete-Time Control System 57
Figure 7.2 : A Useful way of Looking at Feedforward System Dynamics 57
Figure 7.3 : Internal Structure of X 58
Figure 7.4 : Internal Structure of X if the Plant is Linear 59
Figure 7.5 : Disturbace Cancelling via X-In Place 61
Figure 7.6 Disturbance Graph of Linear Plant 61

LIST OF SYMBOLS

C : The adaptive controller.

CCOPY : A filter whose weights are a digital copy of those in the adaptive controller C.

E : An adaptive estimator used to predict future disturbance values.

ECOPY : A filter whose weights are a digital copy of those in the adaptive estimator E.

F : A filter used to predict the current sensor noise given the past values of estimated disturbance.

M : The reference model.

P : The plant.

$\hat{P}$: The adaptive plant model.

$\hat{P}_{COPY}$: A filter whose weights are a digital copy adaptive plant model $\hat{P}$ weights

X : The adaptive disturbance canceller.

c_k : Impulse response of linear controller.

d_k : Desired response for system output.

$\vec{d}_k$: The infinite vector containing $\vec{d}_k$, $\vec{d}_{k-1}$ . . .

$\tilde{d}_k$: Desired response for sensor output.

$e_k^{(\mathrm{mod})}$: Plant modelling error

$e_k^{(sys)}$: System error.

$\tilde{e}_k^{(sys)}$: Measured system error.

$\tilde{\varepsilon}_k$: Modified system error.

P_k : Impulse response of linear plant model.

r_k : Reference input.

$\vec{r}_k$: The infinite vector containing $\vec{r}_k$, $\vec{r}_{k-1}$ . . .

u_k : Controller output.

$\vec{u}_k$: The Infinite vector containing $\vec{u}_k$, $\vec{u}_{k-1}$ . . .

$\tilde{u}_k$: Disturbance canceller output.

$\vec{\tilde{u}}_k$: The infinite vector containing $\tilde{u}_k$, $\tilde{u}_{k-1}$ . . .

w_k : Disturbance at plant output.

$\vec{w}_k$: The infinite vector containing $\vec{w}_k$, $\vec{w}_{k-1}$ . . .

$\hat{w}_k$: Estimate of disturbance at plant output.

$\vec{\hat{w}}_k$: The infinite vector containing $\vec{\hat{w}}_k$, $\vec{\hat{w}}_{k-1}$ . . .

x_k : Impulse response of linear disturbance canceller.

y_k : Plant output (including disturbance).

$\vec{y}_k$: The infinite vector containing $\vec{y}_k$, $\vec{y}_{k-1}$ . . .

$\hat{y}_k$: Plant model output.

$\mathrm{E}[.]$: Returns the expected value of $(.)$.

$[.]_+$: Takes the inverse z-transform of $(.)$, sets the non-causal part to zero, and returns the z-transform of the remaining causal part.

$(.)^+(z)$: The minimum-phase part of the spectral factorization of $(.)(z)$, such that $(.)(z) = (.)^+(z)(.)^-(z)$

.

$(.)^{-}(z)$: The non-minimum-phase part of the spectral factorization $(.)(z)$ of, such that $(.)(z) = (.)^{+}(z)(.)^{-}(z)$

$Z\{.\}$: Returns the z-transform. When $(.)$ is a Laplace transform, this operator takes the inverse Laplace transform of its operand, samples the result, and returns the z-transform of the sampled signal.

$\partial A / \partial B$: Returns the ordinary partial derivative of A with respect to B.

$\partial^{+} A / \partial B$: Returns the ordered partial derivative of A with respect to B. The ordered derivative is equal to the total derivative for ordered systems. An ordered system is a mathematical system of equations which is evaluated in a specific sequence.

$\mathrm{FIR}_{(a,0):b}$: An FIR filter with a tapped delays on each input, and b outputs.

$\mathrm{IIR}_{(a,b):c}$: An IIR filter with a tapped delays on each exogeneous input, b tapped delays on each feedback input, and c outputs.

$\mathrm{N}_{(a,b):c:d...}$: A neural-network based NARX filter with one stream of exogeneous input. There are a tapped delays on each input, b tapped delays on each output; c neurons in the first layer, d neurons in the second layer, and so forth.

$\mathrm{N}_{([a_1,a_2],b):c:d...}$: A neural-network based NARX filter with two streams of exogeneous input. There are a_1 tapped delays on each input from the first stream, a_2 tapped delays on each input from the second stream, b tapped delays on each output; c neurons in the first layer, d neurons in the second layer, and so forth.

ÖZET

Kontrol Teorisinin amacı dinamik sistemin en doğru ve sağlam olarak istenilen şekilde davranmasını sağlamaktır. Bu amaç, sistemin kararlı hale getirilmesi, kontrolü ve sistemdeki gürültünün yok edilmesi olarak üç ana gruba ayrılabilir. Konvansiyonel kontrol sistemleri, lineer olmayan veya sistemin dinamiklerinin zamanla değiştiği durumlarda yetersiz kalmaktadırlar.

Uyarlamalı ters kontrol metodolojisi bu tip sistemlerin kontrolünde kullanılabilir. Bu çalışmada lineer ve lineer olmayan sistemler kontrol edilmeye çalışılmıştır. Yapay Sinir Ağları ve Uyarlamalı FIR filtreler, Gradient-Descent tabanlı algoritmalarla eğitilmiş, sistemin modeli, kontrolörü ve gürültü yok edici olarak kullanılmıştır.

Algoritma sistemin modelinin çıkarılmasına, kontrolörünün ve gürültü yok edicinin elde edilmesinde ayrı izin vermektedir. Kullanıcının belirleği sınırlı kontrol de sağlanabilir.

Bu tezde iki sistem araştırılmıştır, birinci sistemde amaç tankın içindeki sıvının sıcaklığının kontrol edilmesidir. Tanka giren su miktarı ile çıkan su miktarı birbirine eşittir. Giren sıvının sıcaklığı bir vana ile sıcak ve soğuk kaynaklardan gelen sıvının karıştırılması ile elde edilir. Sistemimiz linear, minimum fazda ve durağandır. Buda tersinin oluşturulmasını sağlar. Birinci sistemdeki amaç, gürültü etkisi ve kullanıcının belirlediği kısıtlamalar altında sistemin kontrolünü sağlamaktır.

İkinci sistem lineer değildir. Kontrol sisteminin tanımı: Gemi pilotları geminin kafa açısını istedikleri yönde tutmak isterler. Sert dönüşlerde, zamanla değişen bir referans yönü oluşur, bu referansın izlenmesi istenir. Sistemdeki gürültü dalgaların dümene etkisi olarak alınmıştır.

Bu sistemdeki kısıtlamalar, sistemin dinamiğinin içindedir. İkinci sistemdeki amaç, gürültü etkisi ve sistemin iç kısıtlamaları altında sistemin kontrolünü sağlamaktır.

ABSTRACT

The aim of control theory is; to force the dynamical system to behave in user specified manner as accurately, and as robust as possible. The aims may be separated into three parts; stabilizing the plant, controlling the plant and disturbance cancelling. Conventional control systems are not adequate in such as non linear or time varying dynamic in controlled system.

Adaptive inverse control is a methodology, which achieves to control these kinds of systems. In this work both linear and nonlinear plants are tried to be controlled. Neural networks and FIR filters, which are trained by gradient-descent based algorithms, are used for modelling, controlling and disturbance cancelling.

The algorithm allows separate implementation of the adaptive controller, plant model and disturbance canceller. General user specified constraints on the control effort may be satisfied.

In this thesis , two plants are investigated, in first plant the goal is to control the temperature of a tank of liquid. The flow-rate of water into the tank is constant and equal to the flow rate of water out of the tank. The temperature of the incoming liquid is controlled by a mixing valve that adjusts the relative amounts of hot and cold supplies of the water. The plant is linear , quasistatically stationary and in minimum phase. A perfect stable and causal delayed inverse may be constructed. The aim in first plant is to control the plant under the effect of disturbances and user specified constraints.

The second example is selected as non linear. The control problem is; autopilots for ships are often designed to keep the ship's heading angle in a desired direction. There are some applications, such as course changing and turning, however, where it is desirable to be able to track a time-varying reference direction. The disturbances experienced in the dynamics of the ship are caused almost exclusively by the action of sea waves acting on the rudder angle. The constraints are inside the dynamics of

the plant. The aim in second plant is to control the plant under the effect of disturbances and constraints.

1 INTRODUCTION

The aim of control theory is to force the dynamical system to (the “plant”) behave in a user specified manner as accurately and robust as possible. There are many kinds of plants dynamics and several control strategies are introduced according to be either linear or nonlinear. Linear dynamical systems obey the superposition principle and nonlinear do not [19].

As one researcher states: “From a mathematical point of view, even the control of known nonlinear dynamical systems is a formidable problem. This becomes substantially more complex when the representation of the system is not completely known [22].”

In this work adaptive controller which has adjustable parameters are used, for controlling the nonlinear and linear plants. In thesis, control problem is investigated under adaptive inverse control.

2 ADAPTIVE INVERSE CONTROL

The basic idea of adaptive inverse control is illustrated in Figure 2.1. This diagram can be thought of as a filter which has adjustable parameters, an input and an output, and an extra input called error, which is used to adjust the parameters of the controller.

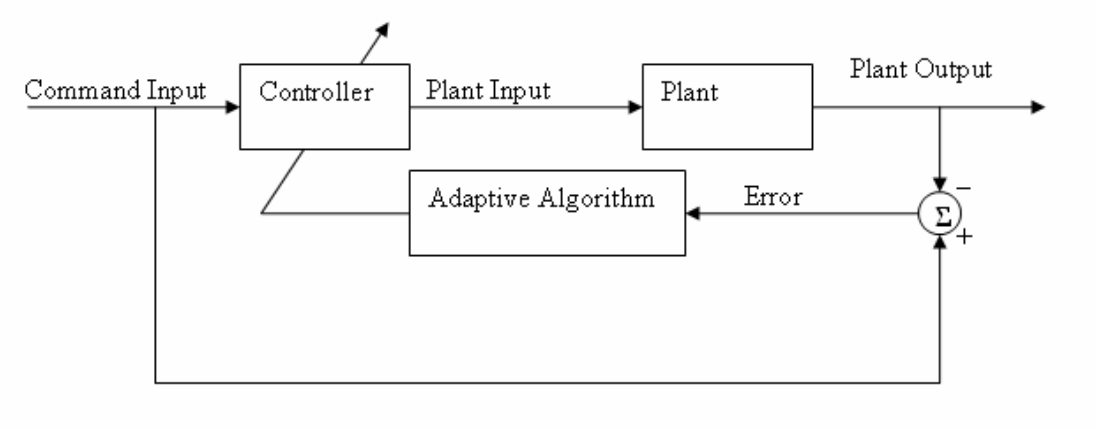

Figure 2.1: Basic Concept of Adaptive Inverse Control

Minimization of the mean square error is the main objective of the adaptation algorithm. The error is the difference between the plant output and the command input. If the error approaches to zero, the transfer function of the controller becomes the inverse of the plant. Combined transfer function of the plant and the controller becomes unity, so the plant output will track command input.

In above system, the controller is assumed to be convergent, linear and quasistatically stationary, the plant is linear, that it varies slowly so that it is quasistatically stationary.

Basic concept of adaptive inverse control is discussed above, can be developed according to complex cases. In some cases a smoothed or delayed version of command input model is generally designated as a reference model. This is why the system illustrated Figure 2.2 is called Model Reference Adaptive Inverse System.

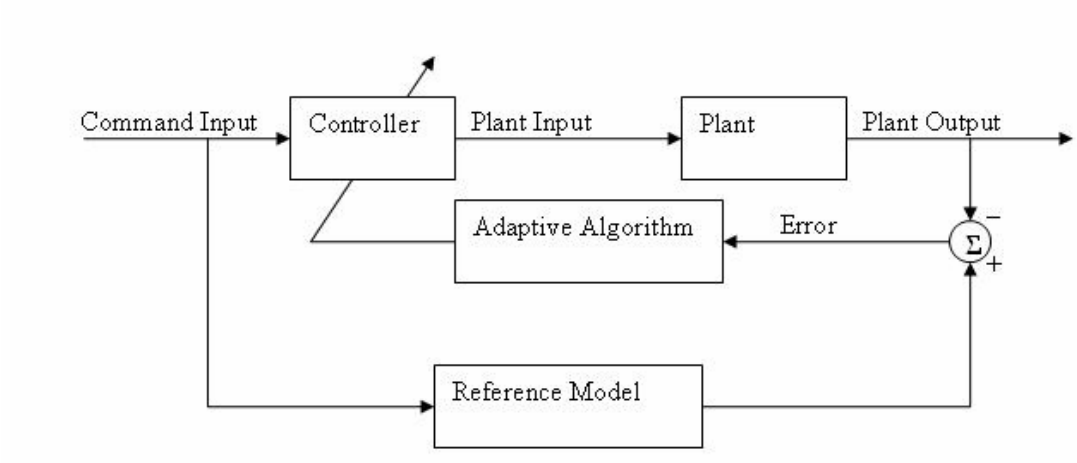

Figure 2.2: Model Reference Adaptive Inverse Control

In this system the combined transfer function of the plant and the controller is closely approximate of the transfer function of the reference model.

Plant noise and disturbance present a problem for adaptive inverse control approach. Lack of feedback from plant output permits internal plant noise and disturbance to exist unchecked at the plant output. Various signal processing methods for noise cancelling have been developed and with some modification they have been applied to the cancellation of plant noise and disturbance. [2].

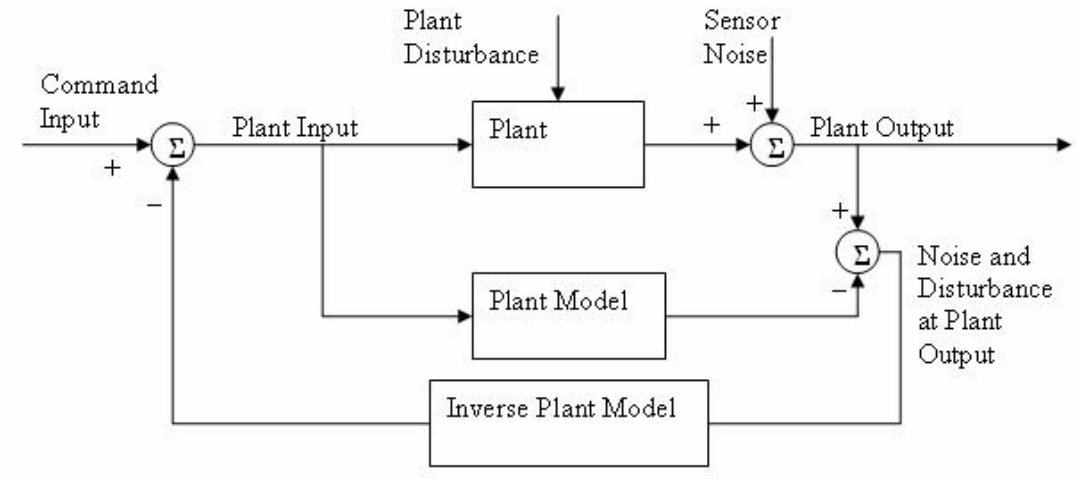

Figure 2.3 : Noise and Disturbance Cancelling

In the Figure 2.3 the plant and plant model has same transfer function, the plant disturbance and sensor noise is filtered via inverse plant model. The filtered signals are subtracted from command input, in order to cancel noise and disturbance at the plant output.

The general view is discussed in this chapter. In next chapters, these concepts will be developed.

3 LINEAR ADAPTIVE FILTERS

The development of adaptive inverse control is based on adaptive inverse filtering. They are used for modelling the plant, controlling the plant and cancelling plant disturbance. This chapter introduces the idea of adaptive linear filtering and performance criteria of minimizing mean square error.

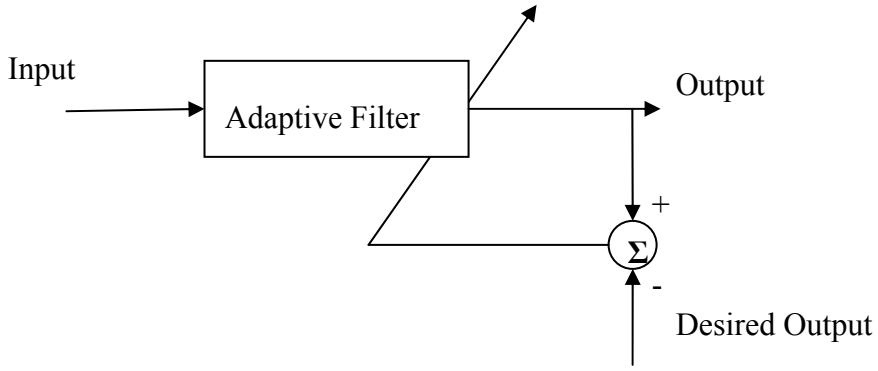

Figure 3.1 : Symbolic Representation of Basic Adaptive Filter

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. [45]

Linear adaptive filters may be divided into two groups: finite impulse response (FIR) and infinite impulse response (IIR). When an FIR filter is excited by an impulse, the response of the filter in non-zero for a finite period of time, on the other hand, the response of the IIR filter is non-zero for an infinite period of time. Adaptive IIR filter is showned in Figure 3.2

Adaptive FIR filter is showned in Figure 3.2. This filter would be used for direct and inverse modelling, which is comprises a tapped delay line, variable weights, a summation block to add weighted signals, and an adaptation process. The inputs of adaptive filter are digitized input and outputs of unknown systems. The weights of

the filter automatically adjusted by adaptation algorithm, which minimizes mean square error, in order to give optimal impulse response..

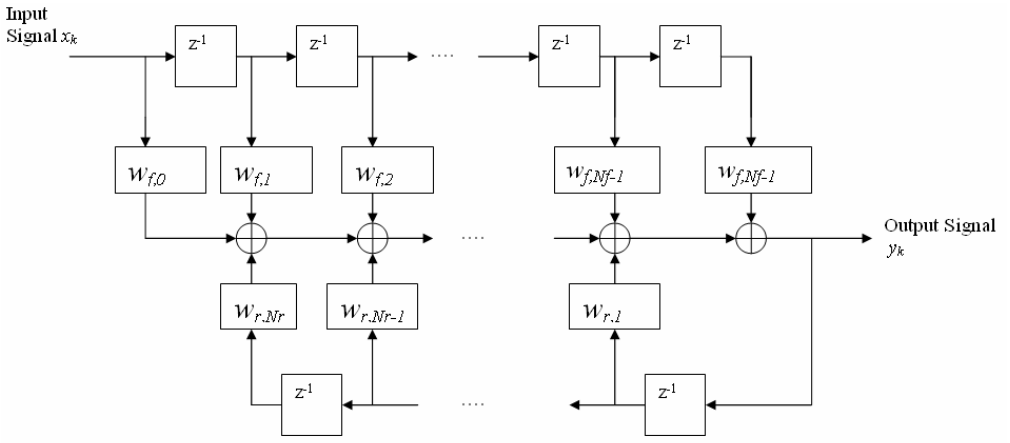

Figure 3.2 : IIR Adaptive Filter

Any stable linear system may be approximated by a "sufficiently long" FIR filter. Hence in this thesis this type of filter is concentrated, also FIR filters are stable with finite weights, IIR filters may not.

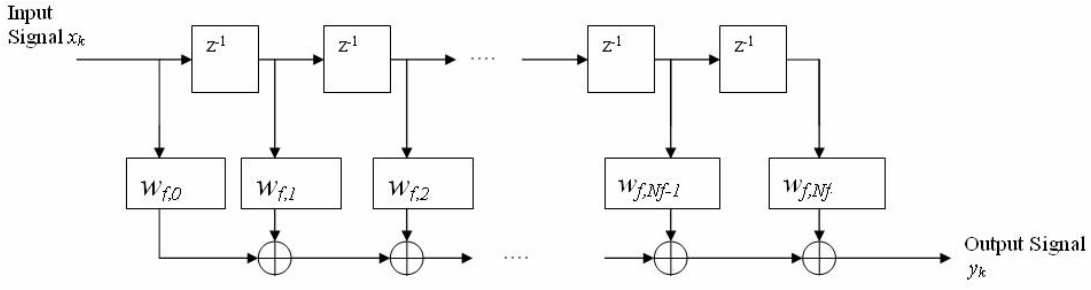

Figure 3.3: Modelling an unknown system by a discrete adaptive filter

3.1 Adapting Linear Filters

Linear adaptive filters are adapted as ; at each time instant, a desired response signal d_k is applied to the filter. The true output y_k, is compared to this desired response, and the error is computed as $\in_k = d_k - y_k$. After that weight are adapted according to minimization of the cost function J_k,

$$J_k = \sum_{j=0}^{k} \left\| \in_k \right\|^2 \qquad \textbf{(3.1)}$$

The cost function J_k, is computed for k time steps, weights are adapted in the direction of the negative gradient of the cost function.

$$W := W - \mu \nabla_W J_k \tag{3.2}$$

In Equation 3.2, W stands for weight vector, the scalar parameter μ is called the convergence factor (learning rate) that controls stability and the rate of adaptation. It is necessary to select convergence factor as below for stability.

$$\frac{1}{\lambda_{\max}} > \mu > 0 \tag{3.3}$$

Where $\lambda_{\max}$ is the eigenvalue of R (please refer Appendix A for R)

A mathematical tool called ordered partial derivatives $\partial^+(.)/\partial x_k$,are used for calculating the gradients. Ordered partial derivatives are useful for easily finding derivatives of complex dynamical systems.

Ordinary partial derivative of (.) with respect to x_k refers the direct causal impact of x_k on (.), while the ordered derivative refers to the total causal impact, including direct and indirect effects [42]. The derivatives of equations which are evaluated in a specific time order may be calculated by the ordered partial derivative easily.

Suppose, for example, there is a function as

$$y_k = f(x_k, x_{k-1}, ..., x_{k-n}, W) \tag{3.4}$$

The main advantage of the ordered derivative is that; complex dynamical systems may be differentiated using a simple chain rule expansion.

$$\frac{\partial^+ y_k}{\partial W} = \frac{\partial y_k}{\partial W} + \sum_{j=0}^{n} \frac{\partial y_k}{\partial x_{k-j}} \frac{\partial^+ x_{k-j}}{\partial W} \tag{3.5}$$

The forward equation for the filter is

$$y_k = WX_k \tag{3.6}$$

where W is the weight matrix of the filter and X_k is a composite vector comprising all of the delayed inputs

$$X_k = \left[x_k{}^T, x_{k-1}{}^T, ...x_{k-Nf}{}^T\right]^T \quad \textbf{(3.7)}$$

The weight vector is

$$W^T = \left[w_0, w_1, ...w_{Nf}\right] \quad \textbf{(3.8)}$$

The gradient of the cost function can be computed with respect to the weights as follows

$$\frac{\partial^+ J_k}{\partial W} = \sum_{j=0}^{k} \frac{\partial^+ \left\| e_j \right\|}{\partial W} \quad \textbf{(3.9)}$$

Which is equal to

$$\frac{\partial^+ J_k}{\partial W} = \sum_{j=0}^{k} -2e_j{}^T \frac{\partial^+ y_k}{\partial W} \quad \textbf{(3.10)}$$

where

$$\frac{\partial^+ y_k}{\partial W} = \frac{\partial y_k}{\partial W} + \sum_{j=0}^{Nf} \frac{\partial y_k}{\partial x_{k-j}} \frac{\partial^+ x_{k-j}}{\partial W} \quad \textbf{(3.11)}$$

$$= diag\left\{X_k^T, X_k^T, X_k^T, X_k^T ....X_k^T\right\} \quad \textbf{(3.12)}$$

$$= \begin{bmatrix} X_k^T & 0 & \cdots & 0 \\ 0 & X_k^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & X_k^T \end{bmatrix} \quad \textbf{(3.13)}$$

Summation term disappeared in Equation 3.11 since x_j is not a function of W

$$\frac{\partial^+ J_k}{\partial W} = \sum_{j=0}^{k} -2e_j^T diag\left\{Xj^T, Xj^T, ..., Xj^T\right\} \quad \textbf{(3.14)}$$

The system run for k time steps $\partial^+ J_k / \partial W$ is computed, and the weighs are updated using Equation 3.2. However if the learning rate is small, adaptation may be done at each time step as Equation 3.15.

$$W_{k+1} = W_k + 2\mu\varepsilon_k diag\left\{X_k^T, X_k^T, ..., X_k^T\right\} \quad \textbf{(3.15)}$$

Which is equal to and easy to use

$$W_{k+1} = W_k + 2\mu\varepsilon_k X_k^T \quad \textbf{(3.16)}$$

3.2 Optimal Solution for Linear Adaptive Filters

In linear systems, MSE performance function is quadratic function of the weights. It is bowl shaped surface and adaptive process continuously adjusts weights in order to find bottom of the bowl. There is one and only one minimum (optimal) solution when the cost function used is MSE, and that gradient descent methods will converge to the solution.

The solution is mathematically tractable if certain statistical information about the input and desired response is available. This solution is known as the Wiener solution.

If the cross correlation function between the input x_k and the desired response d_k, is $(\phi_{xd})_n$ and the input autocorrelation function is $(\phi_{xx})_n$, then the unconstrained solution, $W^{(opt)}(z)$ is

$$W^{(opt)}(z) = \left[\Phi_{xd}(z)\right]\left[\Phi^{-}{}_{xx}(z)\right]^{-1} \quad \textbf{(3.17)}$$

where $\Phi_{xd}(z)$ and $\Phi_{xx}(z)$ are the z-transform of $(\phi_{xd})_n$ and $(\phi_{xx})_n$, respectively. Note that this solution allows for the filter $W^{(opt)}(z)$ to be non-causal. The Shannon-Bode solution for the optimal causal filters is

$$W_{causal}{}^{(opt)}(z) = \left[\left[\Phi_{xd}(z)\right]\left[\Phi^{-}{}_{xx}(z)\right]^{-1}\right]_{+}\left[\Phi^{+}{}_{xx}(z)\right]^{-1} \quad \textbf{(3.18)}$$

where, $\Phi_{xx}(z) = \Phi^{+}{}_{xx}(z)\Phi^{-}{}_{xx}(z)$ and $\Phi^{+}{}_{xx}(z)$ has all the poles and zeros of $\Phi_{xx}(z)$ which are inside the unit circle in the z-plane. Furthermore, the $[.]_+$ operator means, "take the time series generated by the inverse-z-transform of the operand, retain only the causal section (set the non-causal entries to zero), and take the z-transform of the result". Mathematical analysis of a system constrained to be causal is not simple, but in certain specific cases, useful results may be obtained.[35]

For modelling and controlling of the linear systems, linear adaptive filters are used, hence the linear adaptive filters and their properties are discussed in this chapter, in next chapter, nonlinear adaptive filters will be discussed. For more information of linear filters please refer, Appendix A.

4 NON- LINEAR ADAPTIVE FILTERS

Non linear adaptive filers are used to for controlling and plant modelling of non linear plants. For inverse control, the command input is applied to a non linear controller, whose adjustable parameters are adapted, so that when the output of the controller drives the plant input, the plant output becomes a best squares match to the reference model's output. The resulting controller would be a good inverse only for the particular input command signal not in general. If the characteristics of the command input signal were to change, it would be necessary for the controller to adapt rapidly and keep up with the changes. As long as this is feasible, non linear inverse control will work. [1]

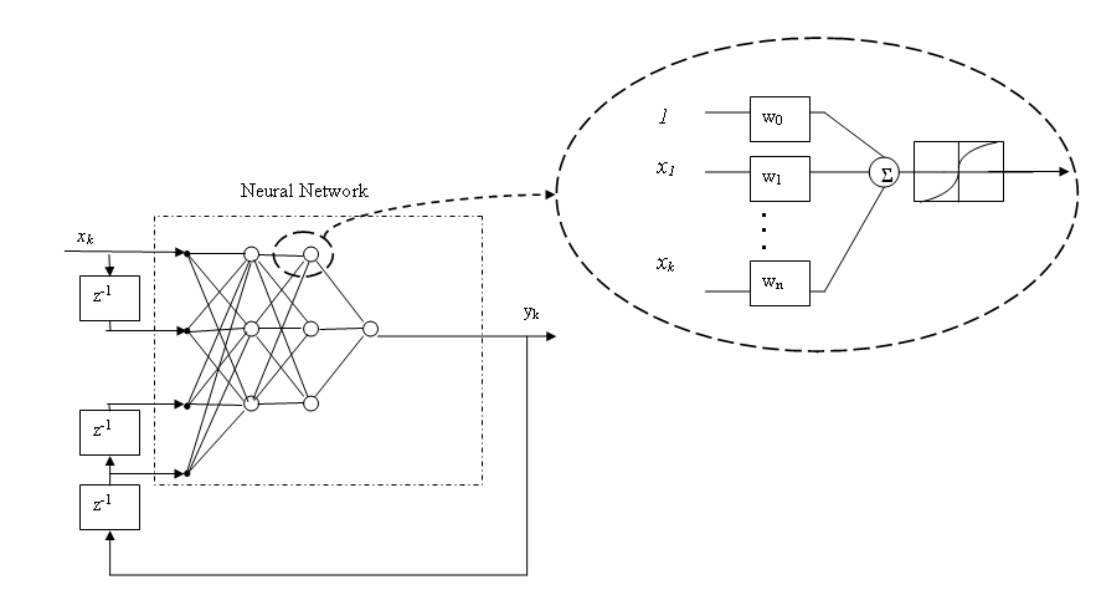

Figure 4.1 : Modelling an Unknown System with Neural Network

Figure 4.1 illustrates the structure of the nonlinear adaptive filters. Input signal is tabbed, possibly the output signal is tapped; the output the filter is nonlinear function of both delayed inputs and outputs. The nonlinear function may be implemented in any way, in this work neural networks are used.

A neural network is an interconnected set of very simple processing elements called neurons. Each neuron computes an internal sum which is equal to a constant plus the

weighted sum of its inputs. [31] The output of the non linear function is called activation function. It is chosen tangent sigmoid in this work.

$$neronoutput = \text{tansig}\left(constant + \sum_{i=1}^{\#inputs} w_i.input \right) \quad \textbf{(4.1)}$$

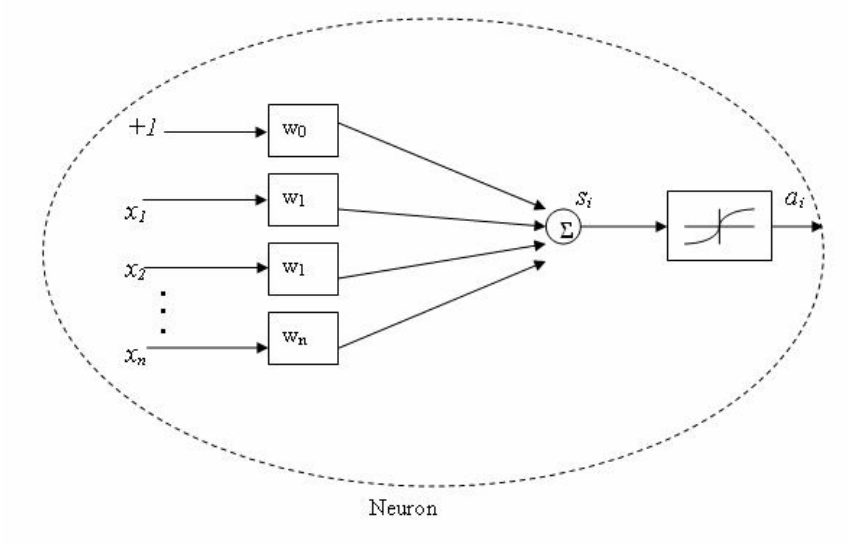

Figure 4.2 : Neuron

The neural networks may be connected randomly, but it is better to arrange them in order. Layers are groups of identical neurons where each neuron has identical inputs, these inputs are the outputs of the previous layer. The final layer of the network is called output layer. The other layers are called hidden layers. The layered network has feedforward structure (non-recurrent) which computes a static nonlinear function. Dynamics are introduced via tabbed delay at the input to the network.

In non linear filter following notation is used: N(a,b):α,β… which means: "The filter input is composed of a tapped delay line with 'a' delayed copies of the exogenous input vector x_k , and 'b' delayed copies of the output vector . Furthermore, there are 'α' neurons in the neural network's first layer of neurons, 'β' neurons in the second layer, and so on." For instance, the filter in Figure 4.1 would be represented as N(2,2):3:3:1

4.1 Adapting Nonlinear Filters

Gradient Descent algorithms may be used to adapt the weights of adaptive nonlinear filter with feedforward and externally recurrent network. Due to the differences in details, two networks are discussed separately.

4.1.1 Adapting a Feedforward Neural Network

A feedforward neural network may be adapted using *backpropagation algorithm*, discovered independently by several researchers [41, 29] and popularized by Rumelhard, Hinton and Williams. [35] In this method weights are updated recursively, based on the error at the output. The figure is shown below.

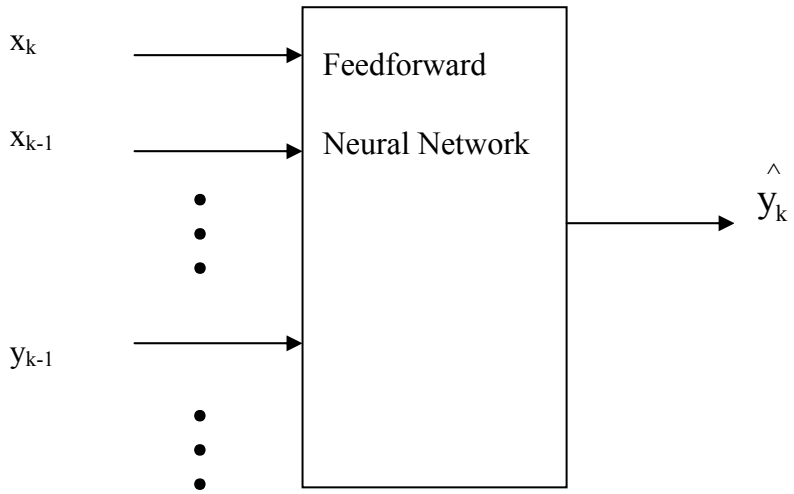

Figure 4.3 : Feedforward Neural Network

Output of network is function of inputs and weight vector

$$y_k = f(x_k, x_{k-1}, ..., x_{k-n}, W) \quad \textbf{(4.2)}$$

The error is the difference between the actual and the desired output. Sum of square of the error, which is called as "cost function" is minimized by adapting the weight vector. It is done by adapting the weighs in the direction of negative gradient of the cost function.

The minimization process is; any arbitrary weight, $w_{i:j}^{(\iota)}$-the one which connects neuron i in the layer l-1 with neuron j in layer l -in network, is selected. The derivative of the error squared with respect to that weight is computed. After calculating the derivatives, weights will be updated. Proceeding the chain rule

$$\frac{\partial^{+}\|\varepsilon\|^2}{\partial w_{i:j}^{(\iota)}} = \frac{\partial\|\varepsilon\|^2}{\partial s_j^{(\iota)}} \frac{\partial^{+} s_j^{(\iota)}}{\partial w_{i:j}^{(\iota)}} \tag{4.3}$$

$$= \delta_j^{(\iota)} a_i^{(\iota-1)}$$

where

$$\delta_j^{(\iota)} \stackrel{\Delta}{=} \frac{\partial\|\varepsilon\|^2}{\partial s_j^{(\iota)}} \tag{4.4}$$

The values of $a_i^{(\iota-1)}$ are known from the forward pass network. Then $\delta_j^{(\iota)}$ should be calculated.

The output layer of the network has no activation function then $y_i = a_i^{(L)} = s_i^{(L)}$, and $\delta_i^{(L)} = -2\varepsilon_i$. In the hidden layers, no specific error signal exists. The chain rule expansion is used for determining an equivalent sensitivity of the output.

$$\delta_i^{(\iota)} \stackrel{\Delta}{=} \frac{\partial\|\varepsilon\|^2}{\partial s_i^{(\iota)}} \tag{4.5}$$

$$= \sum_j \frac{\partial\|\varepsilon\|^2}{\partial s_j^{(\iota+1)}} \frac{\partial s_j^{(\iota+1)}}{\partial a_i^{(\iota)}} \frac{\partial a_i^{(\iota)}}{\partial s_i^{(\iota)}}$$

$$= f'(s_i^{(l)}) \sum_j \delta_j^{(l+1)} w_{i:j}^{(l+1)} \qquad \text{l=1:L-1}$$

Thus $\delta_j^{(l)}$ is calculated by propagating values of $\delta_j^{(l+1)}$ backwards through the network. To sum up, there are two operations in backpropagation forward phase and reverse phase. In forward phase, the input is propagated to the output to compute y_k. In the reverse phase, the error is applied at the output, changed into $\delta_i^{(L)}$ form and $\delta_i^{(L)}$ is propagated backward through the network. The weights are updated using $\delta_i^{(l)}$ and $a_i^{(l)}$ as;

$$\Delta w_{i:j}^{(l)} = -\mu \delta_j^{(l)} a_j^{(l-1)} \qquad \textbf{(4.6)}$$

Calculating Jacobians of Neural Networks:

It is necessary to calculate the Jacobian of the function implemented by a neural network. The neural network is the function of inputs and weights *y=f(X,W)*, the two Jacobians may be calculated as ,

$$\frac{\partial y}{\partial W} \text{ and } \frac{\partial y}{\partial X}$$

The difference between derivation for backpropagation algorithm is the definition of δ_i which is defined as

$$\delta_i^{(l)} \stackrel{\Delta}{=} v^T \frac{\partial y}{\partial s_i^{(l)}} \qquad \textbf{(4.7)}$$

where v is the error vector.

Backpropagation algorithm is used to propagate the redefined δs backward through the network. If δs are propagated to the inputs, and define

$$\delta_i^{(0)} \stackrel{\Delta}{=} v^T \frac{\partial y}{\partial s_i^{(0)}} \qquad \textbf{(4.8)}$$

and

$$\left[\delta_1^{(0)},...,\delta_{Nx}^{(0)}\right]=v^T\frac{\partial y_k}{\partial X} \tag{4.9}$$

Since

$$\frac{\partial s_j^{(\iota)}}{\partial w_{i:j}^{(\iota)}}=a_i^{(\iota-1)}, \tag{4.10}$$

$$\left[.....,\delta_j^{(l)}a_i^{(\iota-1)},...,\right]=v^T\frac{\partial y_k}{\partial W} \tag{4.11}$$

Where the terms in $v^T\,\partial y_k/\partial W$ are ordered according to the same implementation-dependent ordering of $W=\left[\cdots w_{i:j}^{(\iota)}\cdots\right]$

This ability of the backpropagation algorithm is called the "dual-subroutine" introduced by Werbos. Dual subroutine is the recursive calculation of the Jacobians with the network.

4.1.2 Adapting an Externally-Recurrent Neural Network

Up to now computation the Jacobians of network and adapting feedforward network are discussed. These algorithms are extended to externally recurrent neural networks. This was firs done by Williams and Zipser and called "real time recurrent learning" (RTRL). Similar presentation is showned as below.

$$y_k=f(x_k,x_{k-1},.....,x_{k-n},y_{k-1},.....,y_{k-m},W) \tag{4.12}$$

The figure of the externally Recurrent Neural Network is shown in below.

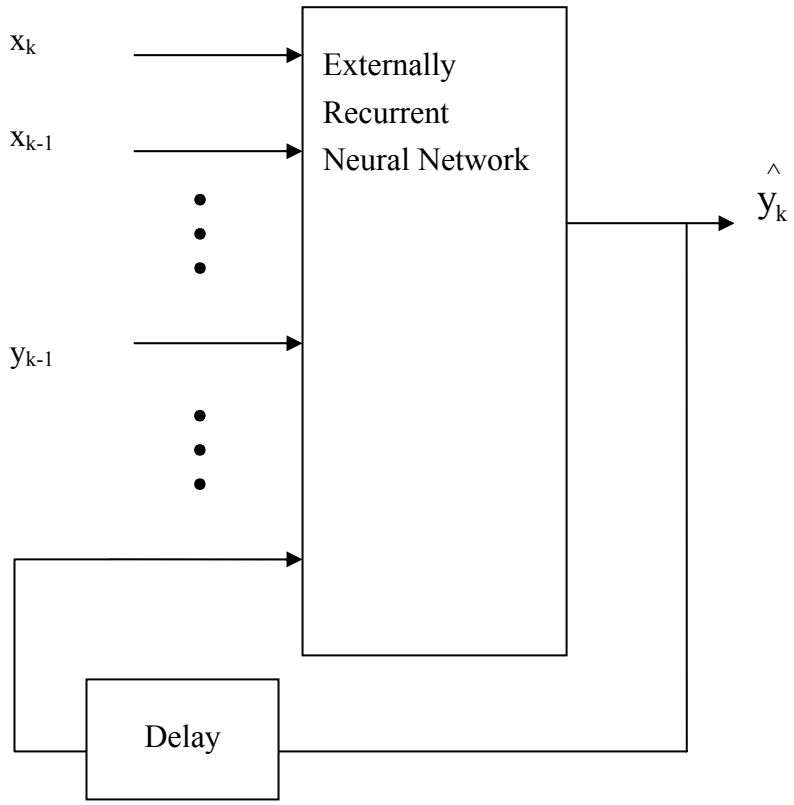

Figure 4.4: Externally Recurrent Neural Network

For calculating sum of squared error;

$$\frac{\partial^+ \|\varepsilon\|^2}{\partial W} = -2\varepsilon_k{}^T \frac{\partial^+ y_k}{\partial W} \tag{4.13}$$

$$\frac{\partial^+ y_k}{\partial W} = \frac{\partial y_k}{\partial W} + \sum_{i=0}^{n} \frac{\partial y_k}{\partial x_{k-i}} \frac{\partial^+ x_{k-i}}{\partial W} + \sum_{i=0}^{m} \frac{\partial y_k}{\partial y_{k-i}} \frac{\partial^+ y_{k-i}}{\partial W} \tag{4.14}$$

The first term $\partial y_k / \partial W$ in Equation 4.14, is a Jacobian, and it has direct effect of a chance in weights on y_k and which is calculated by dual subroutine of the backpropagation algorithm. The second term is zero because x_k is not a function of W, so $\partial^+ x_k / \partial W$ is zero. The last term may split up two parts. $\partial y_k / \partial y_{k-i}$ is a component of matrix $\partial y_k / \partial X$, because the delayed versions of y_k are part of the network's input vector X. The dual subroutine algorithm is used to compute this first

part. The second part is $\partial^+ y_{k-i} / \partial W$ is the previously calculated values of $\partial^+ y_k / \partial W$. When the system is "turned on" $\partial^+ y_k / \partial W$ are set to zero for I=0, -1, -2,,, and the rest of the terms are calculated recursively from that point on.

The dual-subroutine calculates the Jacobians in a way that the weight update is done with simple matrix multiplication, Lets,

$$(d_w y)_k \stackrel{\Delta}{=} \left[\left(\frac{\partial^+ y_{k-1}}{\partial W} \right)^T \left(\frac{\partial^+ y_{k-2}}{\partial W} \right)^T \cdots \left(\frac{\partial^+ y_{k-m}}{\partial W} \right)^T \right]^T \tag{4.15}$$

and

$$(d_x y)_k \stackrel{\Delta}{=} \left[\left(\frac{\partial y_k}{\partial y_{k-1}} \right)^T \left(\frac{\partial y_k}{\partial y_{k-2}} \right)^T \cdots \left(\frac{\partial y_k}{\partial y_{k-m}} \right)^T \right]^T \tag{4.16}$$

The weight update is calculated as

$$\Delta W = \left(2\mu \varepsilon_k^T \left[\frac{\partial y_k}{\partial W} + (d_x y)_k (d_w y)_k \right] \right)^T \tag{4.17}$$

4.2 Optimal Solution for Non Linear Adaptive Filters

In principle, a neural network can emulate a very general nonlinear function. It has been shown that any "smooth" static nonlinear function may be approximated by a two-layer neural network with a "sufficient" number of neurons in its hidden layer [21]. Furthermore, a NARX filter can compute any dynamical finite-state-machine (It can emulate any computer with finite memory) [36].

In practice, a neural network rarely achieves its full potential. In solution space, gradient descent algorithms converges a local minimum, not a global minimum, but a neural network will get quite close to this bound.

The theorem below shows the optimal solution, suppose that input vector of adaptive filter is X_k and the output is y_k, and the desired response is d_k, then the optimal filter shows the function

$$y_k = \mathrm{E}\left[d_x \middle| X_k\right] \tag{4.19}$$

It is proved as; suppose that y_k is the claimed optimal estimate, and $\hat{y}_k$ is another estimate, $\hat{y}_k$ must yield an MSE no smaller than does y_k

$$MSE(\hat{y}_k) = \mathrm{E}\left[\left\| d_k - \hat{y}_k \right\|^2\right] \tag{4.20}$$

$$= \mathrm{E}\left[\left\| d_k - y_k + y_k - \hat{y}_k \right\|^2\right] \tag{4.21}$$

$$\mathrm{E}\left[\left(d_k - y_k\right) \middle| X_k\right] = 0 \tag{4.21}$$

$$\geq MSE\left(y_k\right) + 2\mathrm{E}\left[\left\| \left(d_k - y_k\right)^T + \left(y_k - \hat{y}_k\right)^T \right\|^2\right] \tag{4.22}$$

Recall that $y_k = \mathrm{E}\left[d_x \middle| X_k\right]$ and hence

$$\mathrm{E}\left[\left(d_k - y_k\right) \middle| X_k\right] = 0 \tag{4.23}$$

Since $y_k - \hat{y}_k$ is deterministic function of $\mathrm{X_k}$

$$\mathrm{E}\left[\left\| \left(d_k - y_k\right)^T + \left(y_k - \hat{y}_k\right) \middle| X_k \right\|\right] = 0 \tag{4.24}$$

By iterated the expectation

$$\mathrm{E}\left\{\mathrm{E}\left[\left(d_k - y_k\right)^T + (y_k - \hat{y}_k)\middle| X_k\right]\right\} = \mathrm{E}\left[\left(d_k - y_k\right)^T + (y_k - \hat{y}_k)\middle| X_k\right] = 0 \qquad \textbf{(4.25)}$$

which shows

$$MSE(\hat{y}_k) \geq MSE(y_k) \qquad \textbf{(4.26)}$$

5 SYSTEM IDENTIFICATION

Systems generally work insufficiently, in order to take better performance; the engineers often aim to control the systems. If there is a need of better control, the model of the system should be known. In simple systems, it is reliable to build mathematical model from physics, but in most cases, it is difficult because the system involves complex equations or it is totally unknown, so the states of the system cannot be determined.

For these cases, the model of the system cannot be found by using physical laws. Then, a new method that should be introduced. The procedure to get the mathematical model of the system by using input and output data is called “system identification”.

5.1 Identification of Linear Systems

If a black box model is trying to be identified, the structure of the model may be chosen to be either: a state-space implementation of adjusted by subspace methods[40]; an auto-regressive ARMA IIR filter adjusted by recursive least-squares methods [24]; or an FIR model adjusted by an algorithm such as LMS [44]

They have different advantages. The state-space methods are numerically very robust. The ARMA model has many fewer parameters than FIR, and hence may learn more quickly. The FIR model is very simple and unbiased by zero-mean disturbances if they are uncorrelated with the system input. Adaptive FIR models are used to identify linear system in this thesis.

Black-box adaptive system identification is performed as shown in Figure 5.1. The plant is excited with the signal u_k, and the disturbed output y_k is measured. The plant model $\hat{P}$ is also excited with u_k , and its output $\hat{y}_k$ is computed. The modelling error, which is the difference between the model output and the measured plant output, is used by the adaptation algorithm, to update the weight values of adaptive filter.

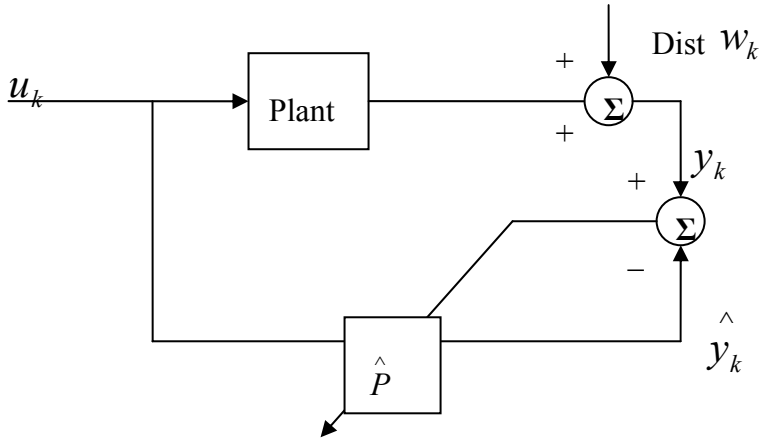

Figure 5.1 : Adaptive System Identification

The desired response of the filter is y_k , the input of the filter is u_k, the output of the filter is $\hat{y}_k$ The input of the filter and the desired response are used to compute Wiener solution for the optimal plant model.

$$(\phi_{uy})_n = \mathrm{E}[u_k - y_{k+n}] \quad \textbf{(5.1)}$$

$$= \mathrm{E}[u_k(p_{k+n} * u_{k+n} + w_{k+n})] \quad \textbf{(5.2)}$$

$$= \mathrm{E}[u_k(p_{k+n} * u_{k+n})]\mathrm{E}[u_k w_{k+n}] \quad \textbf{(5.3)}$$

$$(\phi_{uy})_n = p_n * (\phi_{uu})_n \quad \textbf{(5.4)}$$

where p_k is the impulse response of the plant. If the disturbance is zero-mean and uncorrelated with the plant input, then

$$(\phi_{uy})_n = p_n * (\phi_{uu})_n \quad \textbf{(5.5)}$$

$$\Phi_{uy}(z) = P(z)\Phi_{uu}(z) \quad \textbf{(5.6)}$$

$$\hat{P}^{(opt)}(z) = \frac{\Phi_{uy}(z)}{\Phi_{uu}(z)} \quad \textbf{(5.7)}$$

$$\hat{P}^{(opt)}(z) = P(z) \tag{5.8}$$

Which means adaptive model converges the plant.

5.2 Identification of Nonlinear Systems

Modelling of nonlinear systems, are more complicated than modelling of linear systems. In this work NARX model is used for modelling, since sufficient ordered NARX is a universal appoximater of any universal dynamic system.

The delayed outputs of are the inputs of the model in NARX as a feedback. When training an adaptive plant model two kinds of connections may be applied, parallel connection and series-parallel connection. It is depend on the connection of the feedback. If the feedback is connected model output, it is called parallel connection or the feedback may be connected plant output which is a series connection showned in Figure 5.2.

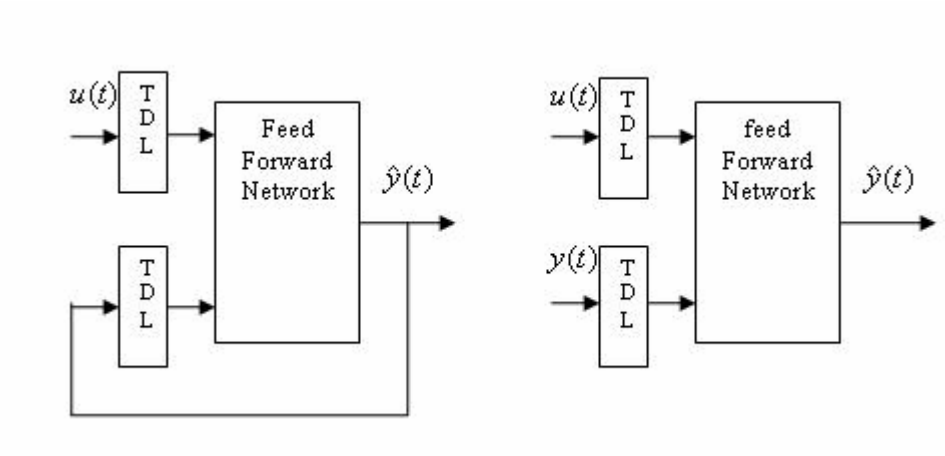

Figure 5.2 : Parallel and Series-Parallel Connection Identification

In parallel connection identification networks are trained either real-time-recurrent-learning (RTRL) or backpropagation-through-time (BPTT). Parallel connection for system identification is simple, but is biased by disturbance. The series-parallel connection for system identification is more complex to train, but is unbiased by disturbance. In this work, nonlinear system identification is done with both configuration.

5.3 Simulation Examples

This part introduces the plants used throughout this thesis as examples of their respective control categories, also modelling of the plants are showned. Both linear and nonlinear systems are examined. The examples were chosen because they are typical of actual control problems, but simple enough to be thoroughly understood. In the following pages, the dynamics of each plant are outlined, reference signals that the plant's outputs are required to track are specified, and the characteristics of expected disturbances are presented.

5.3.1 Linear Plant

The dynamics of linear continuous-time systems may be expressed mathematically in a number of ways. One of these is by linear constant coefficient differential equations. In this work, the both examples are defined by differential equations.

Continuous-time plants are discreatized by realizing that the plant input is held constant for T seconds (where T is the sampling period), and the plant output is sampled every T seconds. The transfer function of the plant in the z-transform domain may then be readily calculated from the transfer function in the (Laplace) s-plane. Notationally, we say

$$H(z) = (1 - z^{-1})\mathbb{Z}\left\{\frac{H(s)}{s}\right\} \qquad \textbf{(5.9)}$$

where the operator Z {.} means "take the inverse Laplace transform of (.), sample the resulting time sequence at *1/T* samples per second, and return the z-transform of the sampled sequence." The resulting transfer function *H(z),* along with its region of convergence in the z-plane, uniquely defines a linear time invariant discrete-time system. Important properties of the system may be quickly deduced from *H(z).* [31]

The roots of its denominator polynomial are called poles, and the roots of its numerator polynomial are called zeros if *H(z)* is in rational polynomial form. If all of the poles are within the unit circle in the z-plane, the system is stable and causal. If any pole is outside the unit circle, the system is either unstable or non-causal. If all of the zeros are inside the unit circle, the system is called minimum phase, that a stable, causal inverse of the system exists. This makes controlling the system relatively easy. If any zero is outside the unit circle, the system is called non-minimum phase, and a stable, causal inverse does not exist. However, a delayed, causal, approximate

inverse does exist, and works very well for controlling such systems. In this thesis only minimum phase plant is introduced for linear example.

The linear example was selected from reference [31]. The goal is to control the temperature of a tank of liquid. The flow-rate of water into the tank is constant and equal to the flow rate of water out of the tank. The temperature of the incoming liquid is controlled by a mixing valve that adjusts the relative amounts of hot and cold supplies of the water (see Figure. 5.3). A length of pipe, assumed to have negligible heat loss, separates the mixing valve from the tank. This distance causes a time delay between the application of a change in the mixing valve and the discharge of the flow with the changed temperature into the tank. If our goal were to design an analog controller for this plant, this time delay significantly complicates the task. No exact analysis techniques are available to handle pure delays, so approximations, such as the Pade Approximation should be used to design the controller.

It is assumed that the mixing in the tank is instantaneous, and that negligible heat is lost in the pipe connecting the valve to the tank, the differential equation governing the tank temperature is

$$\dot{T}_t(t) + \frac{\dot{m}}{M} T_t(t) = \frac{\dot{m}}{M} T_v(t - \tau_d) \qquad \textbf{(5.10)}$$

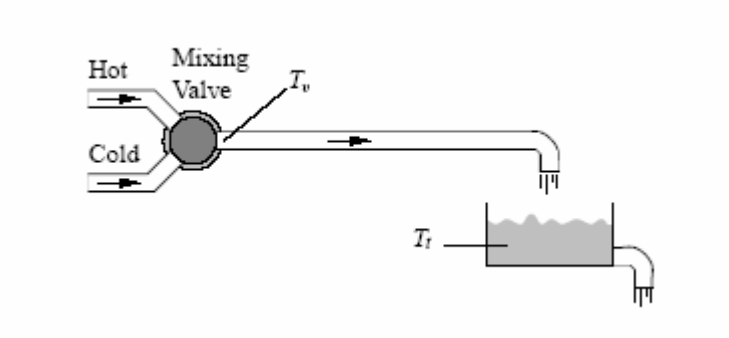

Figure 5.3 : Tank Temperature Control

where

T_v = temperature of water immediately after the control valve and directly controllable by the valve,

T_t =tank temperature,

$\dot{m}$ = mass flow rate, $(\dot{m}_{in} = \dot{m}_{out})$

M =water mass contained in the tank,

τ_d = delay time of water between valve and tank.

When transformed, let $a = \dot{m}/M$, the transfer function from T_v to T_t becomes

$$H(s) = \frac{e^{-\tau_d s}}{(s/a)+1} \tag{5.11}$$

The equivalent z-transform of the discretized plant may be computed as

$$H(z) = (1-z^{-1})\mathbb{Z}\left\{\frac{H(s)}{s}\right\} \tag{5.12}$$

The delay time is approximated by using Pade approximation, as $\tau_d = \theta T - \psi T$ $0 \le \psi < 1$.

$$H(z) = (1-z^{-1})\mathbb{Z}\left\{\frac{e^{-\theta sT}e^{\psi sT}}{s\left[(s/a)+1\right]}\right\}$$

$$= \frac{(1-e^{-a\psi T})}{z^{\theta}} \frac{z + \dfrac{e^{-a\psi T}e^{-aT}}{1-e^{-a\psi T}}}{z-e^{-aT}} \tag{5.13}$$

The location of the zeros of *H(z),* may be considered, θ zeros located at infinity; alternately, θ poles at the origin. These zeros (or poles) correspond to the built-in time delay of the system. Because of them, a non-delayed inverse cannot be constructed. However, a perfect stable and causal inverse with delay of at least θ time steps may be realizable.

The remaining (finite) zero of *H(z)* will be either inside or outside the unit circle, depending on the value of τ_d. If the zero is inside the unit circle, the system is minimum phase. A perfect stable and causal delayed inverse may be constructed. If the zero is outside the unit circle, the system is non-minimum phase.

m = 2 kg/s;

M = 10 kg;

T = 1 s;

θ =2 time steps

The value ψ for was chosen to be 0.55 so finite zero is inside unit circle. With all the variables substituted, the transfer functions become:

$$H(z) = 0.1042 \frac{z + 0.7402}{z^3 - 0.8187 z^2} \quad \textbf{(5.14)}$$

The transfer function is realized through the following difference equation

$$y_k = 0.8187 y_{k-1} + 0.1042 u_{k-2} + 0.0771 u_{k-3} \quad \textbf{(5.15)}$$

The bode diagram, pole-zero map and impulse response of the transfer function are presented in Figure 5.4, Figure 5.5 and Figure 5.6 respectively.

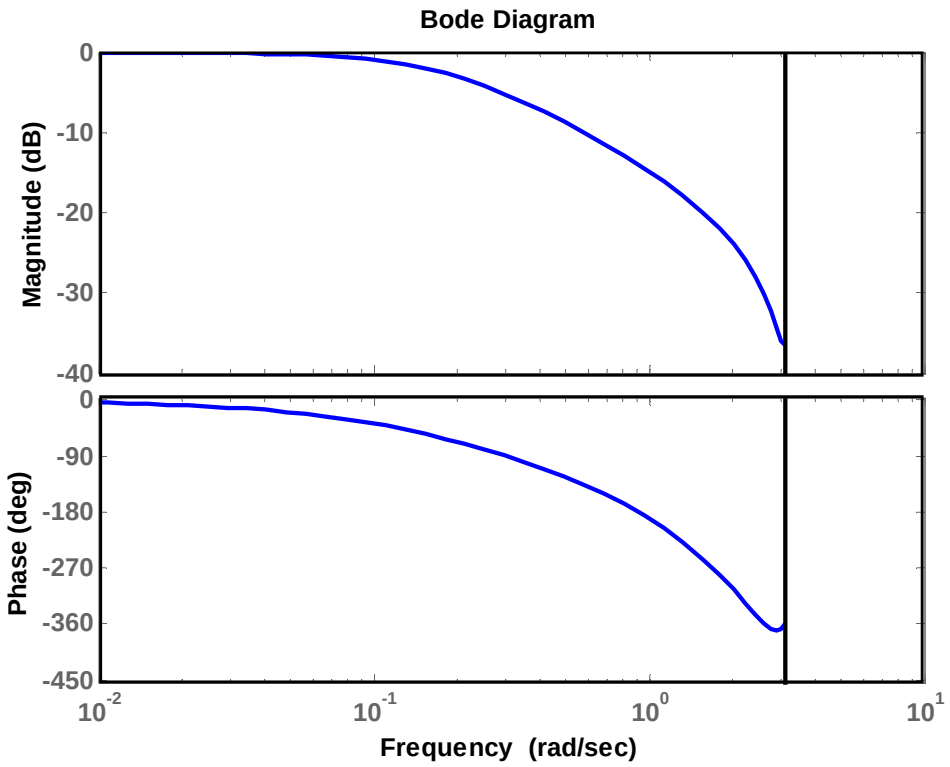

Figure 5.4 : Bode Diagram of Linear Plant

Range of Operation: The reference signal of the plant will be required to track when performing simulations. When constraints on the control effort are considered, they will be as follows: The control effort is allowed to be in the range -0ºC to 0.1ºC. Physically, this means that the hot reservoir is a 0.1 ºC hot liquid source, and that the cold reservoir is a 0ºC cold liquid source. These constraints will be taken into account in control phase.

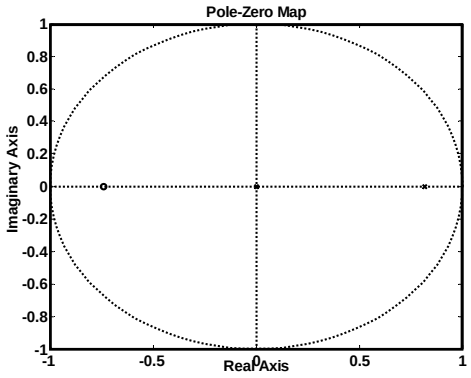

Figure 5.5 : Pole Zero Map of Linear Plant

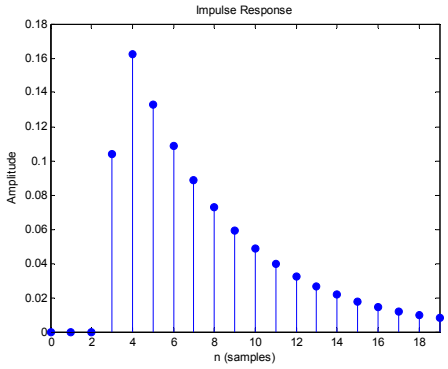

Figure 5.6 : Impulse Response of Linear Plant

First of all the aim is to identify the plant, as accurate as possible, it is known that, the input signal should include plants cut of frequency, which is 0.02Hz. A sine wave may be adequate to achieve this, since sine wave is applied to all filters and neural networks.

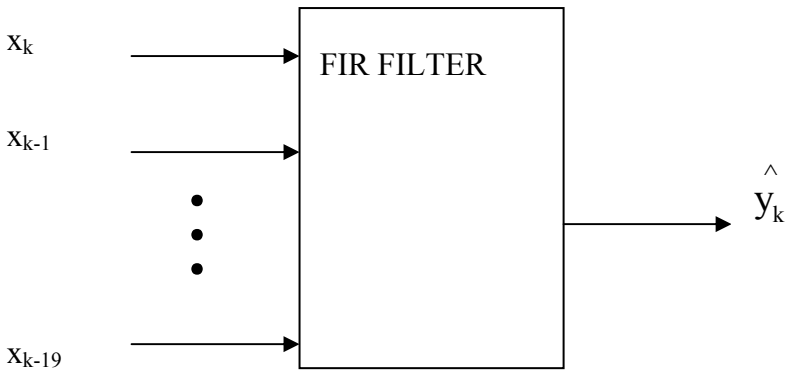

Figure 5.7: 20 Tab FIR Filter

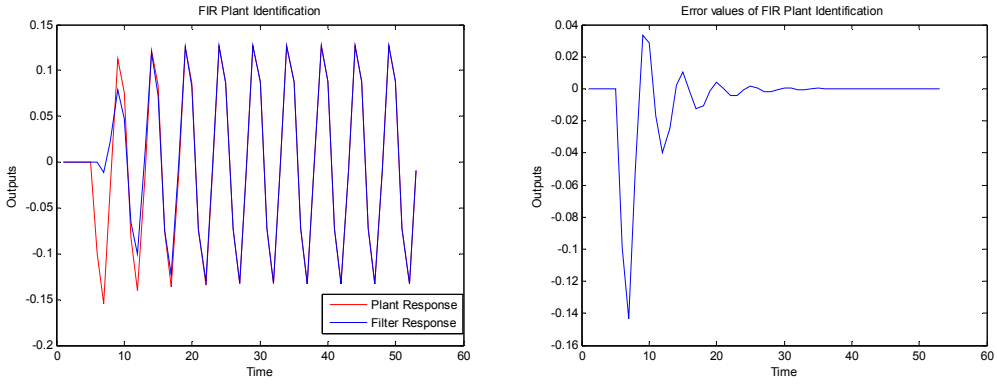

Figure 5.8: Linear Plant Identification with FIR Filter, μ=0.1

First example 20 tab FIR filter shown in Figure 5.7 with μ=0.1, is used with 60 time steps, the simulation results of error and identification are showned on Figure 5.8.

The learning rate is changed to μ=0.01, all the other remain unchanged in second simulation, is showned on Figure 5.9, as seen the leaning rate has direct effect on learning, the smaller values of μ causes slower adaptation. In third simulation, the value of is changed to μ=0.8, which causes unstability, is showned on Figure 5.10.

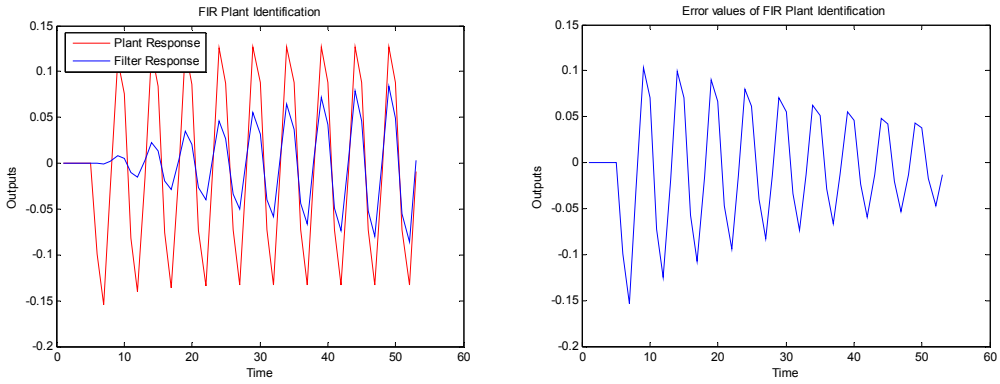

Figure 5.9: Linear Plant Identification with FIR Filter, μ=0.01

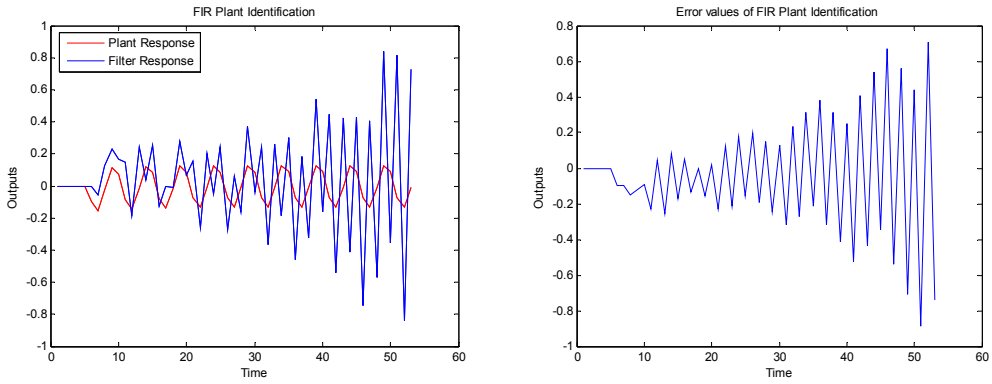

Figure 5.10 : Linear Plant Identification with FIR Filter, μ=0.8

After the investigation the plant identification via FIR filter, the plant identification is also done using neural networks. In first example the identification of plant is done by feedforward neural network, $NN_{(20,1),\ 5,1,\ \ (}x_k,\ x_{k-1},..,\ x_{k-19},y_{k-1)}$, shown in Figure 5.11 μ is selected μ=0.1, and in order to use a nonlinear activation function, a linear activation function y=x is used. The training and validation results are showned in Figure 5.12. As same learning rate is used, neural network has slower adaptation.

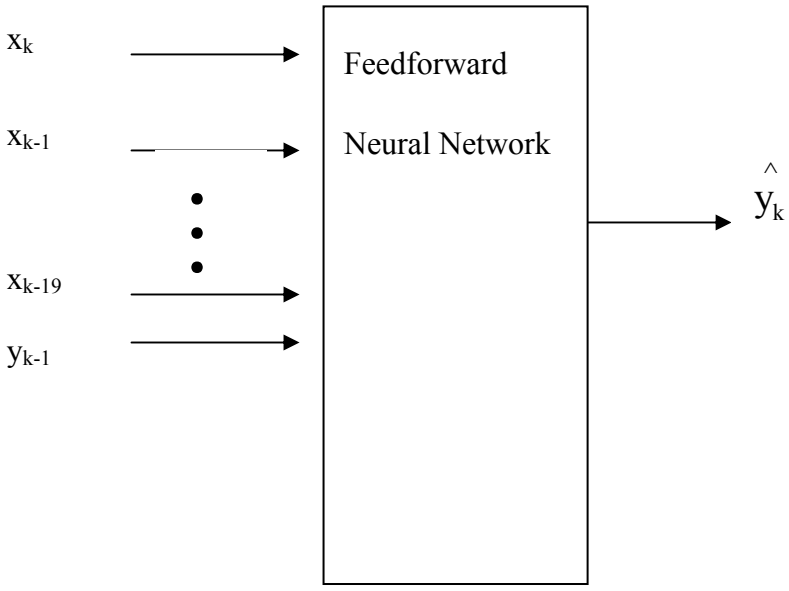

Figure 5.11 : Feedforward Neural Network

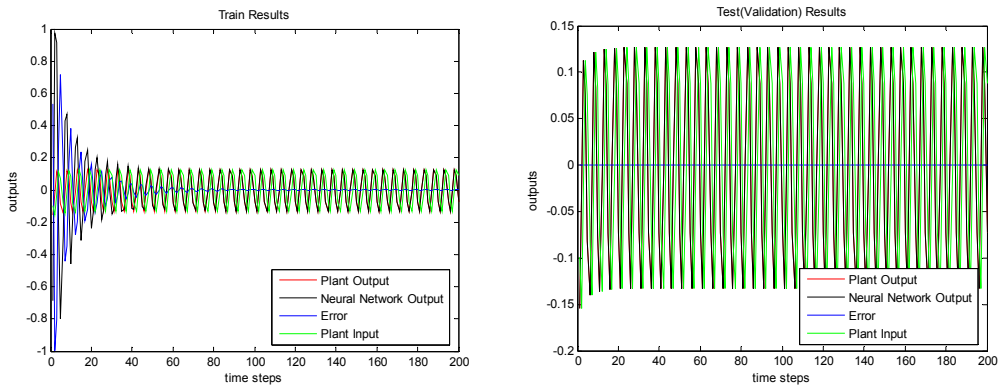

Figure 5.12 : Linear Plant Identification with Feedforward NN, with μ=0.1 linear activation function and 200 time steps

A second neural network identification is done using μ=0.01 all other is unchanged, NN $_{(20,1),5,1}$ (x_k, x_{k-1},.., x_{k-19},y_{k-1}) is showed in Figure 5.13 and Figure 5.14 shows training and validation results. As showned on figure there is not a good result is achieved for 200 time steps if the time step is increased better results will be achieved.

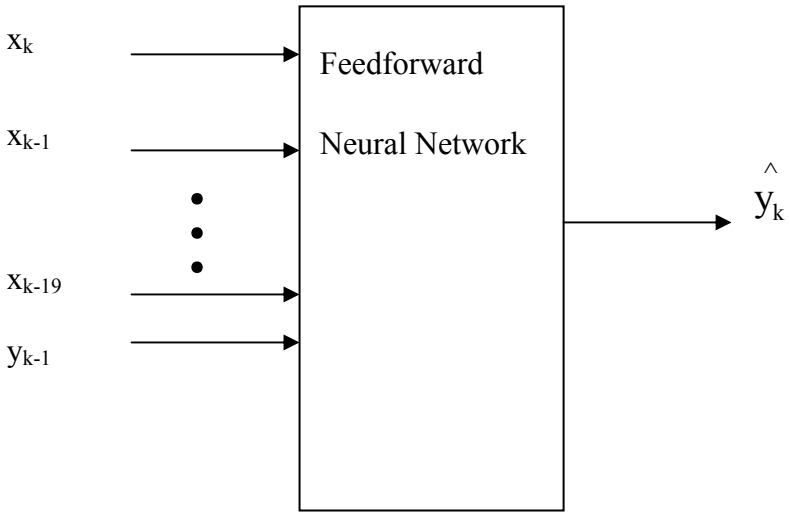

Figure 5.13 : Feedforward Neural Network

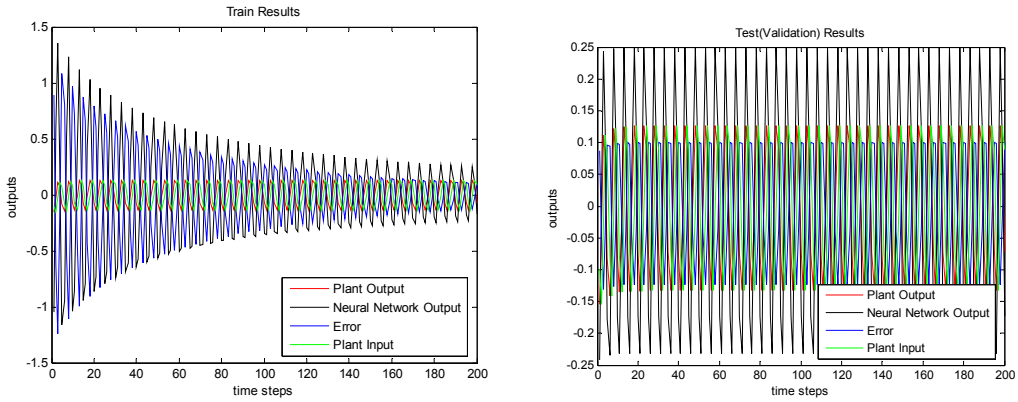

Figure 5.14 : Linear Plant Identification with Feedforward NN, with μ=0.01 Using Linear Activation Function for 200 Time Steps

Third neural network identification is examined by changing the activation function to a nonlinear function tangent sigmoid. The configuration is showed in Figure 5.15. Learning rate is selected as μ=0.1, the results are showned in Figure 5.16. If the results are compared with first neural network example, it may be seen that an error which is caused by nonlinearity in activation function can not be eliminated for 200 time steps.

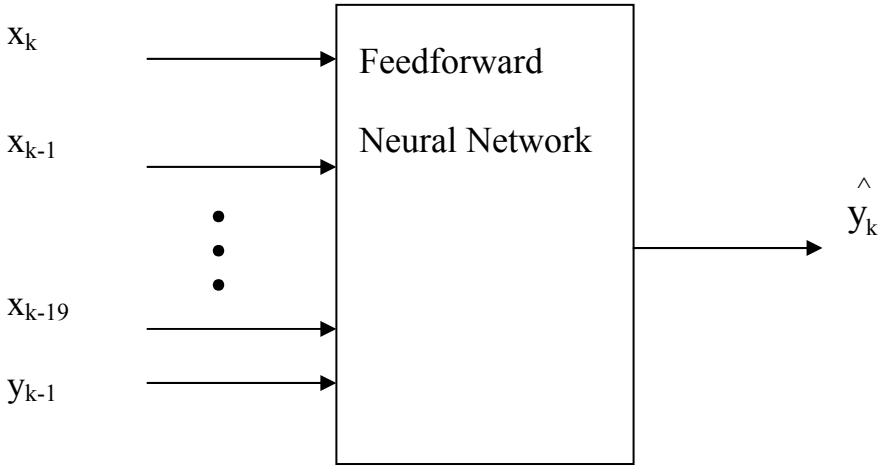

Figure 5.15: Feedforward Neural Network

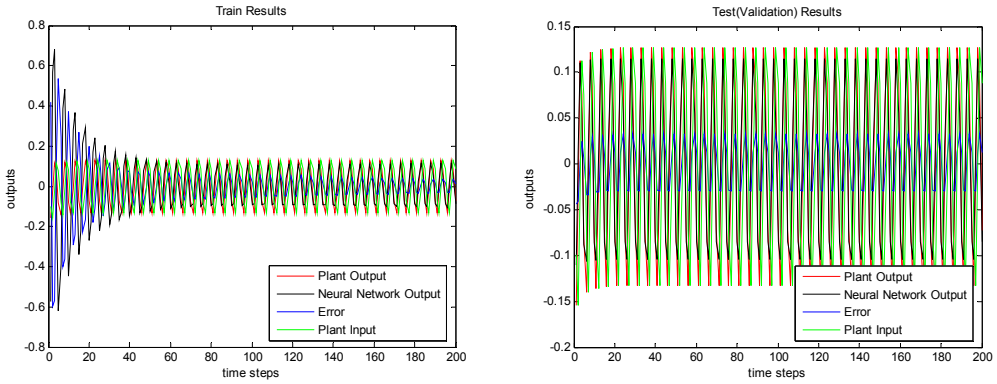

Figure 5.16: Linear Plant Identification with Feedforward NN, with μ=0.1 Using Nonlinear Activation Function for 200 time steps

In this part externally-recurrent neural network with one output from itself, $NN_{(20,1),5,1}$ is used for identification of linear plant which is showed in Figure 5.17. First example is done using with μ=0.1, and linear activation function y=x, is used. Training and validation results are showned in Figure 5.18. Better results are obtained by externally-recurrent neural network, compared to feedforward neural network.

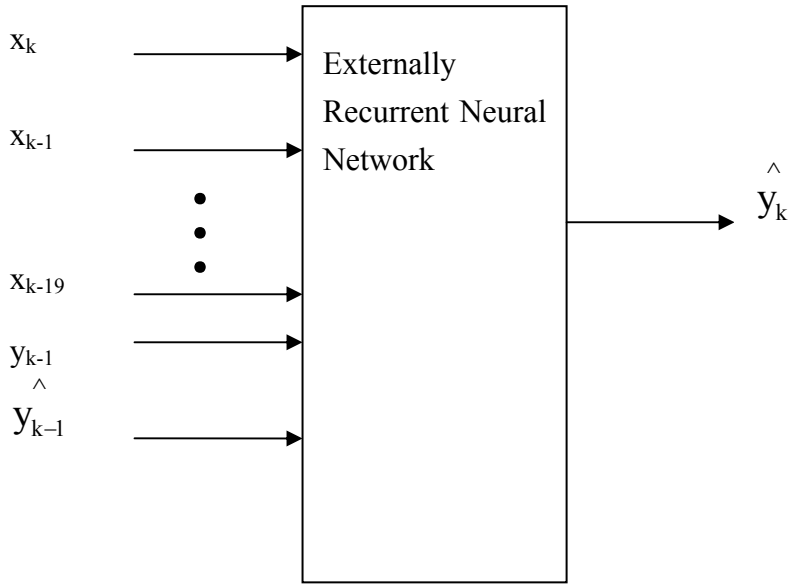

Figure 5.17: Externally Recurrent Neural Network

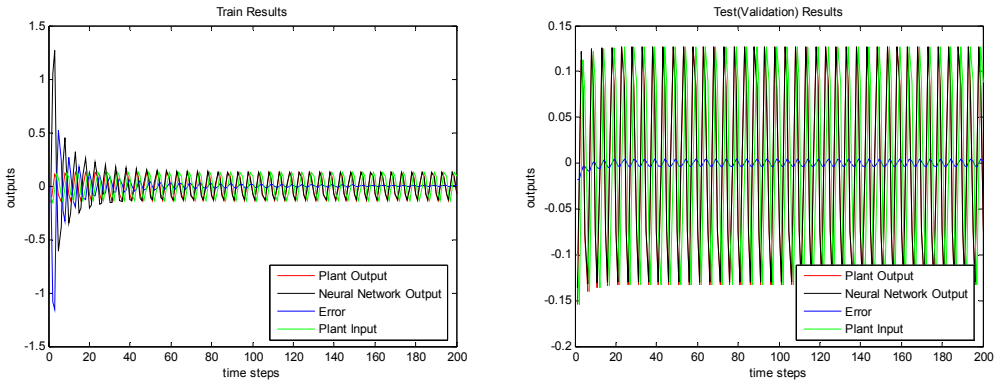

Figure 5.18 : Linear Plant Identification with Externally-Recurrent NN with Using Linear Activation Function for 200 Time Steps

Second example is done by using learning rate μ=0.1, and nonlinear activation function tangent sigmoid. Training and validation results are showned in Figure 5.19. There is no such a big difference using linear or non linear activation function in externally recurrent neural network, compared to feedforward neural network (See Figure 5.12). The best result is obtained with FIR filter with learning rate μ=0.1, than externally recurrent neural network with linear activation function and with learning rate μ=0.1, and the worse result is obtained with FIR Filter with learning rate μ=0.8, As shown in figures learning rate has direct effect on adaptation, if it is chosen small a slow adaptation is obtained, if it is chosen big, unstability may obtained. A linear system may be identified with neural networks. Better results are obtained with linear activation function, also nonlinear activation functions may be used for identification, although they cause slower learning.

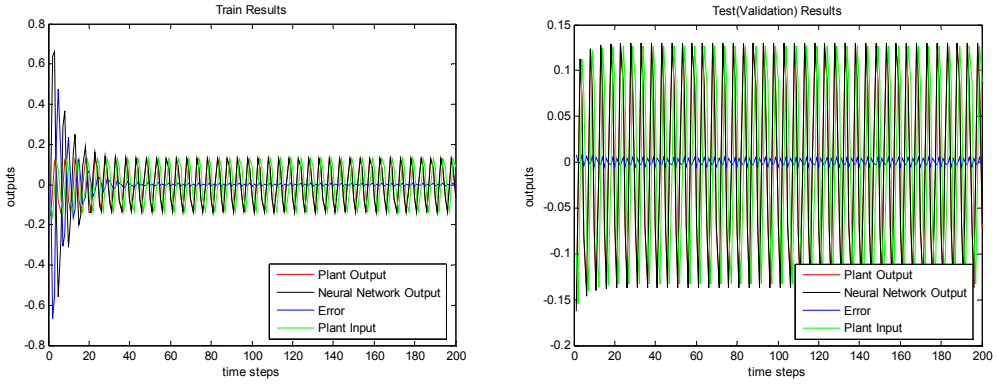

Figure 5.19 : Linear Plant Identification with Externally-Recurrent NN with μ=0.1 Using Nonlinear Activation Function for 200 Time Steps

Disturbances: There are several possible sources of disturbance for this particular plant. There could be heat loss in the pipe between the valve and the tank, loss in the tank itself, non instantaneous mixing in the tank, or poorly regulated hot and cold reservoirs. It is quite reasonable to assume that the pipes and tank are very well insulated, and so heat loss is not considered to be significant. In this section, disturbance is set to zero. For other sections it is assumed that the hot source is poorly regulated. It heats up and cools down in a periodic fashion. Let

T_h=0.1+0.005sin(2лt/60+φ) **(5.16)**

where φ is random variable, uniformly distributed between [−л; л], and independent of u_k.

$$dist(k) = 0.005\sin((2\pi t / 60) + \varphi)) \quad \textbf{(5.17)}$$

5.3.2 Nonlinear Plant

Unlike linear systems, nonlinear systems do not satisfy the superposition principle. Therefore, they cannot be described in terms of impulse responses or transfer functions. They may be described in the time domain. Continuous time systems may be described by systems of nonlinear differential equations, and discrete time systems may be described with sets of nonlinear difference equations. [31]

It is not always possible to analytically discreatize a set of nonlinear differential equations. In many cases it is necessary to discreatize the plant by simulating (numerically integrating) the differential equations over the sampling period. At times, dozens of integration steps need to be taken to advance the system from its current state to its state after a sampling period. Some work has already been done in the area of nonlinear adaptive inverse control [2, 4].

This work focused on nonlinear a SISO plant and successful feedforward control was achieved. It is considered difficult nonlinear control problems with greater practical motivation. Much more insight may be gathered from studying them since we already have an expectation of what their limits of performance might be. The dynamics are computed by simulating the continuous-time differential equations since they cannot be analytically discretized.

The control problem is; autopilots for ships are often designed to keep the ship's heading angle in a desired direction (see Figure 5.20). There are some applications, such as course changing and turning, however, where it is desirable to be able to track a time-varying reference direction. This scenario was selected as an example of

a very nonlinear control problem. The primary reference is [39] but [30] and [31] was also consulted for additional insight into the meanings of some of the parameters involved.

The model of a ship may be expressed as

$$\ddot{\psi} + kd(\dot{\psi}) = k\delta_r \qquad \textbf{(5.18)}$$

Where $\psi(t)$ is the heading angle of the ship, δ_r is the rudder and $d(\dot{\psi})$ is a damping term of the form

$$d(\dot{\psi}) = d_3\dot{\psi}^3 + d_2\dot{\psi}^2 + d_1\dot{\psi} + d_0 \qquad \textbf{(5.19)}$$

Because of the symmetry, most ships have the property that $d_2 = d_0 = 0$

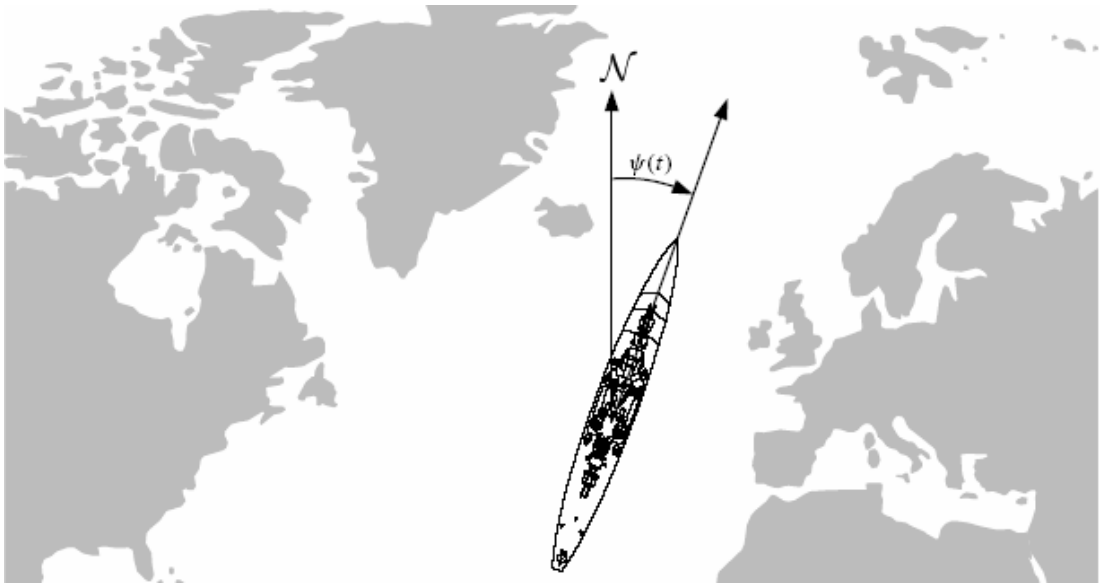

Figure 5.20 : Illustration of Heading Angle

There is a nonlinear dynamical relationship between its heading and rudder angle, also a dynamical relationship with rudder angle with respect to the (steering) wheel position. The rudder angle δ_r does not follow the wheel angle δ_w exactly. The rudder is rate-limited to 6º per second until $|\delta_w - \delta_r| \leq 3$º then the rudder operates in the linear range of its characteristic. One other restriction is that the rudder angle may not exceed 35º in either direction. The ship dynamics may be represented as shown in Figure 5.21

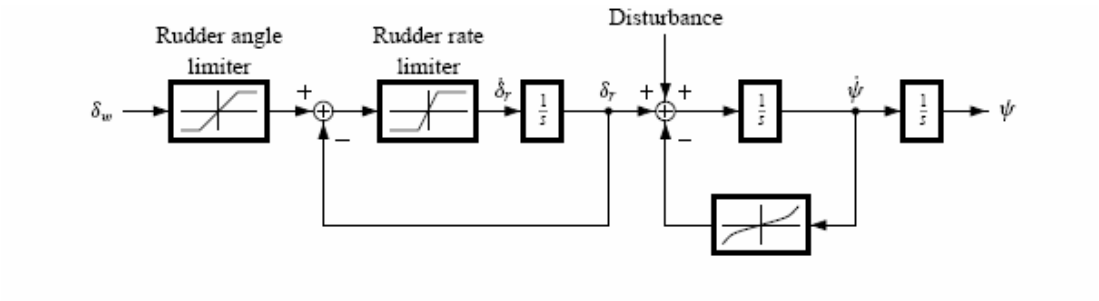

Figure 5.21 : Block Diagram of Ship Yaw Dynamics from Wheel Angle δ_w to the Heading Angle ψ

To summarize, the system dynamics are controlled by the coupled pair of differential equations

$$\ddot{\psi}(t) = k[\delta_r(t) + disturbance(t) - d(\dot{\psi}(t))] \tag{5.20}$$

$$\dot{\delta}_r(t) = 6sat[\frac{35sat(\frac{\delta_w(t)}{35}) - \delta_r(t)}{3}] \tag{5.21}$$

$$sat(x) = \begin{cases} -1, & x < -<-1 \\ x, & -1 \le x < 1 \\ 1, & \text{otherwise} \end{cases} \tag{5.22}$$

Constants are added to convert from degrees to radians and to allow the use of the normalized saturation function sat(x). All parameters were taken from references [39, 31].

k = 0:0107, d_1 = 9:42; d_3 = 2:24;

So it corresponds to the dynamics of a Royal Navy warship traveling at sixteen knots.

Stabilizing the Dynamics:

The dynamics of the ship are unstable. This may easily be seen by applying a step function to the control input. The plant output for step inputs ranging in magnitude of 180 is shown in Figure 5.22(a). A bounded input does not produce a bounded output, and hence the dynamics are unstable. A very simple feedback circuit may stabilize the dynamics. The stabilized ship block diagram is shown in Figure 5.23. The step

response of the stabilized dynamics is shown in Figure 5.22(b). Now, bounded inputs produce bounded outputs. In fact, the feedback loop makes a pretty good control system all by itself. The nonlinear controller will enhance the dynamic response where it can (when the rudder rate and angle limits are not saturated).

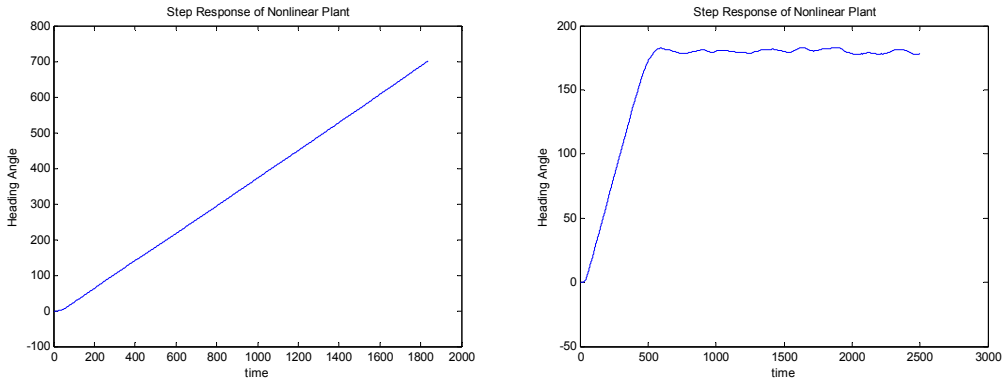

Figure 5.22 : Step Response of Ship Yaw Dynamics with Stabilized and Non-Stabilized Plant

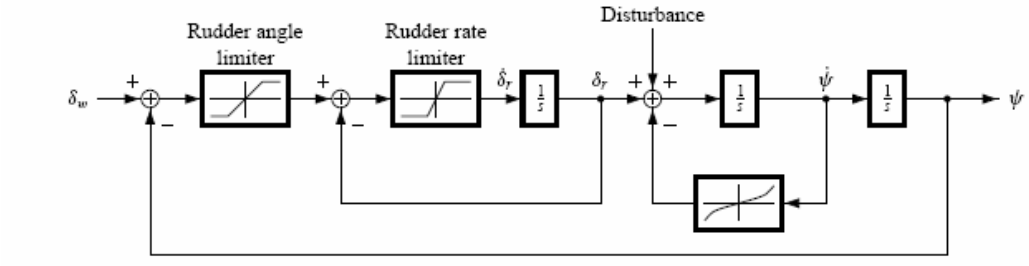

Figure 5.23: Stabilized Plant

Range of Operation: Since the nonlinear controller can not improve performance when the rudder dynamics are in their saturation region, only relatively small perturbations around a fixed heading need to be considered. A default "steady" heading of 0º was used, with perturbations limited to ±30º around that heading. More specifically, the reference command to be tracked was a first-order Markov process, generated by filtering i.i.d. uniform random numbers with maximum magnitude 0.05 using a one-pole filter with the pole at z = 0.99.

Disturbances: The disturbances experienced in the dynamics of the ship are caused almost exclusively by the action of sea waves acting on the rudder angle, and by wind acting on the superstructure. Here, we consider only the effects of waves, as is done in reference [39, 31]. The power spectral density of wave height as a function of wave frequency is

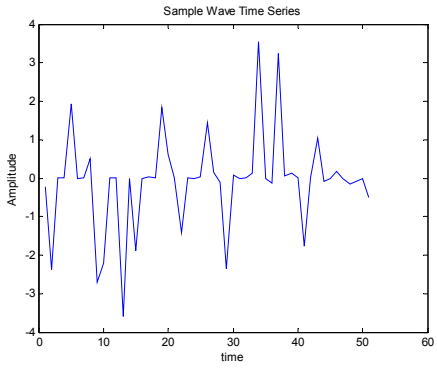

Figure 5.24 : Disturbance Signal

$$S(f) = \frac{\alpha g^2}{(2\pi)^4 f^5} \exp[-\beta \left\{ \frac{g}{2\pi U_{19.5} f} \right\}^4] \qquad \textbf{(5.23)}$$

where

f =wave frequency (Hz)

α =Phillips constant ($8.1x10^{-3}$)

β =dimensionless constant (0.74)

$U_{19.5}$ =wind velocity 19.5m above sea level (knots)

g =acceleration due to gravity

The nominal wind velocity $U_{19.5}$ was taken to be 20 knots.

In order to generate wave disturbances, i.i.d. uniformly distributed random numbers with maximum magnitude 1 are passed through a filter having the same power spectral density *S(f)* Using the random uniform input and this filter, a sample wave time series is plotted in Figure 5.24

In linear plant, the effect of learning rate and activation function are discussed, this part is focused on the number of delayed inputs, and also a comparison between feedforward and externally-recurrent neural network. In all examples learning rate is set to μ=0.1 and tangent sigmoid is used as activation function.

In first example a feedforward neural network *N(4,0),20:1* $(x_k, x_{k-1},.., x_{k-3,})$ is used to identify the system. It takes only the input signal, result is showned in Figure 5.25.

The network is not capable of identifying the plant. The number of inputs are increased in second example that a feedforward neural network $N_{(6,1),20:1}(x_k, x_{k-1},.., x_{k-5},y_{k-1})$ is used for identification. This network is capable to identify the model of the system. Results are showned in Figure 5.26. The third example is a comparison between feedforward and externally-recurrent neural network that externally recurrent neural network, with one delayed output signal from itself, *N(6,1),20:1* is used for identification the plant. Result is showned in Figure 5.27. As shown in figures, both methods are capable to identify this system if sufficient number of delayed inputs are given to the system.

Up to now identification of both linear and nonlinear plants are investigated. In next chapters, these plants are tried to be controlled with FIR filters and neural networks, than the disturbance is tried to be cancelled.

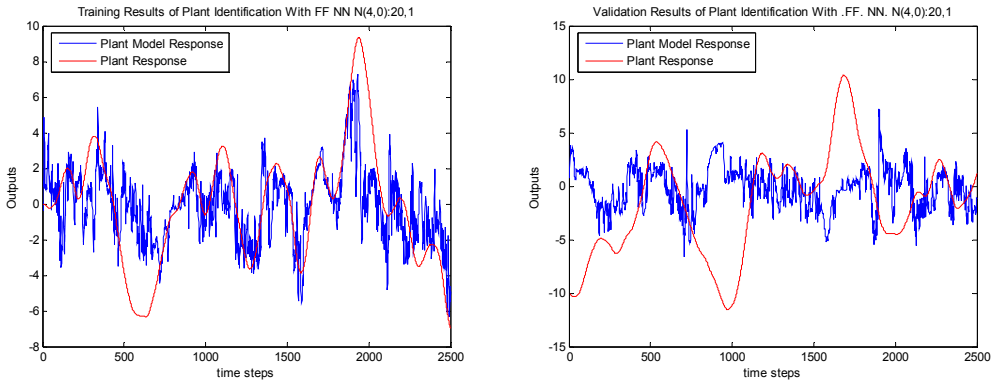

Figure 5.25 : Training and Validation Results of Plant Modelling with Feedforward Neural Network

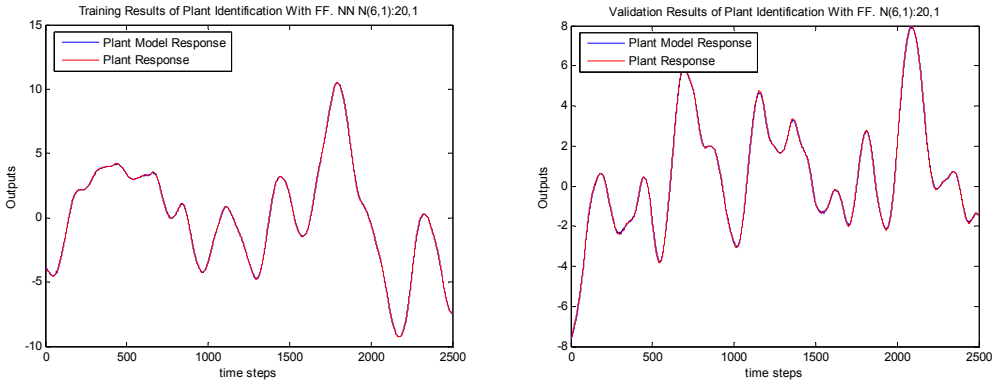

Figure 5.26 : Training and Validation Results of Plant Modelling with Feedforward Neural Network with more inputs.

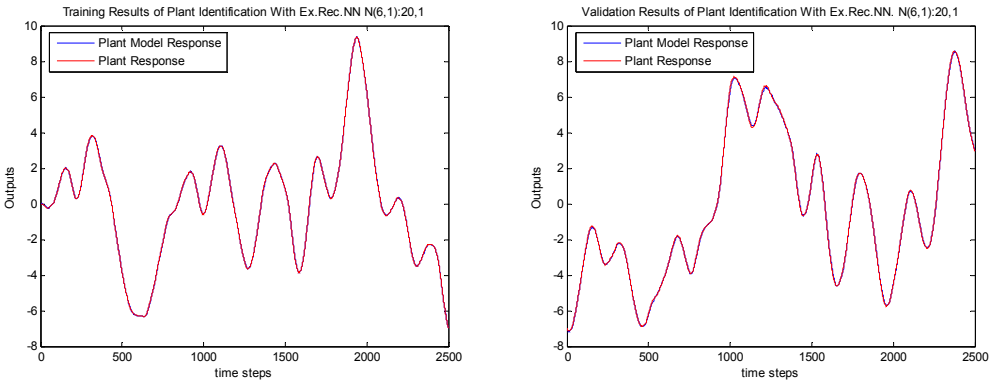

Figure 5.27 Training Results and Validation Results of Plant Modelling with Externally Recurrent Neural Network

6 ADAPTIVE FEEDFORWARD CONTROL

6.1 Introduction

Three filters the plant model P, the controller C, and the disturbance canceller X, need to be adapt to perform adaptive inverse control on Figure 6.1. How to adapt P to make a plant model has already been addressed. This chapter presents an algorithm which may be used to train C to perform constrained model-reference based control of a linear or nonlinear plant.

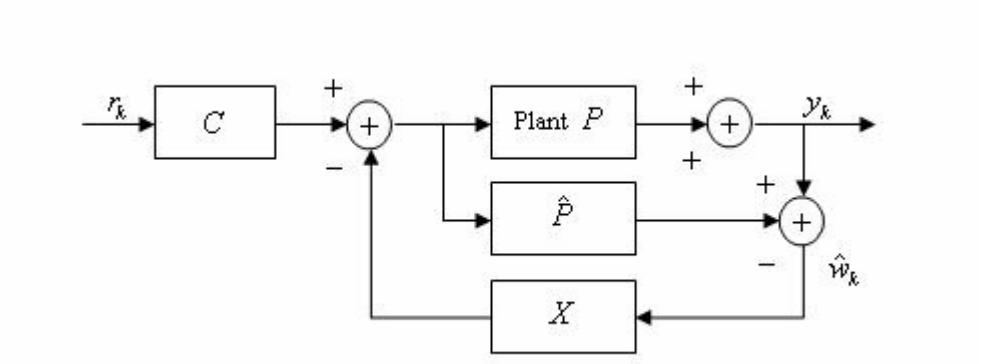

Figure 6.1 : Adaptive Inverse Control

This chapter is organized into two parts. The first part develops an algorithm to train a controller to perform constrained control, and discusses an efficient implementation. The second part presents results from simulations for the plants.

Optimal controller is a controller which minimizes the mean-squared system error. Due to constraints on its architecture, controller may not achieve this level of performance. For example, the controller is restricted to be a linear system, or to be causal, so it may be said the “optimal controller” as being the one which minimizes the mean-squared system error while satisfying the architecture constraints. Note that; the optimal solution for a linear system is the Wiener solution is known from Chapter 3

6.2 Constrained Controller via BPTM Algorithm

This part presents an algorithm, which trains a controller to perform constrained model-reference based control of a linear or nonlinear plant. A key hurdle which must be overcome by the algorithm is to find a mechanism for converting the system error to an adaptation signal used to adjust C. We need some functional block which uses the system error and some form of plant state information to compute the controller error. This block is denoted as "?" in Figure 6.2

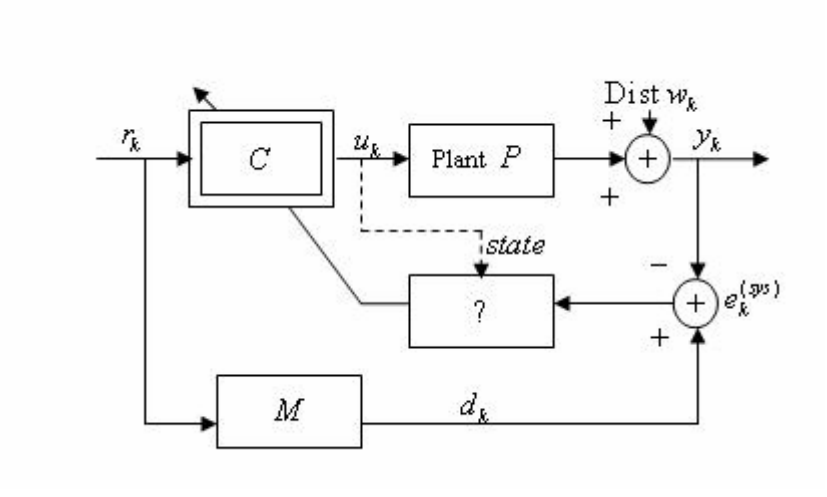

Figure 6.2 : Adaptive Inverse Control

This functional block must describe an algorithm which also satisfies the following design criteria:

- The algorithm must work with linear and nonlinear plants.
- The algorithm must not be biased by disturbances.
- The algorithm must work for autoregressive implementations of $\hat{P}$ and *C*.
- The algorithm must minimize a cost function of the system error and the control effort.

Figure 6.3 shows the general framework to be used. Rather than manipulating the block diagram to generate an indirect error signal with which to adapt *C*, the system error signal is used directly. It is back-propagated through the plant model, and used to adapt the controller. For this reason, the algorithm is named "Backpropagation Through (Plant) Model" (BPTM).[31]

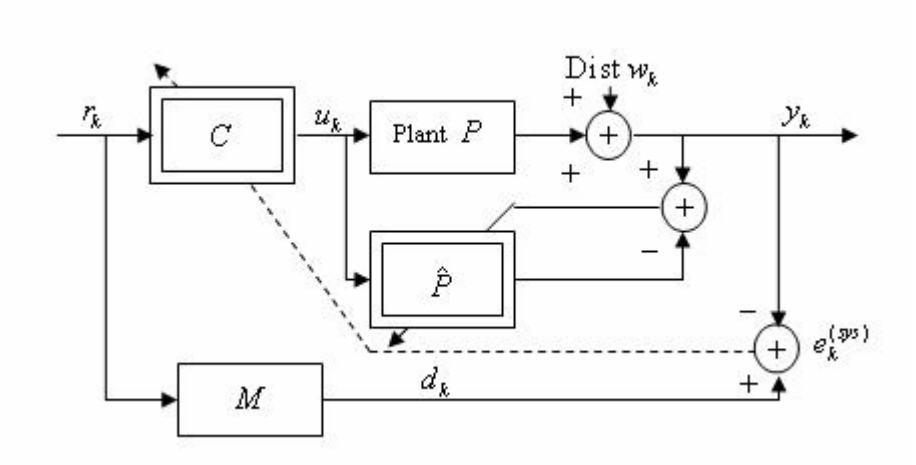

Figure 6.3 : Structure Diagram Illustrating BPTM Method

The aim is to train the controller C, to minimize the squared system error over a certain trajectory and to simultaneously minimize some function of the control effort. The system is run for K time steps. At the end of the K time steps, the following sum is computed

$$J_K = \sum_{j=0}^{K} \left\{ e_{jk}^{(sys)^T} Q e_{jk}^{(sys)^T} + h\left(u_j, u_{j-1}, \cdots, u_{j-r}\right) \right\} \tag{6.1}$$

The function h(.) which can be differentiable, defines the cost function associated directly with the control signal u_k , and is used to penalize excessive control effort, slew rate and so forth. The system error is the signal $e_k^{(sys)} = d_k - y_k$, and the symmetric matrix Q is a weighting matrix which assigns different performance objectives to each plant output. To minimize the system error over a trajectory of length K and simultaneously minimize a function of the control effort, we must minimize the function J_k

In this approach the weighs of controller is not adapted in real time. Time is divided into epochs of K time samples in length, and adaptation of the controller weights is performed at the end of each epoch. In this work, a real-time approach was preferred. Therefore, the same trick is employed as used in [46, 31]. The cost matrix J_K is stochastically approximated at each time step as

$$J_K = \left\{ e_{jk}^{(sys)^T} Q e_{jk}^{(sys)^T} + h\left(u_j, u_{j-1}, \cdots, u_{j-r}\right) \right\} \tag{6.2}$$

The gradients of the approximate cost function *Jk* are not the same as the gradients found for the true cost function J_K . Therefore, adaptation is a "noisy" process. In

practice, however, it works well. Continuing, if we let g(.) be the function implemented by the controller C, and f(.) be the function implemented by the plant model $\hat{P}$, we can state without loss of generality

$$u_k = g(u_{k-1}, u_{k-2}, \cdots, u_{k-m}, r_k, r_{k-1}, \cdots r_{k-q}, W) \tag{6.3}$$

$$y_k = f(y_{k-1}, u_{k-2}, \cdots, u_{k-n}, u_k, u_{k-1}, \cdots u_{k-p}) \tag{6.4}$$

where W are the adjustable parameters (weights) of the controller. As is typical for LMS and backpropagation-like learning methods, the controller weights are updated in the direction of the negative gradient of the cost functional

$$\Delta W_k^{\ T} = -\mu \frac{\partial^+}{\partial W} J_k \tag{6.5}$$

$$= -\mu \frac{\partial^+}{\partial W} \left\{ e_{jk}^{\ (sys)T} Q e_{jk}^{\ (sys)T} + h\left(u_j, u_{j-1}, \cdots, u_{j-r}\right) \right\}$$

where μ is the adaptive learning rate. The adaptation algorithm is derived as

$$\frac{\Delta W_k^{\ T}}{\mu} = 2e_k^{\ T} Q \left(\frac{\partial^+ y_k}{\partial W} \right) - \left[\sum_{j=0}^{r} \left(\frac{\partial h(u_k, \cdots, u_{k-r})}{\partial u_{k-j}} \right)^T \frac{\partial^+ u_{k-j}}{\partial W} \right] \tag{6.6}$$

Using Equation 6.3 and 6.4 and the chain rule for ordered derivatives, two further substitutions may be made at this time

$$\frac{\partial^+ u_k}{\partial W} = \frac{\partial u_k}{\partial W} + \sum_{j=0}^{r} \left(\frac{\partial u_k}{\partial u_{k-j}} \right) \left(\frac{\partial^+ u_{k-j}}{\partial W} \right) \tag{6.7}$$

$$\frac{\partial^+ y_k}{\partial W} = \sum_{j=0}^{p} \left(\frac{\partial y_k}{\partial u_{k-j}} \right) \left(\frac{\partial^+ u_{k-j}}{\partial W} \right) + \sum_{j=1}^{n} \left(\frac{\partial y_k}{\partial y_{k-j}} \right) \left(\frac{\partial^+ y_{k-j}}{\partial W} \right) \tag{6.8}$$

Dimension of each term in Eqs. (6.6), (6.7), and (6.8) are (if the plant has N_i inputs and N_o outputs, and the controller has N_W weights)

$$\left[\frac{\partial^+ u_k}{\partial W} \right]_{N_i \times N_W} \left[\frac{\partial^+ y_k}{\partial W} \right]_{N_o \times Nw} \left[\frac{\partial y_k}{\partial y_{k-j}} \right]_{No \times No} \left[\frac{\partial h(.)}{\partial u_{k-j}} \right]_{Ni \times 1} \left[\frac{\partial y_k}{\partial u_{k-j}} \right]_{No \times Ni} \left[\frac{\partial u_k}{\partial u_{k-j}} \right]_{Ni \times Ni}$$

Also

$$[Q]_{N_o \times N_o} [W]_{N_W \times 1} [u_k]_{Ni \times 1} [e_k]_{No \times 1} [y_k]_{No \times 1}$$

6.2.1 Linear FIR Plant Model, Linear FIR Controller

General update rules are obtained up to now. These rules are specialized for linear plants at this section. Any stable linear plant and linear controller may be approximated with arbitrary precision by FIR filters. In this section the plant model and the controller are assumed to be FIR. The input to the controller filter is a tapped-delay-line of q+1 vectors, each of length N_o. This composite vector, at time k, to be

$$R_k \stackrel{\Delta}{=} \left[r_k{}^T, r_{k-1}{}^T, ...r_{k-q}{}^T \right]^T \quad \textbf{(6.9)}$$

It was noted that the plant has N_i inputs. Therefore, the controller will have N_i different linear filters operating on R_k to produce the control signal u_k. Let W_1^T be the first such filter, W_2^T be the second, and so on.

These filters may organized into a matrix W_C such that

$$W_C = [W_1 W_2, ...W_{Ni}]^T \quad \textbf{(6.10)}$$

Then,

$$u_k = W_C R_k \quad \textbf{(6.11)}$$

However, for adapting all the weight values of the controller it is useful to define the column vector of weights to be

$$W = \left[W_1{}^T, W_2{}^T, ...W_{Ni}{}^T \right]^T \quad \textbf{(6.12)}$$

It should be noted that W_C is a matrix while W is a vector; both W and W_C contains identical information. One is rearrangement of the other. We wish to adapt the values in W to optimize J_k.

The input vector of the plant and plant model is;

$$U_k \stackrel{\Delta}{=} \left[u_k{}^T, u_{k-1}{}^T, \ldots . u_{k-p}{}^T \right]^T \tag{6.13}$$

The plant model output can be computed as,

$$y_k = W_{\underset{p}{\wedge}} U_k \tag{6.14}$$

In order to compute Equation (6.6), three quantities $\partial h(\cdot)/\partial u_{k-j}$, $\partial^+ u_k / \partial W$, $\partial^+ y_k / \partial W$,should be calculated

The $\partial h(\cdot)/\partial u_{k-j}$ can be calculated by a used-specified function *h(.)* as below

$$dH_k \stackrel{\Delta}{=} \left[\left(\frac{\partial h(\cdot)}{\partial u_k} \right)^T \left(\frac{\partial h(\cdot)}{\partial u_{k-1}} \right)^T \cdots \left(\frac{\partial h(\cdot)}{\partial u_{k-r}} \right)^T \right]^T \tag{6.15}$$

The second term $\partial^+ u_k / \partial W$ is calculated as below since the controller is assumed to be FIR

$$\frac{\partial^+ u_k}{\partial W} = \frac{\partial u_k}{\partial W} \tag{6.16}$$

From the definition of u_k in Equation (6.3) $\partial u_k / \partial W$ is computed as

$$\frac{\partial u_k}{\partial W} = diag\left\{R_k^T, R_k^T ....R_k^T\right\} = \begin{bmatrix} R_k{}^T & 0 & \cdots & 0 \\ 0 & R_k{}^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_k{}^T \end{bmatrix} \quad \textbf{(6.17)}$$

The last term $\partial^+ y_k / \partial W$, as expanded in Equation (6.8) may be simplified as

$$\frac{\partial^+ y_k}{\partial W} = \sum_{j=0}^{p} \frac{\partial y_k}{\partial u_{k-j}} \frac{\partial^+ u_{k-j}}{\partial W} \quad \textbf{(6.18)}$$

In this summation there are two terms, the first term $\partial y_k / \partial u_{k-j}$ is equal to the $W_{\hat{p}}$ associated with the input ∂u_{k-j} , the second term $\partial^+ u_{k-j} / \partial W$ is calculated for this time step and previous p time steps. If two of them are put it together,

$$dU_k \stackrel{\Delta}{=} \left[\left(\frac{\partial u_k}{\partial W} \right)^T \left(\frac{\partial u_{k-1}}{\partial W} \right)^T \cdots \left(\frac{\partial u_{k-p}}{\partial W} \right)^T \right]^T \quad \textbf{(6.19)}$$

So Equation (6.18) computed as,

$$\frac{\partial^+ y_k}{\partial W} = W_{\hat{p}} dU_k \quad \textbf{(6.19)}$$

If all of them are put it together Equation(6.6) can be specialized for linear plants as

$$\Delta W_k = \mu \left(\left[2\varepsilon_k{}^T \vartheta W_{\hat{p}} - dH_k{}^T \right] dU_k \right)^T \quad \textbf{(6.20)}$$

6.2.2 Nonlinear NARX Plant Model, Nonlinear NARX Controller

The dynamical behaviour of most nonlinear systems may not be approximated by a nonlinear transversal model very well. Whereas for the linear plant we had some freedom to choose the structure of the plant model, here we need to restrict ourselves to a single paradigm. The BPTM algorithm, for a linear plant, was able to compute the impulse response of the plant model, regardless of the structure of the model, and use that to update the controller.

It is assumed that NARX neural network filters are used for both the plant model and the controller. Such filters are capable of controlling any (controllable) nonlinear system with acceptable accuracy. It is desired to compute the weight-update ΔW_k of Equation. (6.6). Linear deviation of the terms are differ in nonlinear case, the terms, Equations (6.7) and (6.8), are repeated here for convenience

$$\frac{\partial^+ u_k}{\partial W} = \frac{\partial u_k}{\partial W} + \sum_{j=0}^{m} \left(\frac{\partial u_k}{\partial u_{k-j}} \right) \left(\frac{\partial^+ u_{k-j}}{\partial W} \right) \tag{6.7}$$

$$\frac{\partial^+ y_k}{\partial W} = \sum_{j=0}^{p} \left(\frac{\partial y_k}{\partial u_{k-j}} \right) \left(\frac{\partial^+ u_{k-j}}{\partial W} \right) + \sum_{j=1}^{n} \left(\frac{\partial y_k}{\partial y_{k-j}} \right) \left(\frac{\partial^+ y_{k-j}}{\partial W} \right) \tag{6.8}$$

The terms which are needed to be computed each iteration are

$$\frac{\partial u_k}{\partial W} \qquad \frac{\partial u_k}{\partial u_{k-j}} \qquad \frac{\partial y_k}{\partial u_{k-j}} \text{ and } \qquad \frac{\partial y_k}{\partial y_{k-j}}$$

The first term, $\partial u_k / \partial W$ has the direct effect of the controller weights on the controller output, the other terms have effect of the inputs of the controller and plant model. All terms are Jacobian matrices and are very simple to calculate for any neural network, using the backpropagation algorithm.

The term of $\partial u_k^+ / \partial W$ Equation (6.7) is computed by determining the values of $\partial u_k / \partial W$ and $\partial u_k / \partial u_{k-j}$. These are found by back-propagating unit vectors $v = \hat{e}_i$ through the controller neural network by using Equations. (4.9) and (4.11).

To compute the term $\partial y_k^+ / \partial W$ of Equation (6.8), it is needed to know $\partial y_k / \partial u_{k-j}$ and $\partial y_k / \partial y_{k-j}$, which are found by back-propagating unit vectors $v = \hat{e}_i$ through the plant-model neural network and using Equation (4.9). A practical implementation is realized to compact the notation into a collection of matrices as before. The definitions of dU_k and dH_k remain unchanged from Equations. (6.19), and (6.15). Furthermore, it is defined

$$dY_k \stackrel{\Delta}{=} \left[\left(\frac{\partial y_{k-1}}{\partial W} \right)^T \left(\frac{\partial y_{k-2}}{\partial W} \right)^T \cdots \left(\frac{\partial^+ y_{k-n}}{\partial W} \right)^T \right]^T \tag{6.21}$$

$$\partial_U Y_k \stackrel{\Delta}{=} \left[\left(\frac{\partial y_k}{\partial u_k} \right)^T \left(\frac{\partial y_k}{\partial u_{k-1}} \right)^T \cdots \left(\frac{\partial y_k}{\partial y_{k-p}} \right)^T \right]^T \tag{6.22}$$

$$\partial_Y Y_k \stackrel{\Delta}{=} \left[\left(\frac{\partial y_k}{\partial y_{k-1}} \right)^T \left(\frac{\partial y_k}{\partial y_{k-2}} \right)^T \cdots \left(\frac{\partial y_k}{\partial y_{k-n}} \right)^T \right]^T \tag{6.23}$$

$$\partial_U U_k \stackrel{\Delta}{=} \left[\left(\frac{\partial u_k}{\partial u_{k-1}} \right)^T \left(\frac{\partial u_k}{\partial u_{k-2}} \right)^T \cdots \left(\frac{\partial u_k}{\partial u_{k-p}} \right)^T \right]^T \tag{6.24}$$

6.3 Simulation Examples

We have seen some analytic results pertaining to constrained control, and an algorithm for adapting a controller to perform constrained control. This final section presents a number of simulation examples to demonstrate the algorithm just developed.

6.3.1 Linear Plant

The first example is for the minimum-phase tank of Section.5.3.1. Equation (5.15) is the difference equation specifying the dynamics of the plant. For all simulations, the input source is as Equation (6.25) which has frequency 0.02 Hz.

$$u_k = 0.05 + 0.12\sin(\frac{2\pi k}{5}) \tag{6.25}$$

When constraints on the control effort were considered, the control signal u_k was restricted to be between 0ºC and 0.1ºC. The controller was a four-tap FIR filter. It appears to be a hard limit on the control signal. The actual equation governing the penalty function is

$$h(u_k) = \begin{cases} (\frac{u_k - 0.1}{0.01})^2, & \text{if } u_k < 0; \\ (\frac{u_k + 0.1}{0.01})^2, & \text{if } u_k > 0.1; \\ 0, & \text{otherwise.} \end{cases} \tag{6.26}$$

Simulations were performed to determine the controller with and without constraints on the control effort, also plant identification is performed during simulations. Figure 6.4 and Figure 6.5 show the tracking performance of the unconstrained and constrained controllers for identical input signals for last 100 time steps. The red line shows the discrete-time output of the plant, governing the plant output. The black line shows the FIR(20,0) filter output and the green line shows the desired response of the plant output. System Error and Model Error for both algorithms, are also plotted with dark blue and light blue lines respectively in whole simulation time in Figure 6.6 and 6.7 for the 200 time steps

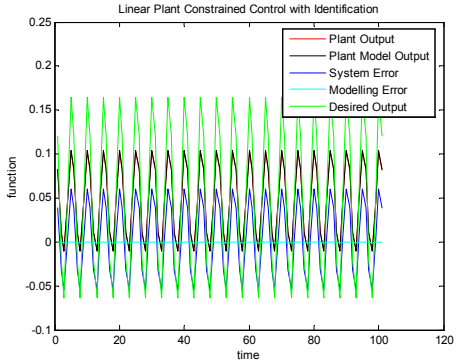

Figure 6.4 : Constrained Control Effort

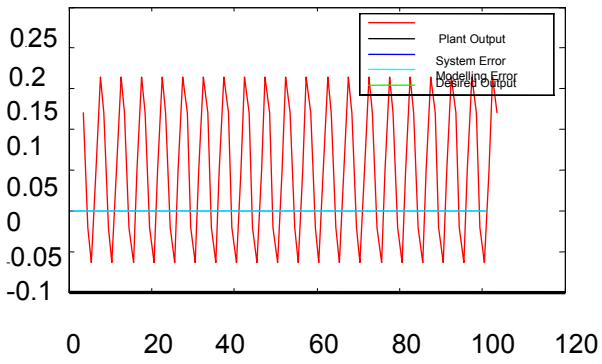

Figure 6.5 : Unconstrained Control Effort

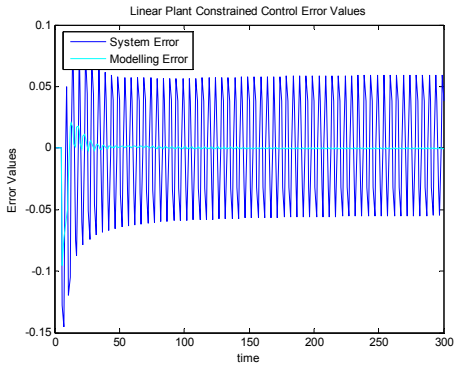

Figure 6.6 : Modelling and System Error Values of Constrained Control

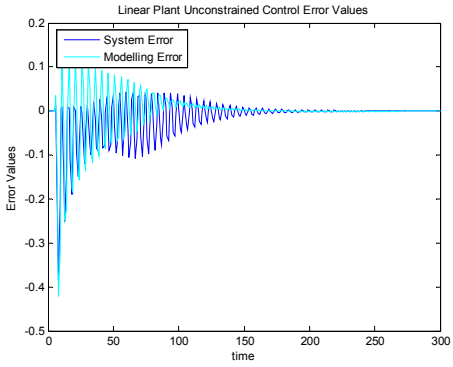

Figure 6.7 : Modelling and System Error Values of Uncon strained Control

6.3.2 Nonlinear Plant

The goal was to control the heading angle of a large ocean going ship (Section. 3.3.1), with constraints on the maximum rudder angle and the rate-of-change of the rudder angle. In this example constraints were built into the dynamics of the ship, and thus no external penalty function was used to adapt the controller to perform constrained control. This method worked very well, and guaranteed that the constraints would be met, regardless of the control input signal.

It is not recommended for linear plants because it implicitly causes the plant dynamics to become nonlinear, and thus a linear plant model is no longer feasible and a linear controller will no longer work well. [31]

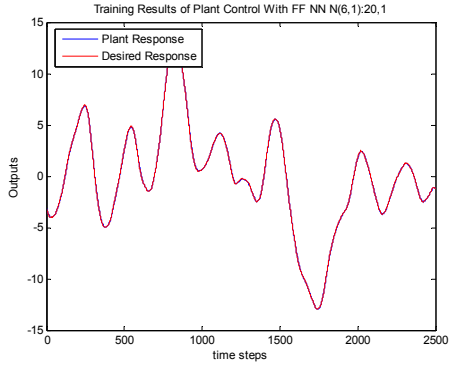

Figure 6.8 Constrained Control of Plant with FF NN

The ship was commanded to track a first-order Markov process which generated the desired heading angle. The Markov process had a pole at z= 0.99 and was fed by i.i.d. uniform random numbers with maximum magnitude 1. A feedforward *N(6,1):20:1* , $(x_k, x_{k-1},.., x_{k-19}, y_{k-1})$ controller was trained to control the ship. The red line is the desired heading angle, and the blue line is the actual heading angle as a function of time. This plant is nonminimum-phase. The meaning of "nonminimum-phase" in the context of nonlinear control is that a stable, causal inverse does not exist. We must use a delay in the reference model in order to provide good control.

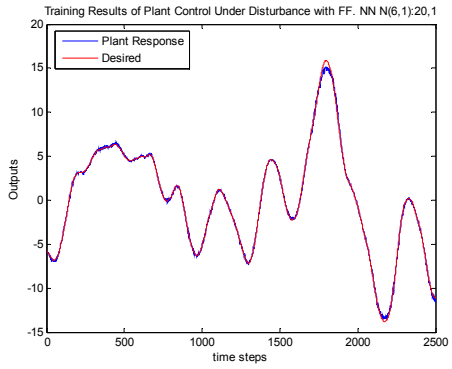

Figure 6.9 Constrained Control of Plant with FF NN

As showned a feedforward controller is trained and a good performance is achieved with this controller. The disturbance effect is not taken into account in this example. The effect of disturbance defined in Section 5.3.2 is added to the system, and the controller is tried to be control the system. As shown in Figure 6.9 controller is not efficiently capable of controlling the plant. The disturbance effect is increased, in third example, the maximum magnitude of the uniform numbers are changed to 6. As shown in Figure 6.10, the neural network is not capable to control the plant. The disturbance should be cancelled, which will describe in Chapter 7.

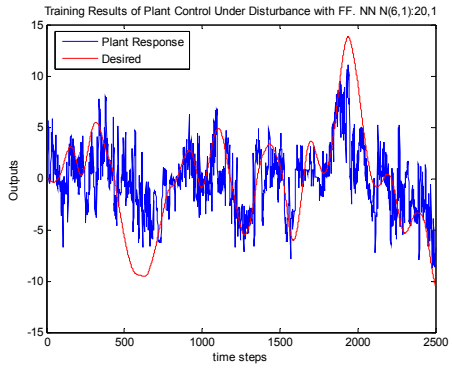

Figure 6.10: Constrained Control of Plant under Disturbance Effect

6.3.3 Summary

This chapter has two divisions. The first derives an algorithm to perform constrained control in the adaptive inverse control paradigm; and the second demonstrates this

algorithm with examples. It was shown analytically, and verified with simulations, that precision of control comes at the cost of high control effort. If very precise control is desired, the actuator signals are very large. Problems with large control effort include (1) The actuator may not be able to respond to the control command due to its physical design, thus causing degradation in the control which is not accounted for in the design; and (2) The actuator or the system being controlled may be damaged by excessive control effort. Since this is a significant problem, a method is devised to perform adaptive inverse control with constraints on the control effort. The controller is adapted such that the mean-squared system error is minimized under the constraint. Simulations have shown that this works very well.

If the plant is nonminimum-phase, its inverse does not exist. However, if a delay in the control action is acceptable, then a "delayed inverse" does exist, and very precise control can be performed. Choosing the correct delay is a significant design issue. The overall conclusion is that very good feedforward control may be achieved, when there is no disturbance, if a disturbance effect is added to the system. The constrained control may not capable to control the system, so a disturbance canceller should be introduced to achieve a good performance.

7 DISTURBANCE CANCELLING

7.1 Introduction

A disturbance-free dynamical system may be controlled with a feedforward adaptive controller. The plant output tracks the desired output as closely as possible in a mean-squared-error sense. It remains to determine what can be done to mitigate plant disturbance should it be present.

Analysis is done on an alternate technique, which is optimal for linear systems, but slightly sub-optimal for nonlinear systems. Method for adapting the disturbance canceller is introduced, and simulations are presented.

7.1.1 Structure of the Disturbance Canceller

This section of analysis concerns itself with the mathematical function that the disturbance cancelling circuit must compute. A little careful thought in this direction leads to a great deal of insight, and some surprising conclusions are reached. First, we must consider some issues of timing which arise since we are performing discrete-time control. Then investigate the function of the disturbance canceller.

Issues of Timing

A discrete-time digital controller is implemented for the type of adaptive inverse control examined in this thesis. Due to the discrete-time nature of the control scheme, a subtle issue arises which is not present in continuous-time control systems. Consider the timing diagram in Figure. 7.1

Suppose sampling rate of the system is equal to $1/T$ samples per second, then the discrete-time/continuous-time correspondence is: $t=kT$ seconds, where t is the physical time in seconds and k is the discrete-time index. The k^{th} command to the plant u_k takes place at $t=kT$, and the k^{th} plant output y_k is sampled in the neighbourhood of $t=kT$ seconds. More precisely, supposing that the plant might not be strictly proper, y_k must be measured at time $t = (kT)^+$ seconds. [31]

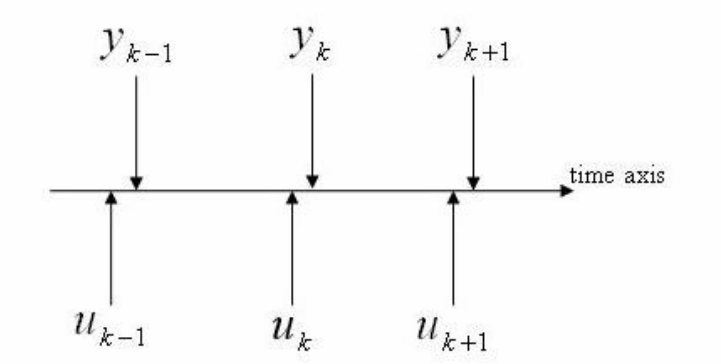

Figure 7.1 : Input Output Timing of a Discrete-Time Control System

The command input to the plant u_k takes finite time to compute, so r_k must be supplied at $t = (kT)^-$ seconds to be able to compute the disturbance-cancelling signal u_k in time. We must also be able to compute $\tilde{u}_k$ slightly before time $t=kT$. This brings us to the important point: To cancel disturbance, there must have an estimate of w_k

A useful way of looking the overall system is drawn in Figure 7.2. Using operator notation, we restate that the control goal is for X to produce an output so that $y_k = M(\vec{r}_k)$. We can express y_k as

$$y_k = w_k + P\left(C(\vec{r}_k) + X(\hat{\vec{w}}_{k-1}, \vec{u}_k)\right) \qquad \textbf{(7.1)}$$

The dashed line in the Figure 7.2 shows that X takes the optional signal u_k. This signal is used when controlling nonlinear plants as it allows the disturbance canceller some knowledge of the plant state.

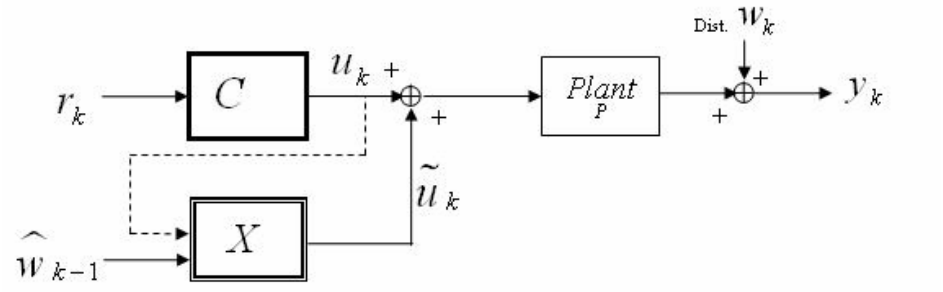

Figure 7.2 : A Useful way of Looking at Feedforward System Dynamics

Next $y_k = M(\vec{r_k})$ is substituted and the desired response of X is rearranged and solved as

$$X^{(opt)}(\vec{\hat{w}}_{k-1}, \vec{u}_k) = P^{-1}\left(M(\vec{r}) - \vec{\hat{w}}_k\right) - C(\vec{r})$$

$$= P^{-1}\left(P(\vec{u}_k) - \vec{\hat{w}}_k\right) - \vec{u}_k$$

The controller has adapted until $P(\vec{u_k}) = M(\vec{r_k})$. The function of X is a deterministic combination of known (by adaptation) elements P and P^{-1}, but also of the unknown signal w_k . Because of the inherent delay in discrete-time systems, we only know w_{k-1} at any time, so w_k must be estimated from previous samples of w_{k-1}, w_{k-2},.. Assuming that the adaptive plant model is perfect and that the controller has been adapted to convergence, the internal structure of X is then shown in Figure 7.3. The w_k signal is computed by estimating its value from previous samples of $\hat{w}_k$. These are combined and passed through the plant inverse to compute the desired signal $\tilde{u}_k$

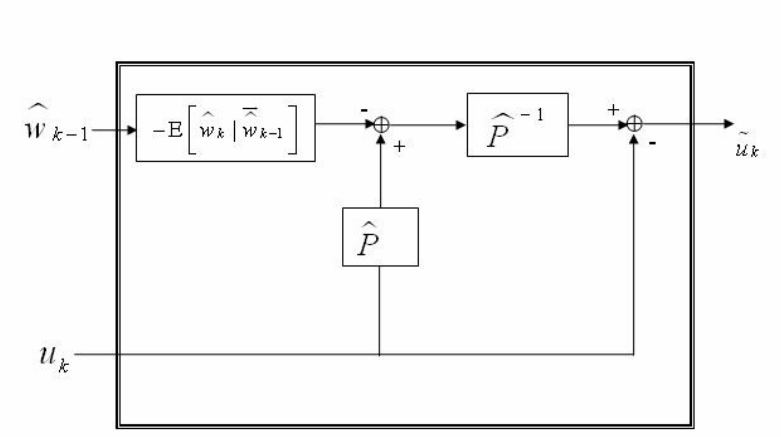

Figure 7.3 : Internal Structure of X

The disturbance canceller contains two parts. The first part is an estimator part which depends on the dynamics of the disturbance source. The second part is the canceller part which depends on the dynamics of the plant. The diagram simplifies for a linear plant since some of the circuitry cancels. Figure 7.4 shows the structure of X for a linear plant.

One very important point to notice is that the disturbance canceller still depends on both the disturbance dynamics and the plant dynamics. If the process generating the

disturbance is not generated by filtering white noise using a linear filter, then the estimator required will in general be a nonlinear function.

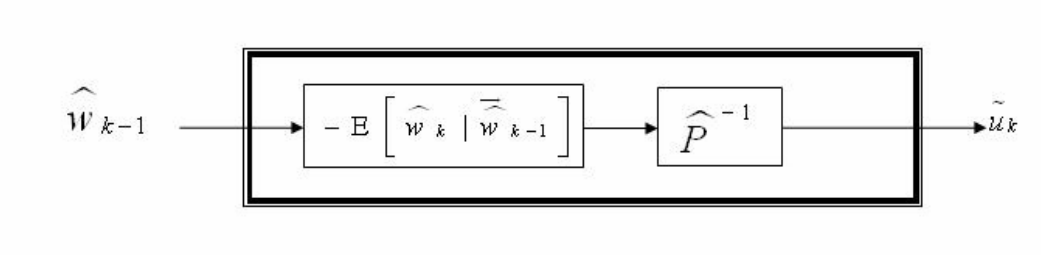

Figure 7.4 : Internal Structure of X if the Plant is Linear

7.2 Synthesis of the Disturbance Canceller via the BPTM Algorithm

How a disturbance cancelling filter can be inserted into the control-system design was discussed in first part. This, the second part of our discussion on disturbance cancelling, describes a method to adapt the disturbance cancelling filter.

7.2.1 Training X In-Place

The method works on the following basis. As the system error is composed of three parts:

- The first part of the system error is dependent on the input command vector $\vec{r}_k$ in C. This part of the system error is reduced by adapting C.

- The second part of the system error is dependent on the estimated disturbance vector $\hat{w}_k$ in X. This part of the system error is reduced by adapting X.

- The third part is the minimum-mean-squared-error. This part of the system error is independent of both the input command vector in C and the estimate disturbance vector in X. It is either irreducible (if the system dynamics prohibit improvement), or may be reduced by making the tapped delay lines at the input to X or C larger. In any case, adaptation of the weights in X or C will not reduce the minimum-mean-squared-error.

- The fourth possible part of the system error is the part which is dependent on both the input command vector and the disturbance vector. However, by assumption, r_k and w_k are independent, so this part of the system error is zero.

The system error is reduced by using the BPTM algorithm adapting C, as discussed in Chapter 5; hence the component of the system error dependent on the input r_k is reduced. Since the disturbance and minimum-mean-squared-error are independent of r_k , their presence will not bias the solution of C. The controller will learn to control the feedforward dynamics of the system, but not to cancel disturbance.

If BPTM algorithm is used and system error backpropagated through the plant model, it is used to adapt *X* as well, the disturbance canceller would learn to reduce the component of the system error dependent on the estimated disturbance signal. The component of the system error due to unconverged C and minimum-mean-squared-error will not bias the disturbance canceller. This method is illustrated in Figure 7.5 where a complete integrated nonlinear control system is drawn. The plant model is adapted directly, as before. The controller is adapted by backpropagating the system error through the plant model and using the BPTM algorithm of Chapter 6.

The disturbance canceller is adapted by backpropagating the system error through the copy of the plant model and using the BPTM algorithm as well. The BPTM algorithm serves two functions: it is able to adapt both C and X. Using BPTM to adapt X works well for either linear or nonlinear systems.

7.3 Simulation Examples

Some analytical results were discussed, related to disturbance cancelling. It is time to present some simulation results for the plants of Chapter 5 in order to verify the analytical results of this chapter and to demonstrate the viability of the disturbance cancelling method.

7.3.1 Linear Plant

The first example is the minimum-phase tank of Section 5.3.1. We have already seen (cf. Chapter 6) simulation results showing that this plant may be very effectively controlled in a feedforward sense using either a linear FIR or nonlinear NARX filter as a controller.

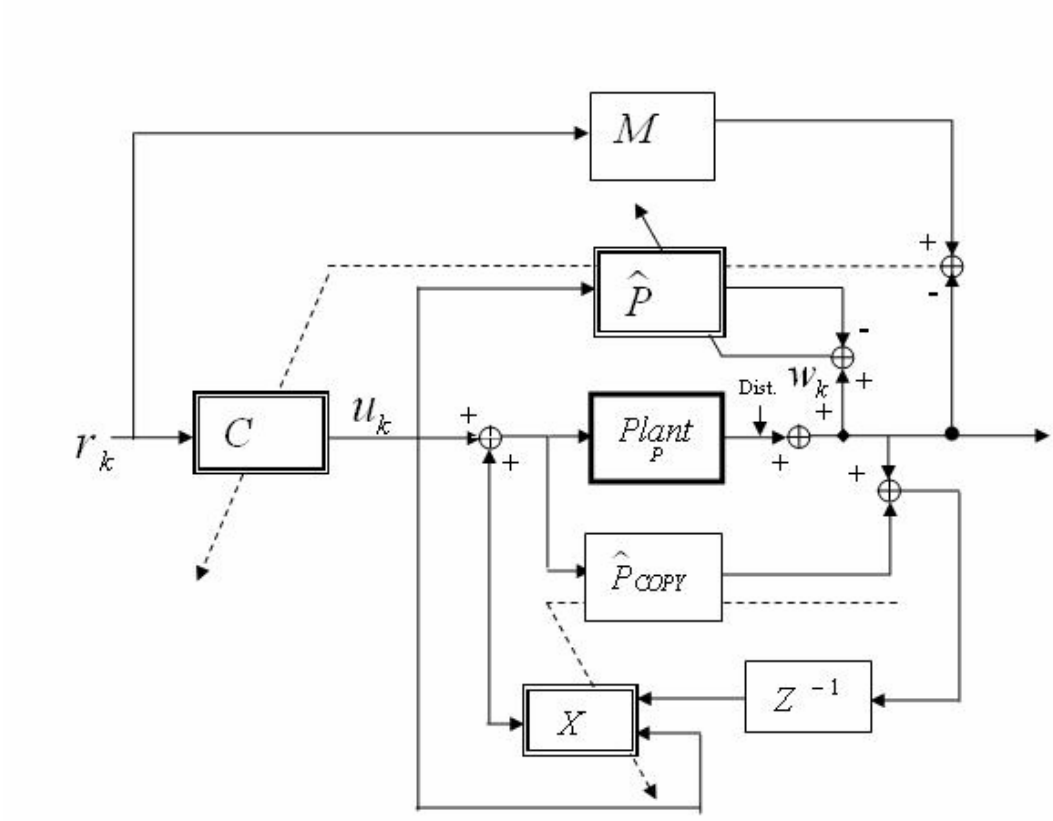

Figure 7.5 : Disturbace Cancelling via X-In Place

Controllers were adapted to perform either unconstrained control, or control where the control effort was limited to be between 0ºC and 0.1ºC. We now look at the problem of disturbance cancelling. The disturbance experienced by this plant was specified to be fluctuation in the temperature of the hot source that the disturbance is statistically dependent on the control signal. However, it can be shown that the disturbance is not correlated with u_k, and so the plant model will adapt to an unbiased solution, despite the disturbance.

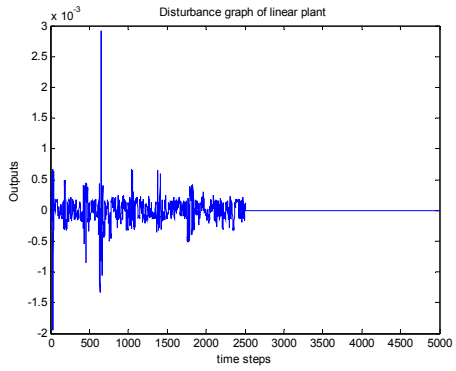

Figure 7.6 Disturbance Graph of Linear Plant

Another interesting feature of this disturbance is that it is a nonlinear random process. That is, the least mean-squared-error predictor of the current disturbance value given all previous disturbance values is a nonlinear function. We will find that the disturbance cancelling filter X, which gives good performance is therefore a nonlinear NARX filter.

Simulation was performed to adapt disturbance cancelling filter X . There were no constraints on the control effort. In the plots the system was run for 2500 seconds with the disturbance canceller turned off. Then, the disturbance canceller was turned on and the system was allowed to run for an additional 2500 seconds. The disturbance cancelling filter X was adapted to convergence for first 2500 seconds during simulation was performed.

The disturbance cancelling filter X was a sixty-tap FIR filter whose input was the estimate of the disturbance $\hat{w}_{k-1}$. It adapts to the solution shown in Figure 7.6. The solution comprises a three-step-ahead estimator (due to the inherent two-step delay in the plant, and the one-step delay in the disturbance measurement process) convolved with a delayed plant inverse.

7.3.2 Nonlinear Plant

Simulations were also performed to demonstrate disturbance cancelling for the nonlinear plant of Section 5.3.2. The reader may recall that the nonlinear plant was selected to be a large ocean-going ship for which we would like to control the heading angle. The plant was initially unstable, and was stabilized using feedback. Simple unity feedback was used to stabilize the ship. This feedback has two effects. Most importantly, it stabilizes the system dynamics. Secondly, however, it also performs some disturbance rejection. The feedback works in such a way that the input command is modified to cancel any error in the output, and this error can include disturbance. Therefore, we find that the results for disturbance cancelling for the nonlinear plants are not as spectacular as the results for the linear plants. The reason for this is; the system error with disturbance and without disturbance are not very different. The disturbance canceller, $NN_{(60,\ 1),60,1,\ ,\ (}x_k, x_{k-1},.., x_{k-59},y_{k-1)}$ improves upon the system error with disturbance, but the difference is so small that the disturbance canceller may be considered unnecessary for these plants. The graph of disturbance is showned on Figure 7.9. The system was run for 2500 seconds with the disturbance canceller turned off. Then, the disturbance canceller was turned on and the system was allowed to run for an additional 2500 seconds. The disturbance

cancelling filter X was adapted to convergence for first 2500 seconds during simulation was performed.

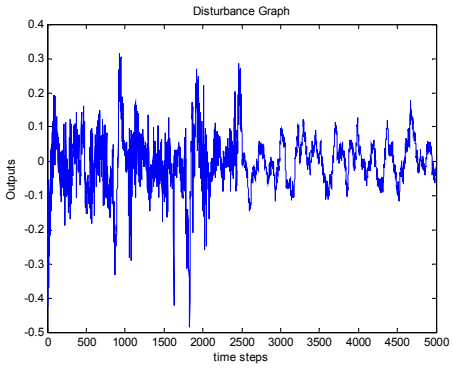

Figure 7-9: The Graph of Disturbance

7.3.3 Summary

This chapter discusses disturbance cancelling for linear and nonlinear, plants. It is organized into two main divisions. The first division presents method to adapt a disturbance canceller; and the third division presents results to verify the analysis and the disturbance cancelling algorithm.

The method works for linear and nonlinear plants, and uses the BPTM algorithm developed in Chapter 4 to adapt the disturbance canceller weights. The second division of the chapter presented simulations to verify the analytical results and the disturbance cancelling algorithms. Simulations were performed to test disturbance cancelling for all of the plants introduced in Chapter 5, with the conclusion that better performance was always obtained using the disturbance canceller. The overall conclusion is that extremely good disturbance cancelling may be achieved.

8 SUMMARY

The problem of controlling a plant may be separated into three separate tasks: stabilization of the plant dynamics; control of plant dynamics; and control of plant disturbance. Conventional control techniques treat all three problems simultaneously. Compromises are necessary to achieve good solutions.

Adaptive inverse control is a method to achieve two tasks separately. First, the plant is stabilized, using conventional methods, secondly, the plant is controlled using a feedforward controller; thirdly, a disturbance canceller is used to reject plant disturbances. Adaptive filters are used as controller and disturbance canceller.

8.1 Constrained Adaptive Feedforward Control

It is assumed that the plant is stable. If it is not stable, it must first be stabilized using conventional feedback. Adaptive inverse control is used to control the stabilized plant.

Next task is to make an adaptive plant model. This process was briefly outlined in Chapter 5. It is assumed that the adaptive plant modelling task continues while the plant is operating, so that any time-variations in the plant dynamics are learned, and so that the controller learns to control the plant as it varies. Thirdly, it is needed to train a feedforward controller for the plant. This task is well understood for SISO linear plants [45] and has been studied for nonlinear plants [2]. In this thesis, the aim is to satisfy constraints on the control effort. Precision of control comes at the cost of high control effort. If very precise control is desired, the actuator signals are very important. Problems with control effort include (1) The actuator do not respond to the control command due to its physical design and (2) The actuator or the system being controlled is damaged by excessive control effort. Since this is a significant problem, a method is devised to perform adaptive inverse control with constraints on the control effort.

A gradient-descent based algorithm was used to update the weights of the controller. The algorithm allows separate implementation of the adaptive controller and plant

model; only local information is needed for the weight update. Very general user-specified constraints on the control effort may be satisfied. Simulation results show that very good performance may be achieved.

If the plant is nonminimum-phase, its inverse does not exist however a “delayed inverse” does exist..

8.2 Disturbance Cancelling

The plant output will track the desired output if there is no disturbance. If there is disturbance, then the plant output will track a signal which is equal to the desired output plus the disturbance. For this reason, a disturbance rejection method is required.

Instead of closing the loop directly, an adaptive filter is trained to perform disturbance cancelling. This method does not bias the plant model if the plant is linear, but does bias the model somewhat if the plant is nonlinear.

The function performed by this disturbance-cancelling filter has two parts. The first part estimates the current disturbance value given past disturbances; the second computes a signal to cancel the disturbance.

The algorithm developed to adapt a feedforward controller also used to adapt the disturbance canceller. This is a great boon to the system designer—only one algorithm needs to be coded! Simulation results show that disturbance cancelling works very well for linear, and good for nonlinear, plants.

BIBLIOGRAPHY

[1] **Astrom, K.J. and Wittenmark, B.,** 1995 Adaptive Control. *Addison-Wesley, Reading*, MA, second edition.

[2] **Bilello, M.** 1996 Nonlinear Adaptive Inverse Control. *PhD thesis*, Stanford University, Stanford, CA.

[3] **Boyd, S.P.,** 1991 Linear controller design: Limits of performance. *Prentice Hall*, Englewood Cliffs, NJ.

[4] **Carbonell, D.,** 1996 Neural Networks Based Nonlinear Adaptive Inverse Control Algorithms. *Thesis for the Engineer degree,* Stanford University, Stanford, CA.

[5] **Economou, C.G. and Morari, M.,** 1986 Internal model control. 5. Extension to nonlinear systems. *Industrial and Engineering Chemistry Process Design and Development,* 25(2):403–11.

[6] **Economou, C.G. and Morari, M.** 1986 Internal model control. 6. Multiloop design. *Industrial and Engineering Chemistry Process Design and Development*, 25(2):411–19

[7] **Franklin, G. F., Powell, J.D. and Emami-Naeini, A.,** 1994 Feedback Control of Dynamic Systems. *Addison-Wesley, Reading,* MA, third edition.

[8] **Franklin, G. F., Powell, J.D. and Workman M. L.,** 1990. Digital Control of Dynamic Systems. *Addison-Wesley, Reading*, MA, second edition.

[9] **Garcia, C. E. and Morari. M.** Internal model control. 1982. 1. A unifying review and some new results. *Industrial and Engineering Chemistry Process Design and Development*, 21(2):308–23.

[10] **Garcia, C.E. and Morari, M.** Internal model control. 1985**.** 2. Design procedure for multivariable systems. *Industrial and Engineering Chemistry Process Design and Development,* 24(2):472– 84.

[11] **Garcia, C.E. and Morari, M.,** Internal model control.1985 3. Multivariable control law computation and tuning guidelines. *Industrial and Engineering Chemistry Process Design and Development,* 24(2):484–94.

[12] **Gazit, R.** Neural control of a multi-link robot arm. Project report for Stanford University class "EE373A,B", June 1994.

[13] **Gersho, A. And Gray, R.M.** 1992. Vector Quantization and Signal Compression. Kluwer *Academic Publishers*, Boston, MA.

[14] **Grace, A., Laub, A.J., Little, J.N. and Thompson**, **C. M.,** 1992.Control System Toolbox for use with MATLAB. *The Math Works Inc*, Natick, MA.

[15] **Gren, M.,**1995. Linear Robust Control. *Prentice Hall,* Englewood Cliffs, NJ.

[16] **Hassibi, B., Sayed, A.H. and Kailath, T.,** 1996. H1 optimality of the LMS algorithm. *IEEE Transactions on Signal Processing,* 44(2):267–80.

[17] **Haykin, H.,** 1996. Adaptive Filter Theory. *Prentice Hall,* Upper Saddle River, NJ, third edition.

[18] **Hizal, A.,** 1999. Improved Adaptive Model Control*, ARI,*51, 181-190.

[19] **Kailath, T.,** 1980. Linear Systems. *Prentice Hall,* Englewood Cliffs, NJ.

[20] **Kaminsky, F.C. Kirchhoff R.H., Syu C.Y., and Manwell J.F.,** 1991.A comparison of alternative approaches for the synthetic generation of a wind speed time series. *Transactions of the American Society of Mechanical Engineers. Journal of Solar Energy Engineering*, 113(4):280–89.

[21] **Kolmogorov, A.N.,** 1957. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. Dokl. Akad. Nauk USSR, 114:953–56.

[22] **Levin, A.U. and Narendra, K.S.,** 1993. Control of nonlinear dynamical systems using neural networks: Controllability and stabilization. *IEEE Transactions on Neural Networks,* 4(2):192– 206.

[23] **Liu, M.C.,** 1998. Statistical Analysis of Quantization—Extended from Widrow's Quantization Theory. *PhD thesis,* Stanford University, Stanford, CA.

[24] **Ljung, L.,** 1987. System Identification: Theorey for the user. *Prentice Hall*, Englewood Cliffs, NJ.

[25] **Murray, R. M. , Li, Z. and Sastry S. S.,** 1994. A Mathematical Introduction to Robotic Manipulation. *CRC Press,* Boca Raton.

[26] **Narendra K. S. and Parthasarathy K.,** 1990. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27.

[27] **Nguyen, D.,** 1991 Applications of Neural Networks in Adaptive Control. PhD thesis, Stanford University, Stanford, CA.

[28] **Oppenheim, A.V. and Schafer, R.W.,** 1989. Discrete-Time Signal Processing. *Prentice Hall,* Englewood Cliffs, NJ.

[29] **Parker, D.B.,** 1982. Learning logic. Technical Report Invention Report S81–64, File 1, Office of Technology Licencing, Stanford University.

[30] **Paulsen, M. J. and Egeland, O.,** 1996. An output feedback tracking controller for ships with nonlinear damping terms. *Modeling, Identification and Control*, 17(2):97–106.

[31] **Plett, G.L.** ,1998.Adaptive inverse control of plants with disturbances. PhD thesis, Stanford University, Stanford, CA.

[32] **Puskorius, G.V. and Feldkamp, L.A.,** Decoupled extended Kalman filter training of feedforward layered networks. In Proceedings of the 1991 International Joint Conference on Neural Networks (San Diego: 1990), volume II, pages 133–141, New York, 1991. IEEE Neural Networks Society.

[33] **Puskorius, G.V. and Feldkamp, L.A.,** Recurrent network training with the decoupled extended Kalman filter algorithm. *In Science of Artificial Neural Networks* (Orlando Florida: 21–24 April 1992), volume 1710, part 2, pages 461–473, New York, 1992. SPIE Proceedings Series.

[34] **Rivera, D. E., Morari, M. and Skogestad, S.,** 1986. Internal model control. 4. PID controller design. *Industrial and Engineering Chemistry Process Design and Development,* 25(1):252–65.

[35] **Rumelhart, D. E., Hinton, G. E. and Williams, R. J.,** 1986. Learning internal representations by error propagation. *Parallel Distributed Processing,* volume 1, chapter 8. The MIT Press, Cambridge, MA.

[36] **Siegelmann, H. T., Horne, B. B. and Giles, C. L.,** 1997. Computational capabilities of recurrent NARX neural networks. *IEEE Transactions on Systems, Man and Cybernetics*—Part B: Cybernetics, 27(2):208–215.

[37] **Touretzky, S.** 1988. Advances in Neural Information Processing Systems (Denver: 1988), pages 133–140, San Mateo, CA.

[38] **Slotine, J. E. and Li, W.,** 1990. Applied Nonlinear Control. *Prentice Hall*, Englewood Cliffs, NJ.

[39] **Sutton, R. and Jess, I. M.,** 1991. A design study of a self-organizing fuzzy autopilot for ship control. *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering,* 205:35–47.

[40] **Overschee, P. and DeMoor, B.**, 1996. Subspace Identification for Linear Systems. *Kluwer Academic Press,* Boston, MA

[41] **Werbos, P.**, 1974. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. *PhD thesis*, Harvard University, Cambridge, MA.

[42] **Werbos, P.**, 1988. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks,* 1(4):339–356.

[43] **Widrow B. and Lehr, M.A..** ,1990. 30 years of adaptive neural networks: Perceptron, Madaline, and backpropagation. *Proceedings of the IEEE,* 78(9):1415–42.

[44] **Widrow, B. and Stearns, S.D.** 1985. Adaptive Signal Processing. *Prentice-Hall*, Englewood Cliffs, NJ.

[45] **Widrow, B. and Walach, E.**, 1996. Adaptive Inverse Control. *Prentice Hall* P T R, Upper Saddle River, NJ.

[46] **Williams, R. J. and Zipser, D.**, 1989. Experimental analysis of the real-time recurrent learning algorithm. *Connection Science*, 1(1):87–111.

[47] **Zhou, K., Doyle, J. C. and Glover K.**, 1996. Robust and Optimal Control. *Prentice Hall,* Englewood Cliffs, NJ.

APPENDIX A

Performance Surface

Linear Combiner is a part of adaptive filter which is illustrated in Figure, is the start point of adaptive filtering.

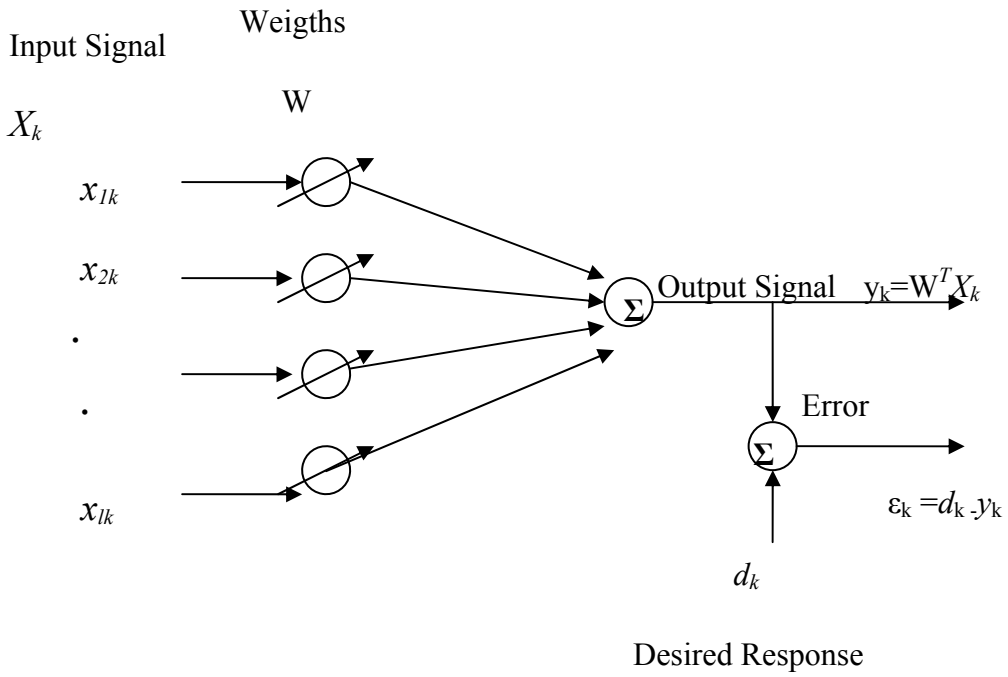

Figure: Adaptive linear combiner

Linear combiner comprises of an input signal vector, weights a summer to add the weighted signals.

The inputs signal.

$$X_k = [x_{1k}, x_{2k}, ...x_{lk}, ...x_{nk}]^T$$

Weight vector

$$W^T = [w_1, w_2, ...w_l, ...w_n]$$

The weights assumed to be fixed then the kth output signal vector

$$y_k = \sum_{l=1}^{n} w_l x_{lk} = W^T X_k = X_k^T W$$

The error between the desired response d_k and k^{th} output signal y_k at k^{th} sampling time is

$$\in_k = d_k - y_k = d_k - W^T X_k = d_k - X_k^T W$$

The square of error is

$$\in^2{}_k = d_k{}^2 - 2d_k X_k^T W + W^T X_k X_k^T W$$

The mean square error between the desired response

$$MSE = \xi \overset{\Delta}{=} E[\in^2{}_k] = E[d^2{}_k] - 2E[d_k X_k^T]W + W^T E[X_k X_k^T]W$$

The cross correlation matrix P is defined as;

$$E[d_k X_k^T] = E\begin{bmatrix} d_k x_{1k} \\ d_k x_{2k} \\ \vdots \\ d_k x_{nk} \end{bmatrix} \overset{\Delta}{=} P$$

The input correlation matrix R is defined as;

$$E[X_k X_k^T] = E\begin{bmatrix} x_{1k}x_{1k} & x_{1k}x_{2k} & \cdots \\ x_{2k}x_{1k} & \ddots & \\ \vdots & & x_{nk}x_{nk} \end{bmatrix} \overset{\Delta}{=} R$$

By writing MSE in terms of P and R

$$\xi = E\left[d^2{}_k\right] - 2P^T W + W^T RW$$

It can be easily seen that that MSE performance function is quadratic function of the weights. It is bowl shaped surface and adaptive process continuously adjusts weights in order to find bottom of the bowl.

The optimal weight vector which minimizes MSE is accomplished by steepest descent algorithm.

The Gradient and Minimum MSE

Gradients of performance surface may be obtained by differentiating the MSE function in Equation 2.9 with respect to the weight vector, which is used Steepest Descent Algorithm.

$$\nabla \stackrel{\Delta}{=} \begin{bmatrix} \dfrac{\partial E\left[\in^2{}_k\right]}{\partial w_1} \\ \vdots \\ \dfrac{\partial E\left[\in^2{}_k\right]}{\partial w_n} \end{bmatrix} = -2P + 2RW$$

Gradient is set to zero in order to find optimal weight vector W^*

$$W^* = R^{-1}P$$

In Equation it is assumed that R is positive definite and R^{-1} exists. This solution is called Wiener solution and the minimum MSE for finite impulse response can be obtained by substituting W^*

$$\xi_{\min} = E\left[d^2{}_k\right] + W^{*T} RW^* - 2P^T W^*$$

$$\xi_{\min} = E\left[d^2{}_k\right] + \left[R^{-1}P\right]^T RR^{-1}P - 2P^T R^{-1}P$$

By simplification

$$\xi_{\min} = E\left[d^2{}_k\right] - P^T W^*$$

Hence we obtain ;

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

The Method of Steepest Descent

The method of steepest descent uses negative gradient while updating the weigth vector.

$$W_{k+1} = W_k + \mu(-\nabla_k)$$

∇_k stands fort he gradient at k^{th} iteration, the scalar parameter μ is called the convergence factor that controls stability and the rate of adaptation. It is necessary to select convergence factor as below for stability.

$$\frac{1}{\lambda_{\max}} > \mu > 0$$

Where $\lambda_{\max}$ is the eigenvalue of R

The Least Mean Squares (LMS) Algorithm

The Least Mean Squres Algorithm is an implementation of Steepent Descent Algorithm, where estimation of gradient is used instead of the actual gradient.

The actual gradient

$$\nabla = \left[\frac{\partial E\left[\in^2_k\right]}{\partial W} \right]$$

The estimation of gradient

$$\hat{\nabla}_k \overset{\Delta}{=} \begin{Bmatrix} \frac{\partial \in^2_k}{\partial w_1} \\ \vdots \\ \frac{\partial \in^2_k}{\partial w_n} \end{Bmatrix} = 2 \in_k \begin{Bmatrix} \frac{\partial \in_k}{\partial w_1} \\ \vdots \\ \frac{\partial \in_k}{\partial w_n} \end{Bmatrix} = -2 \in_k X_k$$

APPENDIX B

Linear Plant Identification with FIR Filter

```
%This function is for Linar Plnat Identification with FIR filter
clc
clear
Ts=1;
H2 = tf(1,[5 1],'inputdelay',1.45)
Hd = c2d(H2,1,'zoh')
n=200;
wn_dm=10;
mu=0.1;
inp_dm=zeros(1,wn_dm);
out_dm=zeros(1,n);
w_dm=zeros(1,wn_dm);
sigma=1e-6;
nLMS=0;
for k=1:1:n
   I(k)=sin(2*pi*k/5);
end
[out_p t]=lsim(Hd,I);
for k=1:n
   inp_dm=[I(k) inp_dm(1:wn_dm-1)];
   out_dm(k)=inp_dm*w_dm';
   e_dm(k)=out_p(k)-out_dm(k);
```

```
    if k<100 & k>=170
        mu=0.1;
    end
    if k>170
        mu=0.1;
    end
    if nLMS==1
        w_dm=w_dm+((inp_dm/(sigma +(inp_dm*inp_dm')))*mu*e_dm(k));
    elseif nLMS==0
        w_dm=w_dm+2*mu*inp_dm*e_dm(k);
    end
end
Pdm=filt(w_dm,[1],Ts);
SE=e_dm.^2;
MSE=(abs(e_dm)/length(e_dm)).^2;
%step(Hd,Pdm)
[out_m t]=lsim(Pdm,I);
figure
plot(out_p(1:n), 'r')
hold on
plot(out_m(1:n), 'b')
plot(e_dm(1:n), 'k')
title('Linear Plant Identification')
ylabel('function')
xlabel('time')
h = legend('Plant Output', 'Filter Output' ,'Error',3);
last_error=e_dm(200)
```

Linear Plant Identification With Feedforward Neural Network

```
% this function is the main funtion of training Feedforward %Neural Network
clear
clc

global M    % train iteration number
global M1   % grafic first model value
global M2   % grafic limit model value
global N    % test iteration number
global N1   % grafic first plant value
global N2   % grafic limit plant value

M=200;
M1=1;
M2=M;
N=200;
N1=1;
N2=N;

[P T]=Do_LinearPlanttrain_Data;
[Wij_0 Wjk_0]=LinearPlantweigthleri_Initialize;
[Wij Wjk A]=LinearPlanttrain_Process(Wij_0, Wjk_0, P, T);
[PV TV]=Do_LinearPlanttest_Data;
LinearPlanttest_Process(Wij, Wjk, PV, TV);

%this function is for preparing the train data
function [P T]=Do_LinearPlanttrain_Data;
```

```
global M


y(1)=0;

y(2)=0;

y(3)=0;

u(2)=0;

u(1)=0;


for k=4:M+6

   u(k)=sin(2*pi*k/5);

   y(k)=LinearPlant(y(k-1), u(k-2),  u(k-3));

end

yp_minus_one =y(6:M+5);

up_minus_two =u(5:M+4);

up_minus_three   =u(4:M+3);

yp          =y(7:M+6);

 P          =[yp_minus_one;  up_minus_two; up_minus_three];

T          =yp;


%This function initializes Neural Networks Weigths

function [Wij Wjk ]=LinearPlantweigthleri_Initialize;


Wij=rand(3,2);

Wjk=rand(2,1);

%This funtion Trains Feedforward Neural Network

function [Wij  Wmn A44]=LinearPlanttrain_Process(Wij,  Wmn, P, T)

global M
```

```
global M1
global M2
mu=0.01;
%mu=0.1;

for k=1:M
    S1=Wij'*P(:,k);
    %A1=tansig(S1);
    S4=Wmn'*A1;
    A4=purelin(S4);
    A44(k)=A4;
    E(k)=T(k)-A4;

    %update weigths
    E4=2*E(k);
    dWmn=mu*A1*E(k)';
    Wmn=Wmn+dWmn;

    %update weigths
    E1=dpurelin(S1,A1).*(Wmn*E4);
    %E1=dtansig(S1,A1).*(Wmn*E4);
    dWij=mu*(P(:,k)*E1');

    Wij=Wij+dWij;
    W11(k)=Wij(1,1);
    Error(k)=(E(k))^2;
end
```

```
LinearPlant_Process_Graphics(T, A44, E, P(1,:),'r', 'k', M1, M2, 'Train Results', 'Plant Output', 'Neural Network Output', 'Error', 'Plant Input')


%this function is for preparing the Test datas
function [PV TV]=Do_LinearPlanttest_Data;
global N
ytt(1)=0;
ytt(2)=0;
ytt(3)=0;
utt(2)=0;
utt(1)=0;

for k=4:N+6
   utt(k)=sin(2*pi*k/5);
   ytt(k)=LinearPlant(ytt(k-1), utt(k-2),  utt(k-3));
end

yt_minus_one =ytt(6:N+5);
ut_minus_two =utt(5:N+4);
ut_minus_three  =utt(4:N+3);
yt         =ytt(7:N+6);
PV          =[yt_minus_one;  ut_minus_two; ut_minus_three];
TV          =yt;

% for testing procedure I use the following codes
function LinearPlanttest_Process(Wij, Wmn, PV, TV)
global N
global N1
```

```
global N2


for k=1:N


   S1t=Wij'*PV(:,k);

   A1t=purelin(S1t);

   %A1t=tansig(S1t);

   S4t=Wmn'*A1t;

   A4t=purelin(S4t);

   A3333(k)=A4t;

   et(k)=TV(k)-A4t;

   Error(k)=(TV(k)-A4t)^2;

end

LinearPlant_Process_Graphics(TV, A3333, et, PV(1,:),'r', 'k', N1, N2,'Test(Validation) Results', 'Plant Output', 'Neural Network Output', 'Error', 'Plant Input')

% to produce actual plant output I use the following codes

function y=LinearPlant( y_1,u_2, u_3);

      y=(0.818*y_1)+(0.1042*u_2)+(0.0771*u_3);

end


% to produce graphs I use the following codes

function LinearPlant_Process_Graphics(T, A,ErrorMSE, u, Tcolor, Acolor, P1, P2, titleG, POut, NNout,Error, Input)

figure

plot (T(P1:P2), Tcolor));

hold on

plot (A(P1:P2), Acolor)

plot(ErrorMSE(P1:P2))
```

```
plot(u(P1:P2), 'g')
title(titleG)
ylabel('outputs')
xlabel('time steps')
h = legend(POut, NNout ,Error,Input, 4);
```

Linear Plant Identification With Externally Recurrent Neural Network

```
% The only difference between Externally Recurrent network anf feedforward neural
Network is updating the weigthis  here is the code

function [Wij  Wmn A44]=LinearPlanttrain_Process(Wij,  Wmn, P, T)
global M
global M1
global M2

mu=0.01;
%mu=0.1;
dyk_dyk1(1)=0;
dWmn=[0;0];
dWij=[0 0;0 0;0 0];

for k=2:M
    S1=Wij'*P(:,k);
    A1=tansig(S1);
    S4=Wmn'*A1;
    A4=purelin(S4);
    A44(k)=A4;
    E(k)=T(k)-A4;
```

```
%Calculating Jacobians
Ek(k)=E(k);
Ej=dtansig(S1,A1).*Wmn*E(k);
Ei=Wij*Ej;
dyk_dyk1(k)=Ei(1);
%Jacobian  W
dWmn=2*mu*E(k).*(A1+dyk_dyk1(k)*dWmn);
Wmn=Wmn+dWmn;
dWij=2*mu*E(k).*(P(:,k)*(Ej/E(k))'+dyk_dyk1(k)*dWij)
Wij=Wij+dWij;
Error(k)=(E(k))^2;


end
```

```
LinearPlant_Process_Graphics(T, A44, E, P(1,:),'r', 'k', M1, M2, 'Train Results', 'Plant Output', 'Neural Network Output', 'Error', 'Plant Input')
```

Nonlinear Plant Identification

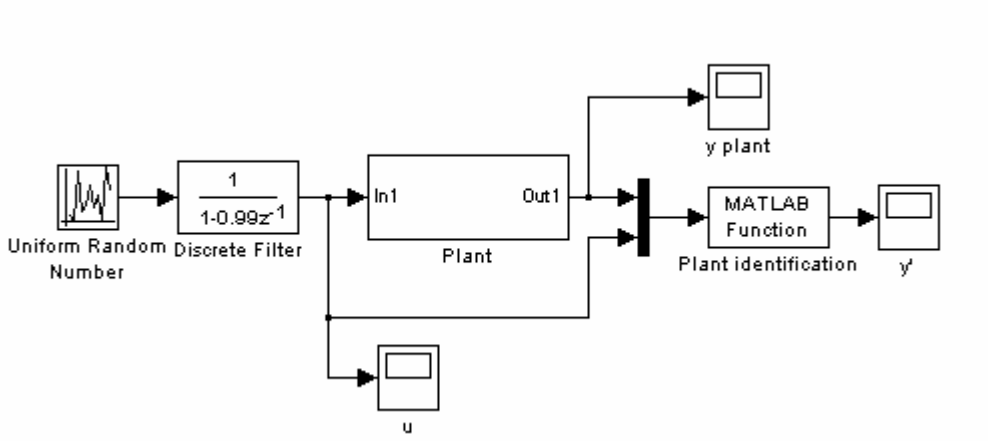

Figure : Simulink Diagram of Nonlinear Plant Identification

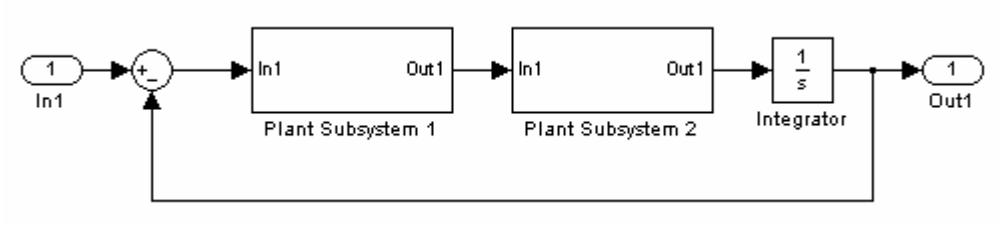

Figure : Simulink Diagrams Subsystem Plant in Nonlinear Plant Identification

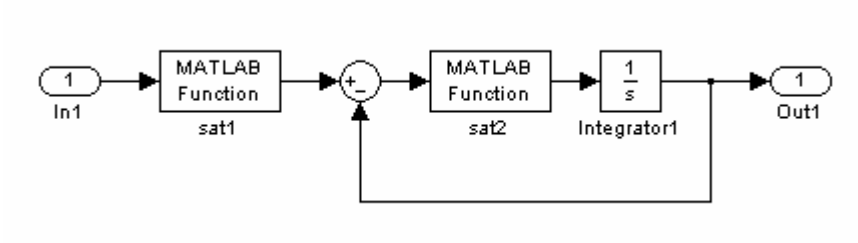

Figure : Simulink Diagrams Plant Subsystem1 in Plant in Nonlinear Plant Identification

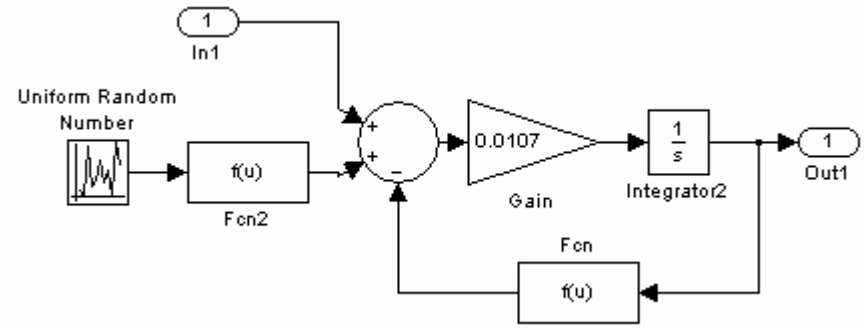

Figure : Simulink Diagrams Plant Subsystem2 in Plant in Nonlinear Plant Identification

```
%This Function is implements the Rudder Angle Limiter for Nonlinear Ship

function sat_Q=sat1(d_Q);

if (d_Q/35)<-1

   sat_Q=-1;

end

if (d_Q/35)>=-1 & (d_Q/35)<=1
```

```
    sat_Q=d_Q;
end
if (d_Q/35)>1
    sat_Q=1;
end

%This Function is implements the Rudder Rate Limiter for Nonlinear Ship

function sat_Q_d=sat2(d_Q_dd);
if (d_Q_dd/3)<-1
    sat_Q_d=-1*2;
end
if (d_Q_dd/3)>=-1 & (d_Q_dd/3)<=1
    sat_Q_d=d_Q_dd;
end
if (d_Q_dd/3)>1
    sat_Q_d=1*2;
end
 %This Function is for identification of  Nonlinear Ship
function [A4]=NlPlant_Ide(u, y,);
global k
global t
global u_
global y_
global Wij
global Wmn
global A44
```

```
t=int64(k*5);


if t<3
    u_(1,:)=0;
    y_(1,:)=0;
    u_(2,:)=0;
    y_(2,:)=0;
    Error=ones(1,127);
   A44=0;
    Wij=rand(5,2);
    Wmn=rand(2,1);
end
if t>=3
    u_(1,:)=0;
    y_(1,:)=0;
    u_(2,:)=0;
    y_(2,:)=0;
    u_(t,:)=u;
    y_(t,:)=y;


    P=[ y_(t-2,:) ;y_(t-1,:); u_(t,:); u_(t-1,:) ;u_(t-2,:)];
    mu=0.01;
    S1=Wij'*P;
    A1=tansig(S1);


    S4=Wmn'*A1;
    A4=purelin(S4);
```

```
A44(t)=A4;
E=y_(t,:)-A4;

%update weigths
E4=2*E;
dWmn=mu*A1*E';
Wmn=Wmn+dWmn;



%update weigths
E1=dtansig(S1,A1).*(Wmn*E4);
dWij=mu*(P*E1');

Wij=Wij+dWij;
W11(t)=Wij(1,1);
Error(t)=(E)^2;

end
k=k+0.2;
plot(Error)
hold on
end
```

Linear Plant Control with Contrained control FIR Filter

```
%this code is used for uncontrained control of linear Plant

clear
```

```
clc
n=3000;
n1=n-100;
Wc1=rand(1,3);
Wc2=rand(1,3);
Wc3=rand(1,3);
Wc4=rand(1,3);
Wc=[Wc1; Wc2; Wc3; Wc4 ];
Wp(6,:)=rand(1,4);

mu_p=0.5;
mu_c=20;
yp(4)=0; yp(5)=0;
yp(3)=0;
yp(2)=0;
yp(1)=0;
r(1)=0.05+(0.12*sin(2*pi*1/5));
r(2)=0.05+(0.12*sin(2*pi*2/5));
r(3)=0.05+(0.12*sin(2*pi*3/5));
r(4)=0.05+(0.12*sin(2*pi*4/5));
r(5)=0.05+(0.12*sin(2*pi*5/5));
yk(3)=0; yk(4)=0; yk(5)=0;
yk(2)=0;
yk(1)=0;
for k=6:n+5
   if k>1000
       mu_p=0.1;
```

```
   mu_c=10;
end
r(k)=0.05+(0.12*sin(2*pi*k/5));

R1k(:,:)=[r(k) r(k-1) r(k-2) r(k-3)];
R2k(:,:)=[r(k-1) r(k-2) r(k-3) r(k-4)];
R3k(:,:)=[r(k-2) r(k-3) r(k-4) r(k-5)];

R1(:,:)=[r(k)   r(k-1) r(k-2)];
R2(:,:)=[r(k-1) r(k-2) r(k-3)];
R3(:,:)=[r(k-2) r(k-3) r(k-4)];
R4(:,:)=[r(k-3) r(k-4) r(k-5)];

Rk=[R1; R2; R3; R4]';
Uk=Wc*Rk;
U=Uk(4,1:4);
yk(k)=U*Wp(k,:)';
yp(k)=0.818*yp(k-1)+0.1042*U(3)+0.0771*U(4);
d(k)= r(k-3);
e(k)=yp(k)-yk(k);
e1(k)=d(k)-yk(k);
dWp=(2*mu_p*e(k)*U);
Wp(k+1,:)= Wp(k,:)+dWp;
dWc1= 2*mu_c*e1(k)*(Wp(k,:).*R1k);
dWc2= 2*mu_c*e1(k)*(Wp(k-1,:).*R2k);
dWc3= 2*mu_c*e1(k)*(Wp(k-2,:).*R3k);
dWc=[dWc1;dWc2;dWc3 ]';
```

```
    Wc=Wc+dWc;


end

figure

plot(yp(n1:n), 'r')

hold on

plot(yk(n1:n),'k')

plot(e1(n1:n),'b')

plot(e(n1:n),'c')

plot(d(n1:n),'g')

title('Linear Plant Control with Identification')

ylabel('function')

xlabel('time')

legend('Plant Output', 'Plant Model Output' ,'System Error','Modelling Error ','Desired Output',5);
```

Linear Plant Control with Contrained control FIR Filter

```
%this code is used for contrained control of linear Plant

clear

clc

n=1000;

n1=n-100;

Wc1=rand(1,3);

Wc2=rand(1,3);

Wc3=rand(1,3);

Wc4=rand(1,3);

Wc=[Wc1; Wc2; Wc3; Wc4 ];

Wp(6,:)=rand(1,4);
```

```
mu_p=0.05;
mu_c=20;
yp(4)=0; yp(5)=0;
yp(3)=0;
yp(2)=0;
yp(1)=0;

r(1)=0.05+(0.12*sin(2*pi*1/5));
r(2)=0.05+(0.12*sin(2*pi*2/5));
r(3)=0.05+(0.12*sin(2*pi*3/5));
r(4)=0.05+(0.12*sin(2*pi*4/5));
r(5)=0.05+(0.12*sin(2*pi*5/5));
yk(4)=0; yk(5)=0;
yk(3)=0;
yk(2)=0;
for k=6:n+5
   if k>1000
      mu_p=0.1;
      mu_c=0.3;
   end
   if k>3000
      mu_p=0.0001;
      mu_c=0.3;
   end
   r(k)=0.05+(0.12*sin(2*pi*k/5));
   if r(k)<0
      hrk(k)=((r(k)-0.1))^2;
```

```
elseif r(k)>0.1
   hrk(k)=((r(k)+0.1))^2;

elseif r(k)<0.1&r(k)>0
   hrk(k)=0;
end

R1k(:,:)=[r(k) r(k-1) r(k-2) r(k-3)];
R2k(:,:)=[r(k-1) r(k-2) r(k-3) r(k-4)];
R3k(:,:)=[r(k-2) r(k-3) r(k-4) r(k-5)];

R1(:,:)=[r(k)   r(k-1) r(k-2)];
R2(:,:)=[r(k-1) r(k-2) r(k-3)];
R3(:,:)=[r(k-2) r(k-3) r(k-4)];
R4(:,:)=[r(k-3) r(k-4) r(k-5)];

Rk=[R1; R2; R3; R4]';
Uk=Wc*Rk;
U=Uk(4,1:4);
yk(k)=U*Wp(k,:)';
yp(k)=0.818*yp(k-1)+0.1042*U(3)+0.0771*U(4);
d(k)= r(k-3);
e(k)=yp(k)-yk(k);
e1(k)=d(k)-yk(k);
```

```
dWp=2*mu_p*e(k)*U;
Wp(k+1,:)= Wp(k,:)+dWp;

if U(1)<0
   dHk1=2*(((U(1)-0.1)/0.01));
end
if U(1)>0.1
   dHk1=2*((U(1)+0.1)/0.01);
end
if U(1)<0.1& U(1)>0
   dHk1=0;
end

if U(2)<0
   dHk2=2*(((U(2)-0.1)/0.01));
end
if U(2)>0.1
   dHk2=2*((U(2)+0.1)/0.01);
end
if U(2)<0.1& U(2)>0
   dHk2=0;
end


if U(3)<0
   dHk3=2*(((U(3)-0.1)/0.01));
end
```

```
if U(3)>0.1
   dHk3=2*((U(3)+0.1)/0.01);
end
if U(3)<0.1& U(3)>0
   dHk3=0;
end

if U(4)<0
   dHk4=2*(((U(4)-0.1)/0.01));
end
if U(4)>0.1
   dHk4=2*((U(4)+0.1)/0.01);
end
if U(4)<=0.1& U(4)>=0
   dHk4=0;
end


dHk=[dHk1 dHk2 dHk3 dHk4];
dUk1=R1k;
dUk2=R2k;
dUk3=R3k;
a1= mu_c*2*e1(k)*Wp(k,:).*R1k;
a2= mu_c*2*e1(k)*Wp(k-1,:).*R2k;
a3= mu_c*2*e1(k)*Wp(k-2,:).*R3k;
a=[a1; a2; a3];
b1=dHk.*dUk1;
```

```
    b2=dHk.*dUk2;
    b3=dHk.*dUk3;
    b=[b1/100; b2/100; b3/100];
    dWc=(a-b)';
    Wc=Wc+dWc;
end
figure
plot(yp(n1:n), 'r')
hold on
plot(yk(n1:n),'k')
plot(e1(n1:n),'b')
plot(e(n1:n),'c')
plot(d(n1:n),'g')
title('Linear Plant Constrained Control with Identification')
ylabel('function')
xlabel('time')
legend('Plant Output', 'Plant Model Output' ,'System Error','Modelling Error ','Desired Output',5);
```

Nonlinear Plant Control

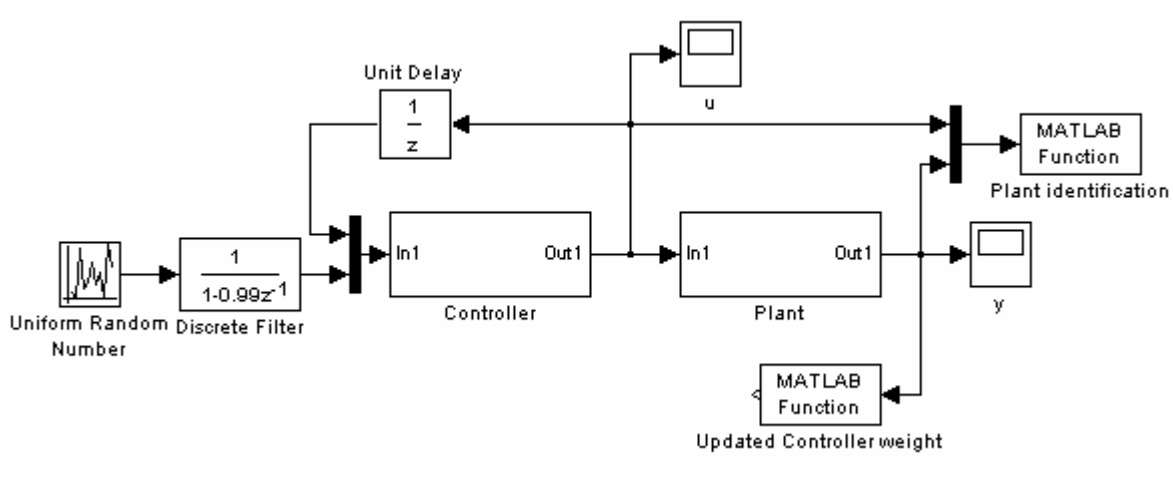

Figure : Simulink Diagram of Nonlinear Plant Control

```
%This Function Produces plant input signal
function [u]=controller(u_1, r);
global k
global t
global u_
global Wc_ij
global Wc_jk
global Ac_1
global Sc_1
global A44
global Rc
t=int32(k*5);

if t<4
    u_(1,:)=0;
    u_(2,:)=0;
    u_(3,:)=0;
    r_(1,:)=r;
    r_(2,:)=r;
    r_(3,:)=r;
    Wc_ij=rand(6,2);
    Wc_jk =rand(2,1);
    Sc_1=[0; 0];
    Ac_1=[0; 0];
    Wc=vertcat(Wc_ij,Wc_jk');
    Rc=[0; 0; 0; 0; 0; 0];
```

```
    u=0;
end

if t>=4
    u_(1,:)=0;    u_(2,:)=0;    u_(3,:)=0;
    r_(1,:)=r;    r_(2,:)=r;    r_(3,:)=r;    r_(t,:)=r;
    Rc=[ r_(t,:); r_(t-1,:); r_(t-2,:); r_(t-3,:); u_(t-1,:) ;u_(t-2,:)];
    Sc_1=(Wc_ij')*(Rc);
    Ac_1=tansig(Sc_1);
    Sc_2=Wc_jk'*Ac_1;
    Ac_2=purelin(Sc_2);
    u=Ac_2;
end

%This Function Updates Controller weigths
function [Wc]=update_C_W(yk);
global Wc_ij
global Wc_jk
global Ac_1
global Sc_1
global Rc
global k
global t
t=int16(k*5);
if t>=4

    mu=0.0002;
```

```
    Q=1;
    dk =Rc(1);
    %Back Propagation
    Ec=dk-yk;
    dWc_jk=zeros(2,1);
    dWc_ij=zeros(6,2);

    %update weigths
    Ec_1=2*Ec*Q;
    dWc_jk=mu*Ac_1*Ec_1';
    Wc_jk=Wc_jk+dWc_jk;

    %update weigths
    Ec_2=dtansig(Sc_1,Ac_1).*(Wc_jk*Ec_1);
    dWc_ij=mu*(Rc*Ec_2');
    Wc_ij=Wc_ij+dWc_ij;
end
```

Linear plant disturbance cancelling

Figure : Simulink Diagram of Nonlinear Plant Control

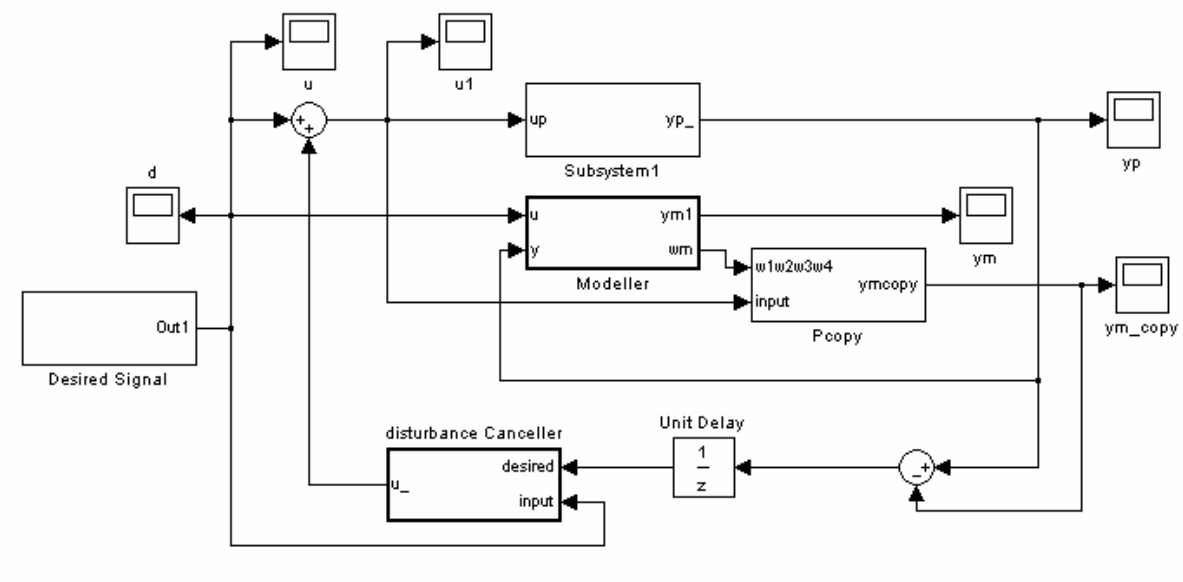

Figure : Simulink Diagram of Linear Plant Disturbance Cancelling

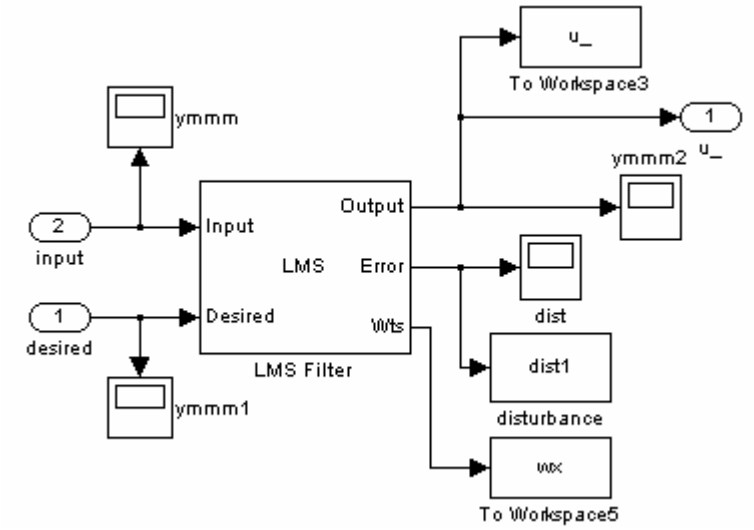

Figure : Simulink Diagram of Linear Plant Disturbance Cancelling of subsystem Disturbance Canceller

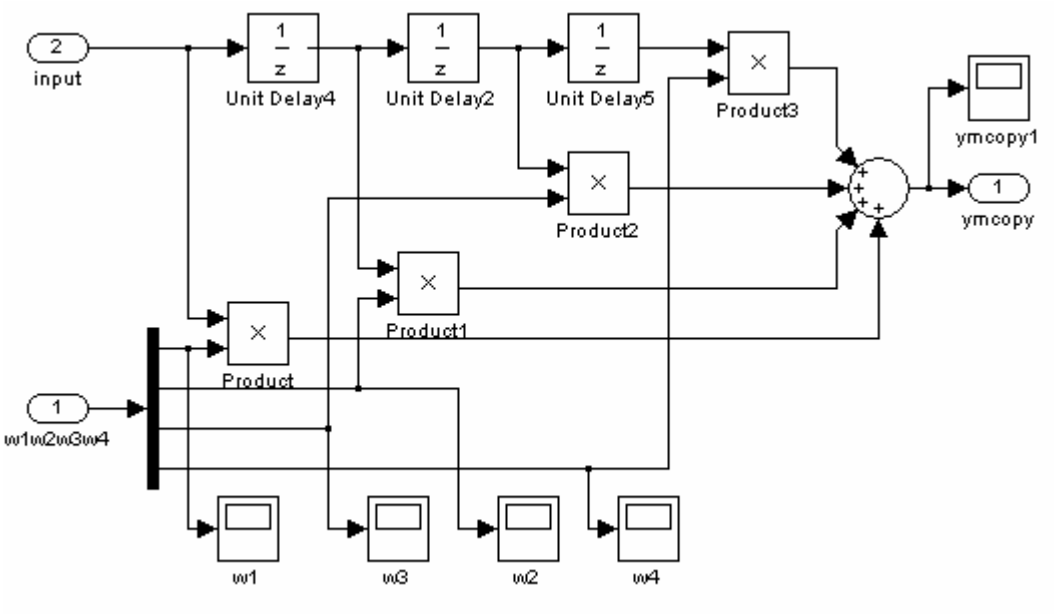

Figure : Simulink Diagram of Linear Plant Disturbance Cancelling of subsystem Pcopy

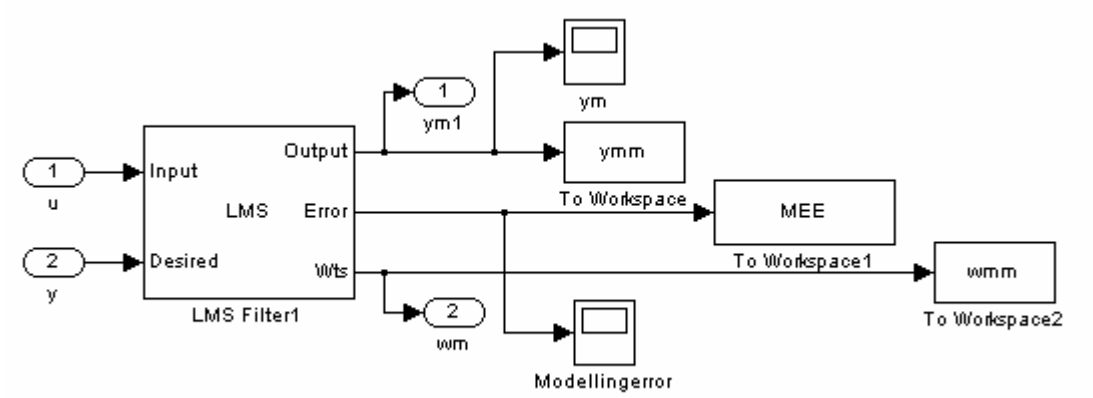

Figure : Simulink Diagram of Linear Plant Disturbance Cancelling of subsystem Modeller

Nonlinear Plant Disturbance Cancelling

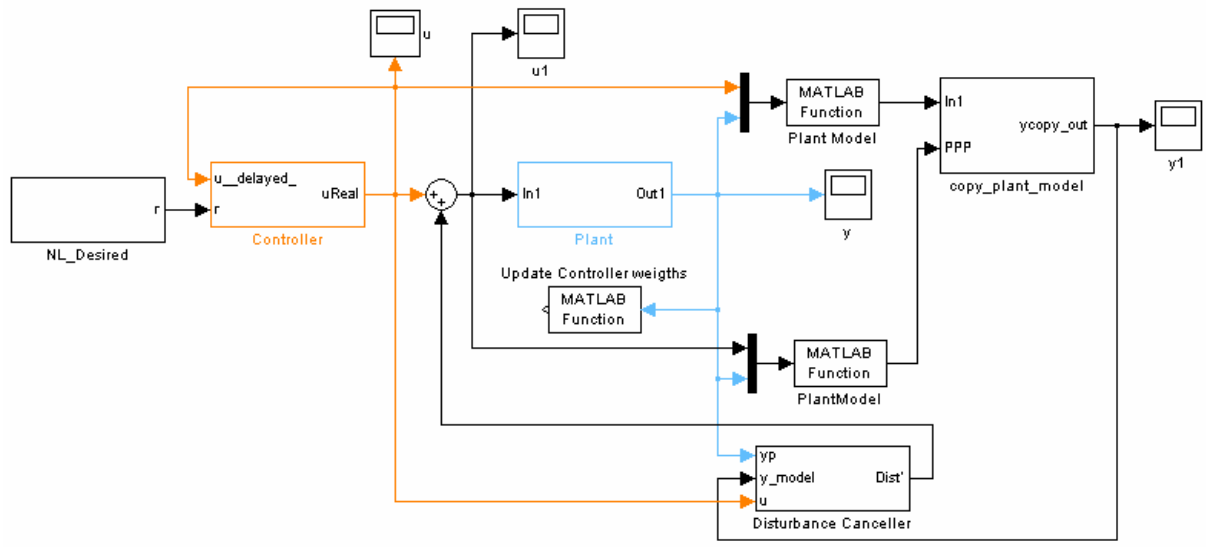

Figure : Simulink Diagram of Nonlinear Plant Disturbance Cancelling

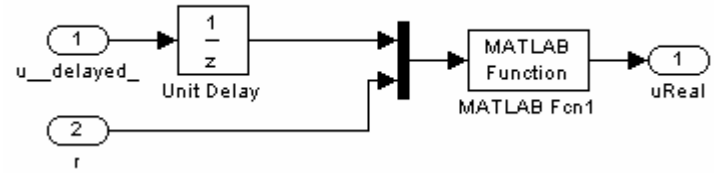

Figure : Simulink Diagram of Nonlinear Plant Disturbance Cancelling of subsystem Controller

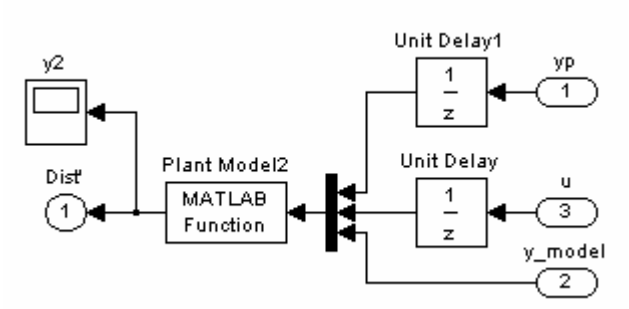

Figure : Simulink Diagram of Linear Plant Disturbance Cancelling of subsystem Disturbance Canceller

```
%This function creates the input values of Copy plant Model

function [PP]=P_al(u, y);

global k

global t

global uu_
```

```
global yy_
global Wij
global Wmn
global A44


if t<3
    uu_(1,:)=0;
    yy_(1,:)=0;
    uu_(2,:)=0;
    yy_(2,:)=0;
    PP=[0; 0; 0; 0];
end
if t>=3
   uu_(1,:)=0;
   yy_(1,:)=0;
   uu_(2,:)=0;
   yy_(2,:)=0;
   uu_(t,:)=u;
   yy_(t,:)=y;

   PP=[ yy_(t-1,:); uu_(t,:); uu_(t-1,:) ;uu_(t-2,:)];
end
end



%This function is the disturbance canceller, that its output is the
%estimate of the disturbance
function [uu]=y_al(y, y1, u);
```

```
global k
global t
global dd
global uuuu
global Wx_ij
global Wx_mn
global Ax44
t=int32(k*5);

if t<3
    u_(1,:)=0;
    uuuu(1,:)=0;
    u_(2,:)=0;
    uuuu(2,:)=0;
    Errorx=1;
    E=1;
    Ax44=0;
    Ax4=0;
    Wx_ij=rand(4,2);
    Wx_mn=rand(2,1);
    Px=[0; 0; 0; 0];
    dd(t,:)=y-y1;

end
if t>=3
    u_(1,:)=0;
    uuuu(1,:)=0;
```

```
    u_(2,:)=0;
    uuuu(2,:)=0;
    u_(t,:)=u;
    uuuu(t,:)=y;
    dd(t,:)=y-y1;
    Px=[ dd(t-1,:); u_(t,:); u_(t-1,:) ;u_(t-2,:)];
    mu_x=0.001;
    Sx1=Wx_ij'*Px;
    Ax1=tansig(Sx1);

    Sx4=Wx_mn'*Ax1;
    Ax4=purelin(Sx4);
    Ax44(t)=Ax4;
    Ex=dd(t,:)-Ax4;
    %update weigths
    Ex4=2*Ex;
    dWx_mn=mu_x*Ax1*Ex';
    Wx_mn=Wx_mn+dWx_mn;
    %update weigths
    Ex1=dtansig(Sx1,Ax1).*(Wx_mn*Ex4);
    dWx_ij=mu_x*(Px*Ex1');
    Wx_ij=Wx_ij+dWx_ij;
    Error_x(t)=(Ex)^2;

end
uu=Ax4;
end
```

CURRICULUM VITAE

Deniz Er was born in Bornova in1980. She graduated from Söke Hilmi Fırat Anatolian High Scholl in 1998, and bachelors degree from Yeditepe University, in System Engineering. She has been a graduate student in Istanbul Technical University since 2004. She is working in Sonar Arge.