# Modeling and Control of Nonlinear Systems Using Wavelet Networks

| | |
|---|---|
| Item Type | Thesis |
| Authors | Hassouneh, Wissam |
| Download date | 2025-07-05 01:14:19 |
| Link to Item | http://hdl.handle.net/11073/116 |

# MODELING AND CONTROL OF NONLINEAR SYSTEMS USING WAVELET NETWORKS

A THESIS IN MECHATRONICS

Presented to the faculty of the American University of Sharjah
School of Engineering
in partial fulfillment of
the requirements for the degree

MASTER OF SCIENCE

by
WISSAM HASSOUNEH
B.S. 2002

Sharjah, UAE

# MODELING AND CONTROL OF NONLINEAR SYSTEMS USING WAVELET NETWORKS

by

Wissam Hassouneh

We approve the thesis of Wissam Hassouneh

Date of signature

Rached Dhaouadi
Associate Professor
Thesis Advisor

Yousef Al-Assaf
Professor
Thesis Advisor

Khaled Assaleh
Associate Professor
Graduate Committee

Wajdi Ahmad
Assistant Professor
Graduate Committee

Mohammad Ameen Al-Jarrah
Director, Mechatronics Engineering
Graduate Program

Yousef Al-Assaf
Dean, School of Engineering

Robert D. Cook
Dean, Graduate Programs and Research

# MODELING AND CONTROL OF NONLINEAR SYSTEMS USING WAVELET NETWORKS

Wissam Hassouneh, Candidate for the Master of Science Degree

American University of Sharjah, 2006

## ABSTRACT

The main objective of this research is to present wavelet networks as a new scheme to model and control nonlinear dynamic systems. The ability to control a system depends on how well it is modeled. Most real-world systems are inherently nonlinear, and conventional PID controllers, having fixed proportional, integral and derivative terms, are not able to deal with time-varying nonlinearities. Such conditions require adaptive controllers that can modify the P, I and D terms to compensate for these nonlinearities. Furthermore, the capability of wavelet networks in the modeling of dynamic nonlinear systems makes them appealing for use in system modeling and control.

This research develops an adaptive PID- Dynamic Wavelet Network controller, comprising a digital PID controller and a proposed new wavelet network scheme called the Dynamic Wavelet Network (DWN), in order to model and control nonlinear systems. The learning strategy for the wavelet network and PID controller is developed based on the gradient descent algorithm. The performance of the proposed controller is demonstrated via extensive numerical simulations.

# CONTENTS

Chapter

Appendix

# LIST OF FIGURES

Figure

TABLES

Table

# ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisors, Dr. Rached Dhaouadi and Dr. Yousef Al-Assaf, for their support, advice and guidance throughout my master thesis at the American University of Sharjah. I thank them for introducing me to the area of adaptive control using wavelet networks, and for fostering an atmosphere of openness and creativity.

I would like to thank Dr. Wajdi Ahmad and Dr. Khaled Assaleh for their valuable time to serve on my graduate committee. Their reviews and constructive criticisms have improved the quality of this work. Their suggestions toward my dissertation are sincerely appreciated.

Finally, this work would not have been accomplished without the loving support and encouragement of my parents. This dissertation is dedicated to them.

# CHAPTER 1

# INTRODUCTION

---

Wavelet networks have received a lot of interest over the past few years in the area of control systems. Wavelet networks are relatively new; however, their applications and growth have come from many areas including wavelet theory, neural networks, statistics, computer science, pattern recognition, and communication, to name a few. Neural networks and wavelet theory, traditionally taught in two different disciplines, were integrated under the common theme of wavelet networks. There are several advantages of combining wavelets and neural networks. Wavelet networks are universal approximators with the ability of achieving faster convergence than neural networks and the capability of dealing with inputs of higher dimensions [7, 10]. Wavelet networks have been used for both static and dynamic modeling [6, 8, 9, 10, 11, 12, 13, 14]. The capabilities of wavelet networks make them a good candidate for nonlinear system identification and control.

Recent progress in wavelet network theory provides us with new tools for modeling, identification and control of complex nonlinear dynamic systems. Intelligent control, with special focus on adaptive control can be used to solve difficult real control problems which are basically nonlinear, noisy and complex. This is because wavelet networks have an inherent ability to learn from input-output functions and approximate an arbitrarily nonlinear function well. A large number of identification and control structures have been proposed on the basis of neural and wavelet networks in recent years [15, 16, 17, 18, 19].

Design and implementation of adaptive control for nonlinear dynamic systems is challenging and extremely difficult. In most cases, developing adaptive control strategies depends on the particular information of the nonlinear structure of the plant that needs to be controlled. Wavelet networks with the ability to deal with nonlinearities can be used to develop an adaptive controller for unknown systems. If the relationship between the input and the output of the unknown nonlinear plant is modeled by an appropriate wavelet network, the obtained model can be used to construct a proper controller. After constructing and training the wavelet network, the wavelet based controller can be implemented on-line. The wavelet network model is updated by measured plant input and output data and then the controller parameters are directly adapted using the updated model.

In this research wavelet networks are used for system identification and the design of an adaptive PID controller for nonlinear dynamic systems. The proposed wavelet network and wavelet-based controller are applied to first order and second order nonlinear systems, and the effect of noise on their performance is analyzed.

## 1.1 Literature Review

The combination of wavelet theory and neural networks has received a lot of attention in recent years for modeling and control of nonlinear systems. This combination led to the development of Wavelet Networks (WN). A wavelet network is basically a feed-forward neural network with one hidden layer of nodes, whose basis functions are drawn from a family of orthonormal wavelets [1]. The popularity of wavelet networks can be attributed to the original work by Pati and Krishnaprasad [2, 3], Zhang and Benveniste [4], Szu, Telfer and Kadambe [5] among others. Zhang and Benveniste [4] found a link between the wavelet decomposition theory and neural networks and presented a basic backpropagation wavelet network learning algorithm. Their wavelet networks preserved the universal approximation properties of the traditional feed-forward neural networks and presented an explicit link between the network coefficients and some appropriate transform. Szu, Telfer and Kadambe [5] applied wavelet networks to speech segmentation and speaker recognition. Zhuang and Baras [7] showed how a wavelet network can be used for the identification of infinite dimensional systems. They developed a systematic approach to find the optimal discrete orthonormal wavelet basis with compact support for spanning the subspaces employed for system identification. This approach was believed to reduce the number of neurons needed to achieve the same performance provided that the wavelets contain useful information of the systems in consideration. The wavelet network also adjusted its wavelet basis according to measurements and was thus called an Adaptive Wavelet Neural Network (AWNN). Marrar, Filho and Vasconcelos [9] introduced a family of polynomial wavelets generated from powers of sigmoids to be used in designing wavelet networks. They showed through examples how this set of wavelets can provide a very good approximation capability, with a fast convergence of the training process. They also observed how only a small number of daughter wavelets were necessary to provide good approximation characteristics. Li and Chen [10] developed a robust learning rule based on Least Trimmed Squares method (LTS) to reduce the sensitivity of wavelet networks to outliers, which are encountered as a result of training data being contaminated by non-Gaussian noise whereby some points fall far outside of the majority of the data. They proposed adaptively adjusting the number of residuals contributing to weight updates and demonstrated through simulation the superiority of the robust learning algorithm over the conventional approach in function approximation from outlying data. Sgarbiy, Coll and Reyneri [11] showed how wavelet networks are a specific case of the more general Weighted Radial Basis Functions Networks (WRBFN). They made a comparison between the two networks for function approximation by using the same initialization and learning rules for both types of networks. They concluded that WRBFN are as good approximators as WN and that the performance of either network depended on how well the chosen mother/activation function "fits" the function to be approximated. The simulations showed that a periodic function is better approximated by a wavelet network with an "oscillating" mother wavelet, while an exponential function is better approximated by a WRBFN with a Gaussian activation function. Oussar and Dreyfus [12] developed an initialization procedure, called initialization by selection, for the parameters of wavelet networks used in function approximation. They used wavelet frames stemming from the discrete wavelet transform to initialize the translation and dilation parameters of wavelet networks trained using gradient based techniques. The procedure consists in generating first a family of wavelets based on the input signals range. The wavelets are next ranked in order of decreasing relevance by the Gram-Schmidt method. Ranking the wavelets involves first estimating the weights of the direct connections of the network by standard least squares (LS) and then deriving a training sequence by subtracting the output of the linear model derived by LS from the initial training sequence. Thereafter, the wavelets (without the direct connections) are ranked and selected using the residuals of the linear model. The authors compared their proposed initialization procedure to the heuristic procedure and simulation results showed that the effect of the random initialization of the weights of the network is much smaller when the wavelet centers and dilations are initialized by selection than when they are initialized heuristically, making WN training more efficient.

In the framework of dynamic modeling of nonlinear systems, wavelet networks have been applied by Zhang to the identification of a robot arm and shown to perform much better than a linear Auto Regressive with eXogenous input model [6], and later to the modeling of a gas turbine system [8]. Yeung and Li [13] used algorithms for the identification of nonlinear multi-input dynamical systems to develop a rainstorm prediction tool. They used a wavelet network for identifying a state space model of the system using radar images as input and the rain gauge measurement as output. By comparing the wavelet network with the traditional neural network, they showed that the WN offered significant improvement in terms of speed, reliability and accuracy. Oussar, Rivals, Personnaz and Dreyfus [14] developed training algorithms for feedback wavelet networks used in the dynamic modeling of single-input-single-output (SISO) processes. The two algorithms were called feedforward wavelet predictor and feedback wavelet predictor. The first was used when it was assumed that noise was acting on the process input, while the second was used when it was assumed that noise was acting on the process output. They also presented an original initialization procedure that takes the locality of the wavelet functions into account. The initialization of the wavelets was based on the domain of the training data, in particular the minimum and maximum values. This procedure guaranteed the dilation and translation parameters were initialized so that the mother wavelet was at the center of the training sequence and extended over the whole input domain. When comparing with classical sigmoidal neural networks, they concluded that the two types of networks can perform equivalently in terms of accuracy and parsimony for low order nonlinear processes, provided proper initialization is done and efficient training algorithms are used.

Adaptive controllers have been implemented using neural networks and, more recently, using wavelet networks to regulate nonlinear dynamic systems. J. Wang, F. Wang, J. Zhang and J. Zhang [15] presented a self-tuning PID controller based on the gradient decent learning algorithm with a neural network. The training of the NN was performed off-line so as to learn the dynamics of the plant, which required long training time. The P, I and D parameters were optimized on-line according to the cost function:

$$J = \sum_{1}^{N} |e_i|$$

Where $e_i$ is the error between setpoint and NN output at the i<sup>th</sup> step and $N$ is the number of steps.

By comparing the self-tuning PID controller to the common PID controller, they concluded that the new controller (i) could compensate for different process and environment uncertainties, (ii) was simple to configure since it did not require a process model, (iii) could track changes of process dynamics on-line, (iv) had all properties of common PID algorithm, and (v) could compensate for large dead time if it existed. The strategy scheme is shown in Figure 1.1.

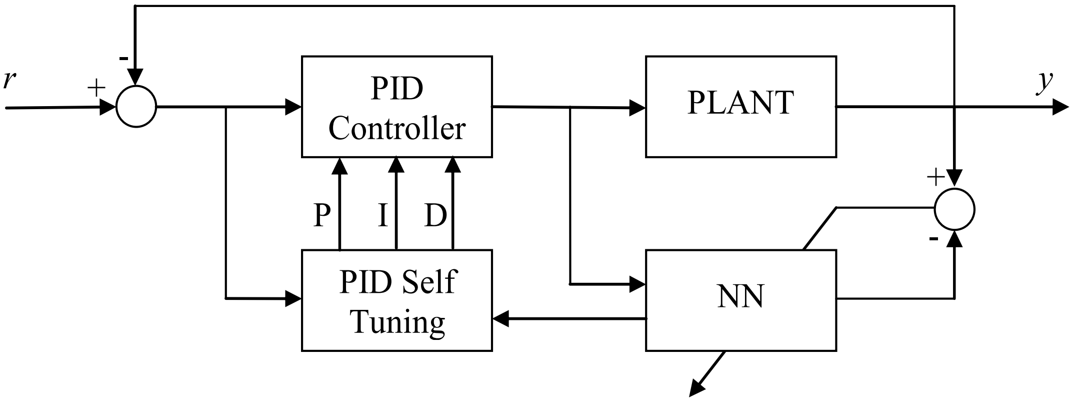


**Figure 1.1** PID Self-Tuning Strategy Scheme

Omatu [16] constructed an adaptive PID controller using two neural networks. The first NN was used to generate the P, I and D terms according to the cost function

$$E = \frac{1}{2}e(n+1)^2 = \frac{1}{2}\left(r(n+1) - y(n+1)\right)^2$$

Where $r(n)$ is the desired reference signal, $y(n)$ is the plant output and $n$ is the sample number.

The second NN was used to emulate the plant. The proposed method was implemented in a SISO temperature control system and in an electric vehicle torque control system. The experimental results showed that the proposed method performed well in both systems. The configuration of the self-tuning neuro-PID controller is shown in Figure 1.2.

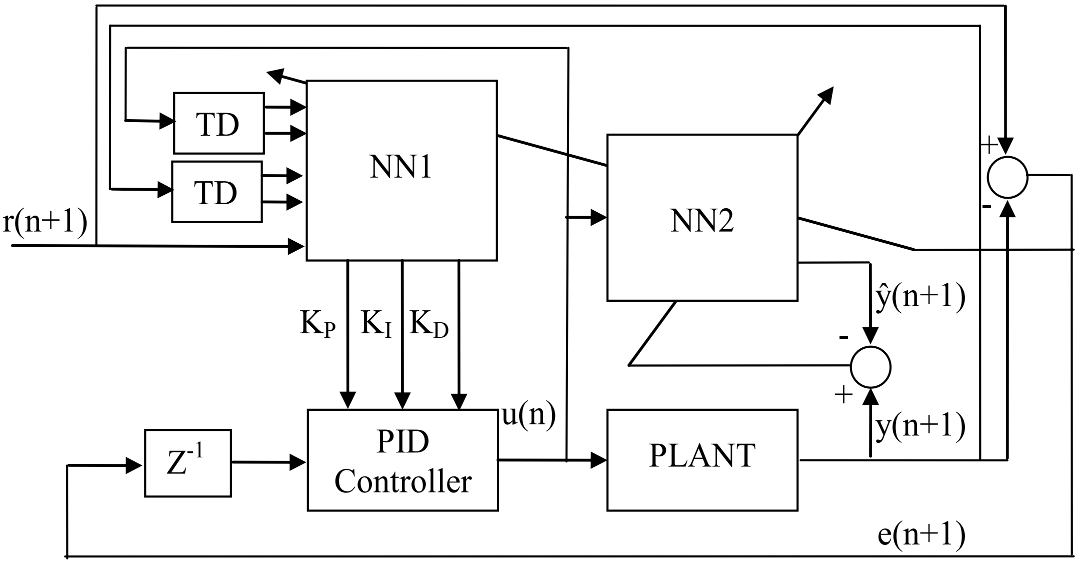


**Figure 1.2** Self-Tuning Neuro-PID controller

Lekutai [17] presented two self-tuning wavelet network controllers. The first was a self-tuning wavelet network controller, shown in Figure 1.3. The controller was constructed with a wavelet network utilizing twenty wavelets and was implemented in the control of a nonlinear dynamic system.

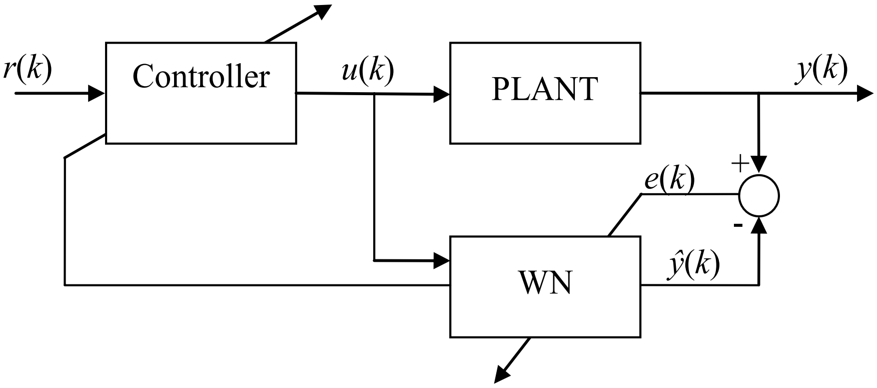


**Figure 1.3** Self-Tuning Wavelet Network Controller

The second was an adaptive PID controller shown in Figure 1.4. The controller was constructed using the same sized wavelet network and was implemented in the control of the same nonlinear dynamic system.

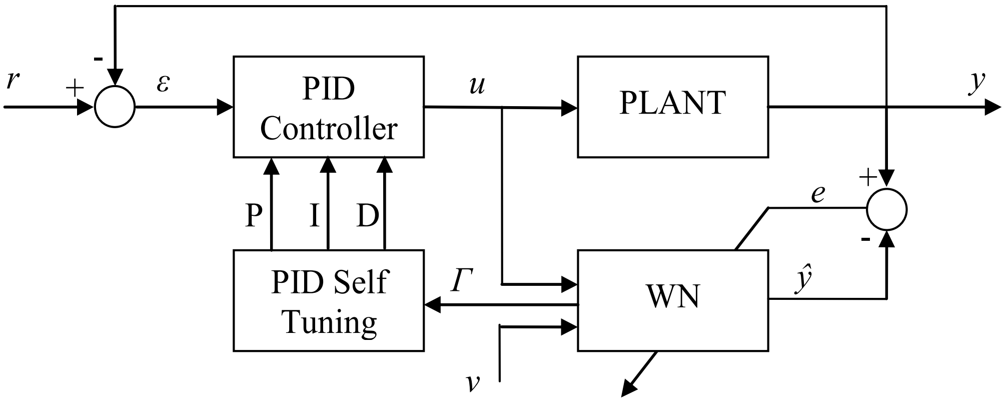


**Figure 1.4** Adaptive PID Controller

The author observed that in the first scheme the WN tracked the set-point reference faster than the actual plant, while in the second scheme the plant response was faster than the WN. The proposed controllers were also utilized in the cases when Gaussian noise contaminated the input of the WN or the output of the plant. The controllers were found to perform better in the case of input noise contamination than in the case of output noise contamination. While the self-tuning wavelet network controller was found to provide quicker tracking adjustment to control changes, the adaptive PID controller was found to be more robust and less sensitive to noise. Cheng, Chen and Shiau [18] developed an adaptive wavelet network controller by using the technique of feedback linearization, the adaptive control scheme and the $H_\infty$ optimal control theory. They obtained an inversion-based nonlinear controller by employing the technique of feedback linearization. They then developed the structure of the adaptive wavelet controller using the certainty equivalent principal of adaptive control theory. The $H_\infty$ optimal control theory was used to obtain the adaptive wavelet network control law and the parameter update algorithm. Two examples – an inverted pendulum system and a two-link robotic system – were given to illustrate the proposed controller's design procedure and performance. In both examples the controller was shown to perform well by attenuating the tracking error. Wai and Chang [19] presented an intelligent control system for an induction servo motor drive utilizing a wavelet network. The WN controller had adaptive learning rates that were derived using the discrete-type Lyapunov stability theorem to guarantee the convergence of the tracking error. The authors used a four layer wavelet network with fourteen one-dimensional wavelets and seven two-dimensional wavelets. A periodic reference signal was used to test the effectiveness of the proposed control system and the simulation and experimental results showed that the controlled servo motor drive had good tracking control performance and good robustness to uncertainties under wide operating ranges.

## 1.2 Objectives of Research

The goal of this work is to investigate wavelet networks for system identification and control. The first part of this work focuses on the design of wavelet networks for nonlinear system identification, where the nonlinearity is identified explicitly as a cascaded block to the linear part of the system. The second part addresses the design of a nonlinear controller by utilizing a wavelet network for the self-tuning control problems of highly nonlinear systems. The proposed method shows how a wavelet network can be combined with a self-tuning control algorithm to control known or unknown discrete-time complex systems. The gradient descent learning algorithm has been deployed to construct this adaptive controller. The control scheme used in this work is an adaptive PID controller based on a self-tuning wavelet network. The original contributions of this work are introducing new wavelet network architecture and defining an adaptive mechanism for the PID controller parameters.

## 1.3 Research Organization

This work is organized as follows: chapter one introduces a general overview of the methodology used. Fundamentals of wavelets and the wavelet transform are presented in chapter two. Then in chapter three wavelet networks are introduced and their application in nonlinear function approximation is presented. Chapter four explains the use of wavelet networks in the dynamic modeling of nonlinear systems along with several examples that highlight the advantage of the new wavelet network architecture, the dynamic wavelet network (DWN), over conventional wavelet networks. The effect of noise on the performance of the DWN is then investigated. Chapter five presents the control structure and design of an adaptive PID-DWN controller and its application in two case studies, including the effect of noise on the performance of the controller. The conclusion of this work is given in chapter six.

# CHAPTER 2

# WAVELETS

The term *wavelet* means a "little wave". This little wave must have at least a minimum oscillation and a fast decay to zero, in both positive and negative directions, of its amplitude. This property is analogous to an admissibility condition of a function that is required for the wavelet transform [20].

Sets of "wavelets" are employed to approximate a signal, whereby the goal is to find a set of daughter wavelets constructed by a dilated (scaled or compressed) and translated (shifted) original wavelet or "mother" wavelet, that best represent the signal. So by "traveling" from the large scales toward the fine scales, one "zooms in" and arrives at more and more exact representations of the given signal.

The mother wavelet used in this research is the Gaussian wavelet of order 1, which is the first derivative of a Gaussian function. The equation of this wavelet is

$$\phi(x) = -xe^{-\frac{x^2}{2}} \tag{2.1}$$

This wavelet is regarded as a differentiable version of the Haar wavelet, and has the universal approximation property [4]. Figure 2.1 shows the Gaussian wavelet of order 1.

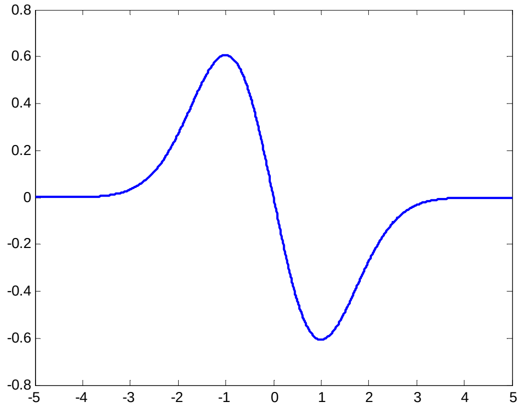


**Figure 2.1** Gaussian Wavelet of Order 1

Other wavelet functions include the Morlet wavelet and the Mexican Hat wavelet, among others. The Morlet wavelet, named after Jean Morlet, has the following equation [21]

$$\phi(x) = e^{-\frac{x^2}{2}} \cos(\omega_0 x) \tag{2.2}$$

This function is the real part of the Morlet's basic wavelet function, which is a multiplication of the Fourier basis with a Gaussian window [24]:

$$h(t) = e^{-\frac{t^2}{2}} e^{j\omega_0 t} \tag{2.3}$$

The imaginary part is a Sin-Gaussian function. The Cos-Gaussian wavelet is a real even function. Figure 2.2 shows the Morlet wavelet.

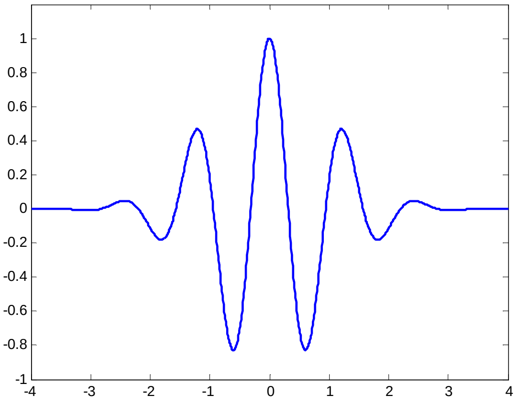


**Figure 2.2** Morlet Wavelet

The Mexican Hat wavelet is proportional to the second derivative function of the Gaussian probability density function and its equation is

$$\phi(x) = \left(\frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}}\right)\left(1 - x^2\right)e^{-\frac{x^2}{2}} \tag{2.4}$$

This wavelet is from the family of POLYnomials WindOwed with Gaussians (POLYWOG) wavelets. It was introduced by Gabor and is well known as the Laplacian operator, mostly used for zero-crossing multiresolution image edge detection [23]. The Mexican Hat wavelet is even and real valued. Figure 2.3 shows the Mexican Hat wavelet.

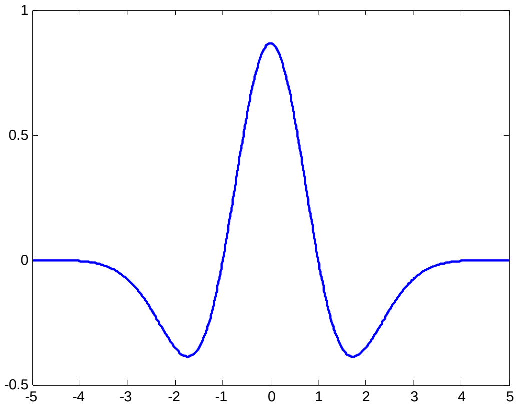
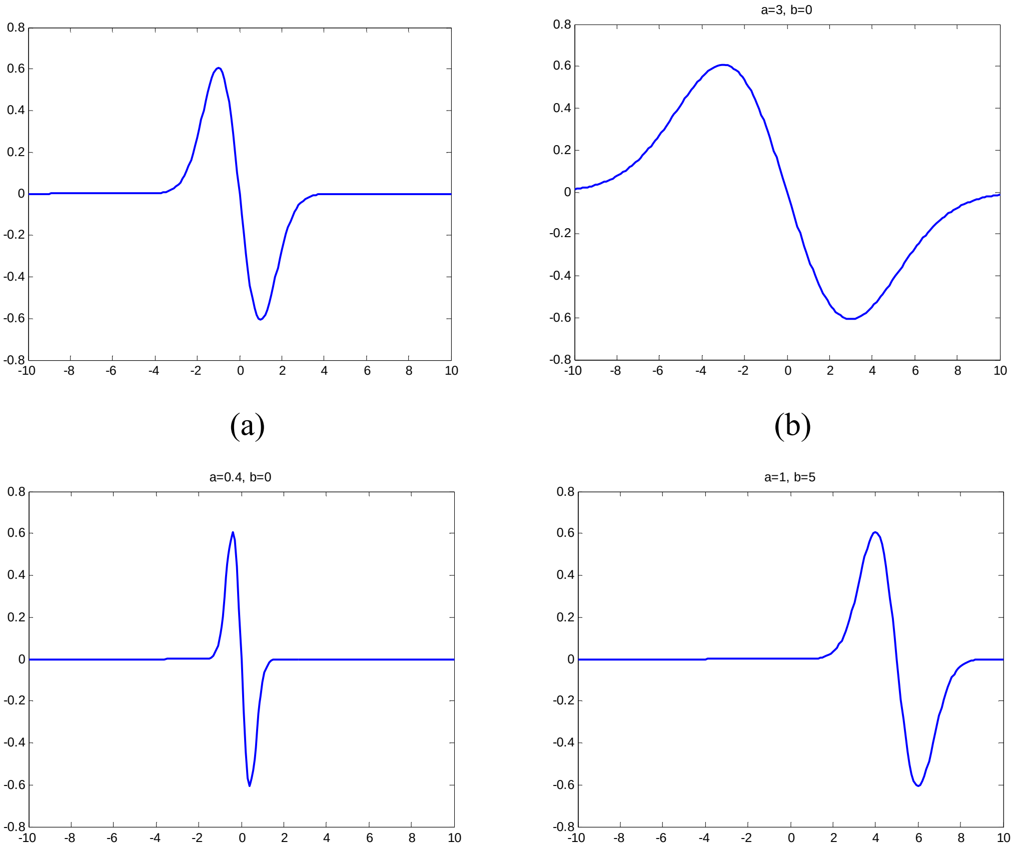


**Figure 2.3** Mexican Hat Wavelet

Figure 2.4 (a) shows the Gaussian wavelet of order 1 and (b)-(d) show possible daughter wavelets obtained from the mother wavelet using different dilation (*a*) and translation (*b*) parameters. Note the constant shape of these daughter wavelets; the same number of oscillations is in each wavelet.



(a)

(b)

(c)

(d)

**Figure 2.4 (a)** GaussianWavelet of Order 1 (mother) and **(b)-(d)** Daughter Wavelets

## 2.1 Wavelet Transform

The Wavelet Transform (WT) was developed as an alternative approach to the Short Time Fourier Transform (STFT). The STFT was developed as a signal processing tool for the analysis of non-stationary signals. The idea behind using the STFT was to get a time-frequency representation of signals with time-varying frequencies. This representation would enable the identification of the points in time at which the different frequencies occurred. In the STFT a window function is used to look at the signal, where it is assumed that the signal is stationary across the width of that window. However, the fixed width of the window function gives rise to a resolution problem. A narrow window gives good time resolution but poor frequency resolution, while a wide window gives good frequency resolution but poor time resolution. The WT solves the dilemma of resolution to a certain extent by analyzing a signal using Multi Resolution Analysis (MRA). MRA analyzes the signal at different frequencies with different resolutions. Spectral components are not resolved equally as done in the STFT. MRA was designed to give good time resolution and poor frequency resolution at high frequencies, and good frequency resolution and poor time resolution at low frequencies. This approach is suitable when the signal at hand has high frequency components for short durations and low frequency components for long durations. It so happens that the signals encountered in practical applications are often of this type [27].

The wavelet transform is an operation that transforms a function by integrating it with modified versions of some kernel function [25]. The kernel function is called the mother wavelet, and the modified version is its daughter wavelet. For a function to be a mother wavelet it must be *admissible*. A wavelet is said to be admissible if it satisfies the Grossmann-Morlet conditions; that is, it must be oscillatory with fast decay to zero and its DC content must be zero. More rigorously, a function $h \in L^2(R)$ (the set of all square integrable or a finite energy function) is admissible if [22]

$$c_h = \int_{-\infty}^{\infty} \frac{|H(\omega)|^2}{|\omega|} d\omega < \infty \tag{2.2}$$

where $H$(w) is the Fourier transform of $h$(t). The constant $c_h$ is the admissibility constant of the function $h$(t), and the requirement that it is finite allows for inversion of the wavelet transform [26]. Any admissible function can be a mother wavelet. For a given $h$(t), the condition $c_h < \infty$ holds only if $H(0) = 0$ i.e., the wavelets are inherently band-pass filters in the Fourier domain or equivalently, if $\int_{-\infty}^{\infty} h(t)dt = 0$ (zero-mean value in time domain).

The wavelet transform of a function $f \in L^2(R)$ with respect to a given admissible mother wavelet $h$(t) is defined as [23]:

$$W_f(a,b) = \int_{-\infty}^{\infty} f(t) h_{a,b}^{*}(t)dt \tag{2.3}$$

Where * denotes the complex conjugate. However, most wavelets are real valued. The daughter wavelets are generated from a single mother wavelet $h$(t) by dilation and translation:

$$h_{a,b}(t) = \frac{1}{\sqrt{a}} h\left(\frac{t-b}{a}\right) \tag{2.4}$$

Where $a > 0$ is the dilation factor and $b$ is the translation factor. The constant $a^{-1/2}$ term is for energy normalization (unitary affine mapping) that keeps the energy of the daughter wavelet equal to the energy of the original mother wavelet.

An example of the wavelet transform is presented. A signal with time varying frequency (chirp) is analyzed using the WT with the Gaussian wavelet of order 1 as the mother wavelet. Figure 2.5 shows the chirp signal with a frequency range of 0.01 to 1 Hz.

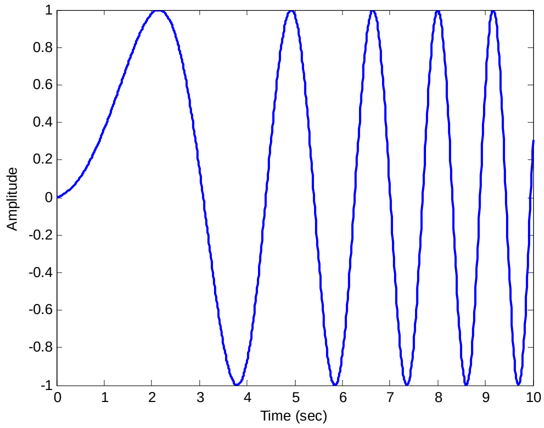


**Figure 2.5** Chirp Sinal (0.01 – 1 Hz)

Figure 2.6 shows the wavelet transform of the chirp signal. The WT contains information about the signal such as, the frequencies present in the signal, the times at which these frequencies occur, and the relative magnitudes of those frequencies. The WT represents frequencies as scales, which are inversely proportional to one another. Thus, lower scales correspond to higher frequencies, while higher scales correspond to lower frequencies. From the WT, shown in Figure 2.6, it can be seen that the signal has a lower frequency initially and then its frequency gradually increases with time. Furthermore, the relative magnitudes decrease with the rise in frequency.

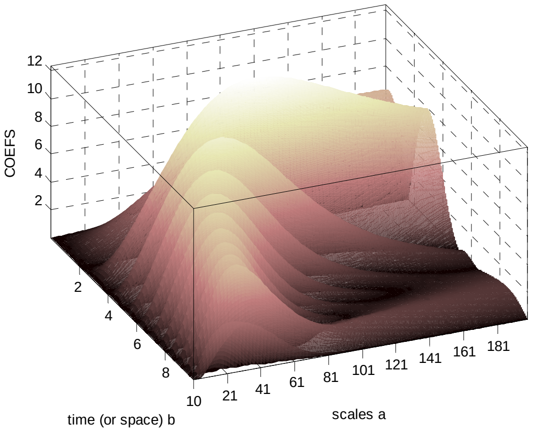


**Figure 2.6** Wavelet Transform of Chirp Sinal

## 2.2 Conclusion

The term *wavelet* means a "little wave". Sets of "wavelets" are employed to approximate a signal, whereby the goal is to find a set of daughter wavelets constructed by a dilated (scaled or compressed) and translated (shifted) original wavelet or "mother" wavelet, that best represent the signal. For a function to be a mother wavelet it must be *admissible*. A wavelet is said to be admissible if it satisfies the Grossmann-Morlet conditions; that is, it must be oscillatory with fast decay to zero and its DC content must be zero. The mother wavelet used in this research is the Gaussian wavelet of order 1, which is the first derivative of a Gaussian function.

The wavelet transform (WT) is a useful signal processing tool for the analysis of non-stationary signals. It was developed as an alternative approach to the Short Time Fourier Transform. The WT employs wavelets to resolve the multiple frequency components of time varying signals and identify the points in time at which they occur. By analyzing a signal using Multi Resolution Analysis (MRA), the WT can give good time resolution at high frequencies, and good frequency resolution at low frequencies.

# CHAPTER 3

# WAVELET NETWORKS

---

Combining the wavelet transform theory of Chapter 2 with the basic concept of neural networks led to the development of a new mapping network called a wavelet network. A wavelet network approximates any desired signal $y$(t) by generalizing a linear combination of a set of daughter wavelets $\varphi_{a,b}$(t), where $\varphi_{a,b}$(t) are generated by dilation, $a$, and translation, $b$, from a mother wavelet $\varphi$(t):

$$\varphi_{a,b}(t) = \varphi\left(\frac{t-b}{a}\right) \tag{3.1}$$

Where the dilation factor $a > 0$. Note that equation (3.1) is similar to that of equation (2.4), but without the energy normalization.

## 3.1 Models

The conventional wavelet network has the architecture shown in Figure 3.1, where the network output $\hat{y}$ is computed as

$$\hat{y} = \sum_{j=1}^{Nw} c_j \Phi_j + \sum_{k=1}^{Ni} a_k x_k + a_0 \tag{3.2}$$

The network has an input layer having $N_i$ inputs, a wavelet layer having $N_w$ weighted wavelet neurons and an output layer having a linear output neuron. The coefficients of the linear part of the network are called direct connections [12].

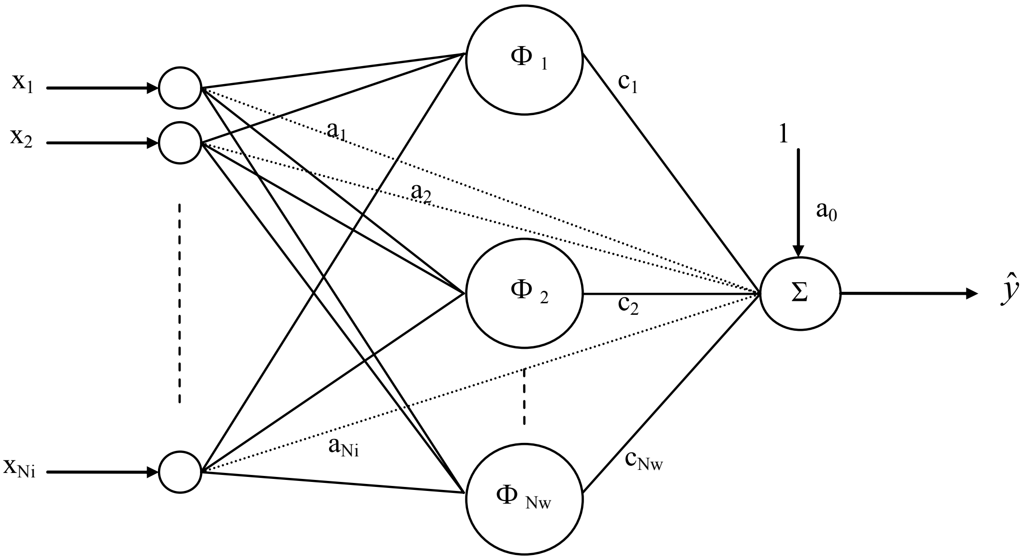


**Figure 3.1** Conventional Wavelet Network

Conventional wavelet networks used for modeling multi-input processes use multidimensional wavelets [12] which can be constructed as the product of $N_i$ scalar wavelets:

$$\Phi_j = \prod_{k=1}^{Ni} \phi(z_{jk}) \tag{3.3}$$

$$z_{jk} = \frac{x_k - m_{jk}}{d_{jk}} \tag{3.4}$$

$N_i$ is the number of input variables, and $m_j$ and $d_j$ are the translation and dilation vectors respectively.

The conventional wavelet network model can be used for both static and dynamic nonlinear system modeling.

The second model is the proposed new wavelet network scheme, called the Dynamic Wavelet Network (DWN). The DWN was designed for the dynamic modeling of nonlinear single input single output (SISO) systems. It is comprised of two cascaded sections: a single input single output wavelet network and a recursive filter. The DWN architecture is shown in Figure 3.2.

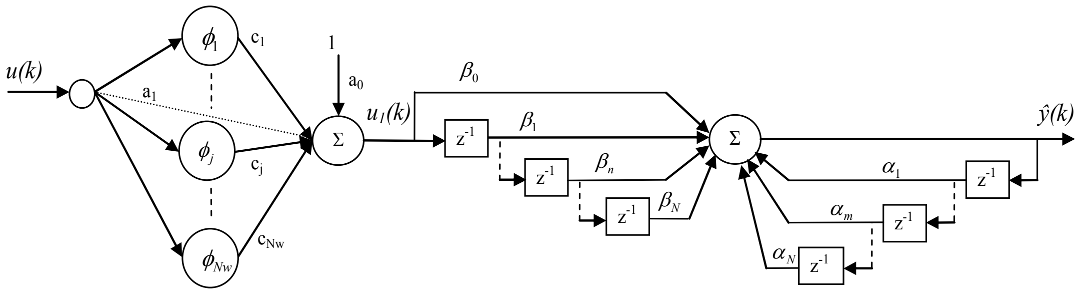


**Figure 3.2** Dynamic Wavelet Network

The network output is computed as

$$\hat{y}(k) = \sum_{n=0}^{N} \beta_n u_1(k-n) + \sum_{m=1}^{N} \alpha_m \hat{y}(k-m) \tag{3.5}$$

$N$ is the order of the system, $\beta_n$ are the feedforward coefficients and $\alpha_m$ are the feedback coefficients of the filter.

The output of the nonlinear wavelet network section written as

$$u_1(k) = \sum_{j=1}^{Nw} c_j \phi_j\left(k\right) + a_1 u(k) + a_0 \tag{3.6}$$

## 3.2 Training

Wavelet network training is based on minimizing the following quadratic cost function [14]:

$$J(\theta) = \frac{1}{2}\sum_{n=1}^{Np}\left(e^n\right)^2 = \frac{1}{2}\sum_{n=1}^{Np}\left(y^n - \hat{y}^n\right)^2 \tag{3.7}$$

Where $e_n = y^n - \hat{y}^n$ is the error between the target output, $y^n$, and the corresponding wavelet network output, $\hat{y}^n$, for training pattern $n$, and $N_p$ is the number of elements in the training set.

The network parameters are represented by the set $\theta$. For the conventional wavelet network, $\theta = \{m_{jk}, d_{jk}, c_j, a_k, a_0\}$, with $j = 1, \ldots, N_w$ and $k = 1, \ldots, N_i$, where $a_0$ is the output bias, $a_k$ are the direct connection weights, $c_j$ are the wavelet neuron output weights, $m_{jk}$ are the translations and $d_{jk}$ are the dilations. For the DWN, $\theta = \{m_j, d_j, c_j, a_1, a_0, \beta_n, \alpha_m\}$, with $j = 1, \ldots, N_w$, $k = 1, \ldots, N_i$, $n = 0,\ldots,N$ and $m = 1,\ldots,N$, where $\beta_n$ are the feedforward coefficients and $\alpha_m$ are the feedback coefficients.

The minimization is performed based on the gradient descent algorithm. The partial derivative of the cost function with respect to $\theta$ is:

$$\frac{\partial J}{\partial \theta} = -\sum_{n=1}^{Np} e^n \frac{\partial \hat{y}^n}{\partial \theta} \tag{3.8}$$

The components of the above vector for the conventional wavelet network are:

$$\frac{\partial \hat{y}}{\partial a_0} = 1 \tag{3.9}$$

$$\frac{\partial \hat{y}}{\partial a_k} = x_k \text{, for } k = 1,\dots, N_i \tag{3.10}$$

$$\frac{\partial \hat{y}}{\partial c_j} = \Phi_j \text{, for } j = 1,\dots, N_w \tag{3.11}$$

$$\frac{\partial \hat{y}}{\partial m_{jk}} = c_j \bullet \frac{\partial \Phi_j}{\partial z_{jk}} \bullet \frac{\partial z_{jk}}{\partial m_{jk}} = c_j \bullet \frac{\partial \Phi_j}{\partial z_{jk}} \bullet \left( -\frac{1}{d_{jk}} \right) \tag{3.12}$$

where $\frac{\partial \Phi_j}{\partial z_{jk}} = \phi(z_{j1})\phi(z_{j2})\dots.\phi(z_{jk})'\dots.\phi(z_{jN})$, and $\phi(z_{jk}) = -z_{jk} e^{-\frac{z_{jk}^2}{2}}$ (3.13)

$$\frac{\partial \hat{y}}{\partial d_{jk}} = c_j \bullet \frac{\partial \Phi_j}{\partial z_{jk}} \bullet \frac{\partial z_{jk}}{\partial d_{jk}} = c_j \bullet \frac{\partial \Phi_j}{\partial z_{jk}} \bullet \left( -\frac{z_{jk}}{d_{jk}} \right) \tag{3.14}$$

$k = 1,\dots, N_i$ and $j = 1,\dots, N_w$

The components of the above vector for the DWN are:

$$\frac{\partial \hat{y}(k)}{\partial a_0} = \sum_{n=0}^{N} \beta_n \frac{\partial u_1(k-n)}{\partial a_0} + \sum_{m=1}^{N} \alpha_m \frac{\partial \hat{y}(k-m)}{\partial a_0} \tag{3.15}$$

$$\frac{\partial \hat{y}(k)}{\partial a_1} = \sum_{n=0}^{N} \beta_n \frac{\partial u_1(k-n)}{\partial a_1} + \sum_{m=1}^{N} \alpha_m \frac{\partial \hat{y}(k-m)}{\partial a_1} \tag{3.16}$$

$$\frac{\partial \hat{y}(k)}{\partial c_j} = \sum_{n=0}^{N} \beta_n \frac{\partial u_1(k-n)}{\partial c_j} + \sum_{m=1}^{N} \alpha_m \frac{\partial \hat{y}(k-m)}{\partial c_j} \tag{3.17}$$

$$\frac{\partial \hat{y}(k)}{\partial m_j} = \sum_{n=0}^{N} \beta_n \frac{\partial u_1(k-n)}{\partial m_j} + \sum_{m=1}^{N} \alpha_m \frac{\partial \hat{y}(k-m)}{\partial m_j} \tag{3.18}$$

$$\frac{\partial \hat{y}(k)}{\partial d_j} = \sum_{n=0}^{N} \beta_n \frac{\partial u_1(k-n)}{\partial d_j} + \sum_{m=1}^{N} \alpha_m \frac{\partial \hat{y}(k-m)}{\partial d_j} \tag{3.19}$$

where $j = 1,\dots, N_w$

$$\frac{\partial \hat{y}(k)}{\partial \beta_n} = u_1(k-n) + \sum_{m=1}^{N} \alpha_m \frac{\partial \hat{y}(k-m)}{\partial \beta_n} \text{ for } n = 0,\dots, N \tag{3.20}$$

$$\frac{\partial\hat{y}(k)}{\partial\alpha_m}=\hat{y}(k-m)+\sum_{m=1}^{N}\alpha_m\frac{\partial\hat{y}(k-m)}{\partial\alpha_m}\ \text{for } m=1,\ldots,N \tag{3.21}$$

For modeling a first order system, the derivative equations would be as follows:

$$\frac{\partial\hat{y}(k)}{\partial a_0}=\beta_0+\beta_1+\alpha_1\frac{\partial\hat{y}(k-1)}{\partial a_0} \tag{3.22}$$

$$\frac{\partial\hat{y}(k)}{\partial a_1}=\beta_0u(k)+\beta_1u(k-1)+\alpha_1\frac{\partial\hat{y}(k-1)}{\partial a_1} \tag{3.23}$$

$$\frac{\partial\hat{y}(k)}{\partial c_j}=\beta_0\phi_j(k)+\beta_1\phi_j(k-1)+\alpha_1\frac{\partial\hat{y}(k-1)}{\partial c_j} \tag{3.24}$$

$$\frac{\partial\hat{y}(k)}{\partial m_j}=\beta_0A(k)+\beta_1A(k-1)+\alpha_1\frac{\partial\hat{y}(k-1)}{\partial m_j} \tag{3.25}$$

where $A(k)=c_j\times\frac{\partial\phi_j(k)}{\partial z_j(k)}\times\frac{\partial z_j(k)}{\partial m_j}$, $\phi_j(k)=-z_j(k)e^{-\frac{z_j(k)^2}{2}}$, and $z_j(k)=\frac{u(k)-m_j}{d_j}$

$$\frac{\partial\hat{y}(k)}{\partial d_j}=\beta_0B(k)+\beta_1B(k-1)+\alpha_1\frac{\partial\hat{y}(k-1)}{\partial d_j} \tag{3.26}$$

where $B(k)=c_j\times\frac{\partial\phi_j(k)}{\partial z_j(k)}\times\frac{\partial z_j(k)}{\partial d_j}$, and $j = 1,\ldots, N_w$

$$\frac{\partial\hat{y}(k)}{\partial\beta_0}=u_1(k)+\alpha_1\frac{\partial\hat{y}(k-1)}{\partial\beta_0} \tag{3.27}$$

$$\frac{\partial\hat{y}(k)}{\partial\beta_1}=u_1(k-1)+\alpha_1\frac{\partial\hat{y}(k-1)}{\partial\beta_1} \tag{3.28}$$

$$\frac{\partial\hat{y}(k)}{\partial\alpha_1}=\hat{y}(k-1)+\alpha_1\frac{\partial\hat{y}(k-1)}{\partial\alpha_1} \tag{3.29}$$

The network parameter set $\theta$ is updated every epoch by using

$$\Delta\theta(k)=-\mu\frac{\partial J}{\partial\theta}+\gamma\Delta\theta(k-1) \tag{3.30}$$

Where $\mu$ is the learning rate and $\gamma$ is the momentum coefficient.

The learning rate and momentum coefficient are set in the interval (0, 1). The training is performed to achieve a specified mean squared error value, given by

$$MSE=\frac{1}{N_p}\sum_{n=1}^{Np}\left(y^n-\hat{y}^n\right)^2=\frac{2}{N_p}J(\theta) \tag{3.31}$$

## 3.3 Initialization Methods

There are two methods for initializing wavelet networks using the Gaussian wavelet of order 1 as mother wavelet, namely the heuristic method and the selection method [12]. The heuristic initialization method is a general one, used for both static and dynamic modeling of nonlinear systems, whereas the initialization by selection method is used for static modeling only.

The heuristic method takes into account the domain $[a_k, b_k]$ containing the values of the $k^{th}$ component of the input vectors $[x_1, \ldots, x_{\mathrm{Ni}}]$. The translation parameters are initialized so that the wavelets are centered in the parallelepiped defined by the $N_i$ intervals. The dilation parameters are initialized so that the wavelets extend initially over the whole input domain.

The translation parameters of wavelet $j$ are initialized to:

$$m_{jk} = \frac{1}{2}(a_k + b_k) \tag{3.32}$$

The dilation parameters of wavelet $j$ are initialized to:

$$d_{jk} = 0.2(b_k - a_k) \tag{3.33}$$

The other network parameters, (output bias, $a_0$, direct connection weights, $a_k$, wavelet neuron output weights, $cj$) are initialized to small random values. Figure 3.3 shows a flowchart for the training process of the wavelet network when using the heuristic initialization method.

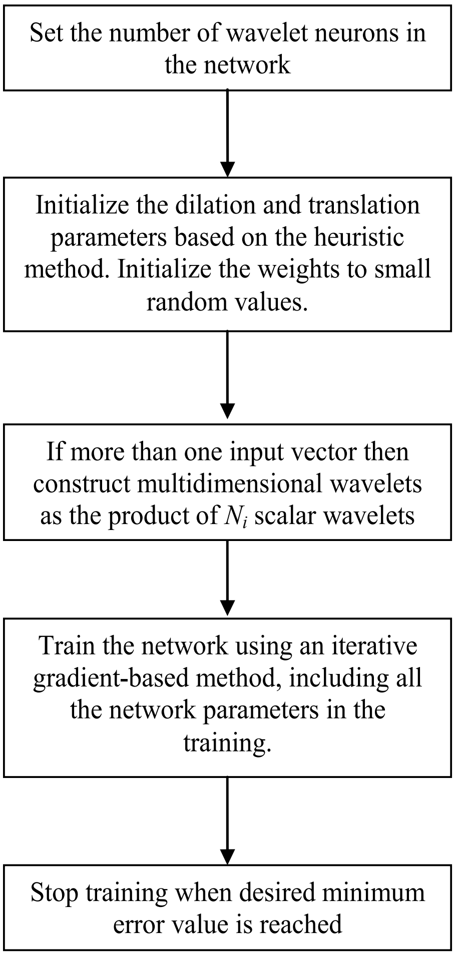

Set the number of wavelet neurons in the network
Initialize the dilation and translation parameters based on the heuristic method. Initialize the weights to small random values.
If more than one input vector then construct multidimensional wavelets as the product of $N_i$ scalar wavelets
Train the network using an iterative gradient-based method, including all the network parameters in the training.
Stop training when desired minimum error value is reached

**Figure 3.3** Heuristic Method Flowchart

The initialization by selection method involves building a library of wavelets that have their translation and dilation parameters initialized based on wavelet frames originating from the discrete wavelet transform [12].The dilation parameters are defined by $d_{jk} = 2^{-m}$, where $m$ is an integer. Considering three successive dilations, where the largest gives a wavelet extending over the whole domain of the corresponding input variable, since $m$ must be an integer, the three values will be:

$$\left\{\left[-\frac{\ln(0.2(b_k-a_k))}{\ln 2}\right]+1,\left[-\frac{\ln(0.2(b_k-a_k))}{\ln 2}\right]+2,\left[-\frac{\ln(0.2(b_k-a_k))}{\ln 2}\right]+3\right\} \tag{3.34}$$

Where [ ] represents the integer part operator.

For each dilated wavelet produced from the dilation set, a corresponding set of daughter wavelets with all possible translations in $[a_k, b_k]$ is produced and entered in the library, where $m_{jk} = 2^{-m}n$. These result in the following relation for translations:

$$a_k \le 2^{-m} n \le b_k \tag{3.35}$$

All the values of *n* obeying the previous condition are of interest and are determined as:

$$2^m a_k \leq n \leq 2^m b_k \tag{3.36}$$

Since *n* must be an integer, the values will be:

$$\left\{\left[2^m a_k\right]+1, \left[2^m a_k\right]+2, \ldots, \left[2^m b_k\right]\right\} \tag{3.37}$$

As a result, the number of translation parameters increases exponentially with $m$.

The wavelet library is ranked by the Gram-Schmidt method in order of decreasing relevance and the most relevant wavelets are selected to be used in the wavelet network.

The weights of the direct connections are estimated using standard least squares:

$$\hat{\theta}_{LS} = \left(X^T X\right)^{-1} X^T Y \tag{3.38}$$

Where $X$ is the matrix of input vectors, $Y$ is the training sequence, and $T$ is the transpose operator.

Subsequently, a new training sequence is derived from subtracting the output of the linear model of the wavelet network from the initial training sequence.

Therefore, the initialization by selection method comprises three main steps:

1. Generate a library of wavelets
2. Rank all wavelets in order of decreasing relevance
3. Select the most relevant wavelets to be used in the network

This method produces large libraries with a high modeling ability; however, the selection process becomes lengthy when dealing with multidimensional wavelets. Figure 3.4 shows a flowchart for the training process of conventional wavelet networks when using the initialization by selection method.

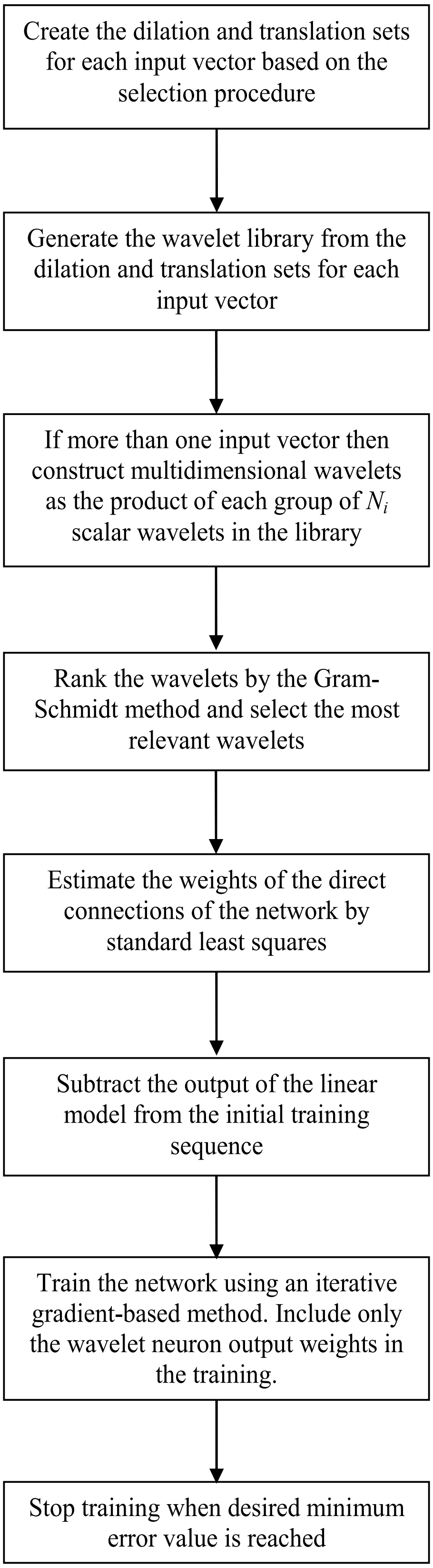

Create the dilation and translation sets for each input vector based on the selection procedure
Generate the wavelet library from the dilation and translation sets for each input vector
If more than one input vector then construct multidimensional wavelets as the product of each group of $N_i$ scalar wavelets in the library
Rank the wavelets by the Gram-Schmidt method and select the most relevant wavelets
Estimate the weights of the direct connections of the network by standard least squares
Subtract the output of the linear model from the initial training sequence
Train the network using an iterative gradient-based method. Include only the wavelet neuron output weights in the training.
Stop training when desired minimum error value is reached

**Figure 3.4** Selection Method Flowchart

For the DWN, the network parameters shared with the conventional WN are initialized according to the heuristic method. Furthermore, the linear coefficients, $\beta_n$, $\alpha_m$, are initialized using least squares estimation. This step is similar to the estimation of the direct connections for the conventional WN when using the initialization by selection method. In this case, the DWN output is considered as:

$$\hat{y}(k) = \sum_{n=0}^{N} \beta_n u(k-n) + \sum_{m=1}^{N} \alpha_m y(k-m) \tag{3.39}$$

Thus, the input matrix will consist of delayed sample vectors of the system input, $u$, and the system output, $y$. This is done because it results in a faster convergence during training compared to the case when the linear coefficients are initialized to small random numbers. Equation 3.39 is similar in form to the linear model of the feedforward wavelet predictor model, which is described in Chapter 4.

## 3.4 Nonlinear Function Approximation

In static modeling the wavelet network performs a mapping between a multivariable input **x** and the output $y$ of a nonlinear function. A block diagram of the model training is shown in Figure 3.5.

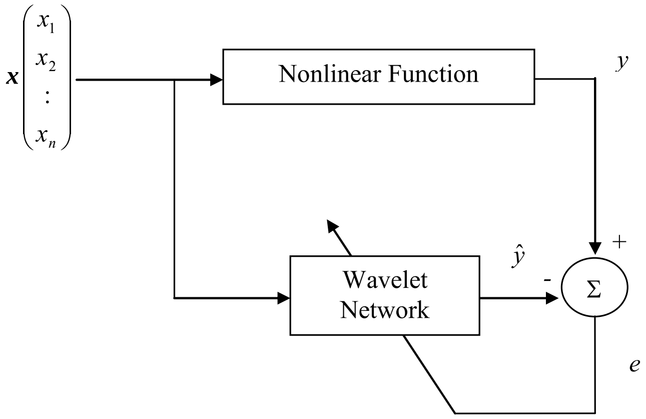
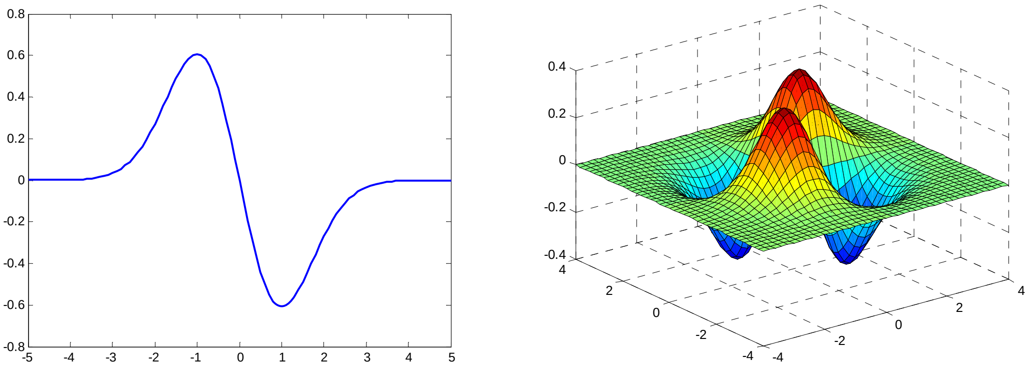


**Figure 3.5** Nonlinear Function Approximation

Two examples are presented. The first involves approximating a nonlinear function of one input variable and the second involves approximating a nonlinear function of two input variables. Each example was performed with wavelet networks initialized by the heuristic and selection methods respectively, and then with a neural network as a baseline comparison. As mentioned in Chapter 2, the Gaussian wavelet of order 1 was used as mother wavelet:

$$\phi(x) = -xe^{-\frac{x^2}{2}} \tag{3.40}$$

Figure 3.6 shows the one and two dimensional mother wavelets used in training.



(a) (b)

**Figure 3.6 (a)** One and **(b)** Two Dimensional Mother Wavelets

The nonlinear function of the first example is as follows, and a plot of the function is shown in Figure 3.7:

$$f(x) = \begin{cases} -2.186x - 12.864 & for \quad x \in [-10,-2) \\ 4.246x & for \quad x \in [-2,0) \\ 10\exp(-0.05x - 0.5)\sin(x(0.03x + 0.7)) & for \quad x \in [0,10] \end{cases}$$

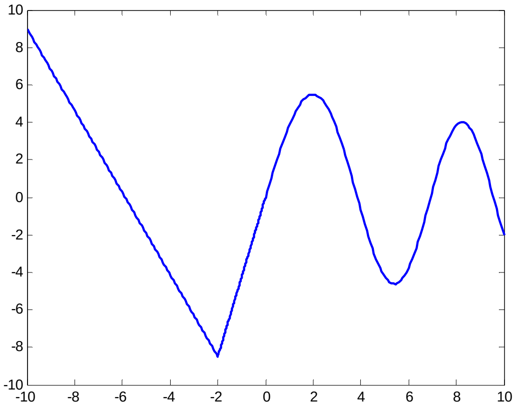


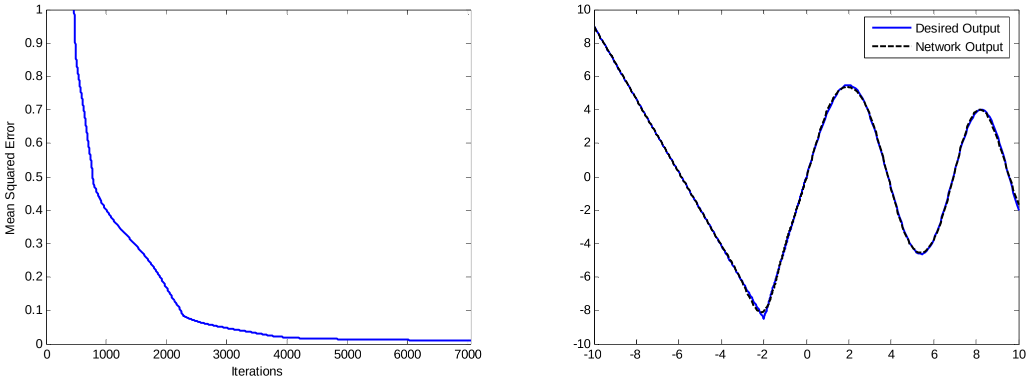
**Figure 3.7** Desired Output of First Example

The first wavelet network was initialized based on the heuristic method. The number of wavelets used was 10. The weights were initialized to small random numbers. The learning rate and momentum term were set to 0.1 and 0.8 respectively. The simulation was run to achieve a mean squared error of $1x10^{-2}$. The statistics are shown in Table 3.1. Plots of the MSE and wavelet network output are shown in Figure 3.8 (a) and (b) respectively.

Table 3.1

Simulation Statistics for One Input Function Approximation using Heuristic Method

| MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1x10^{-2}$ | 7,043 | 41 seconds |



(a) (b)

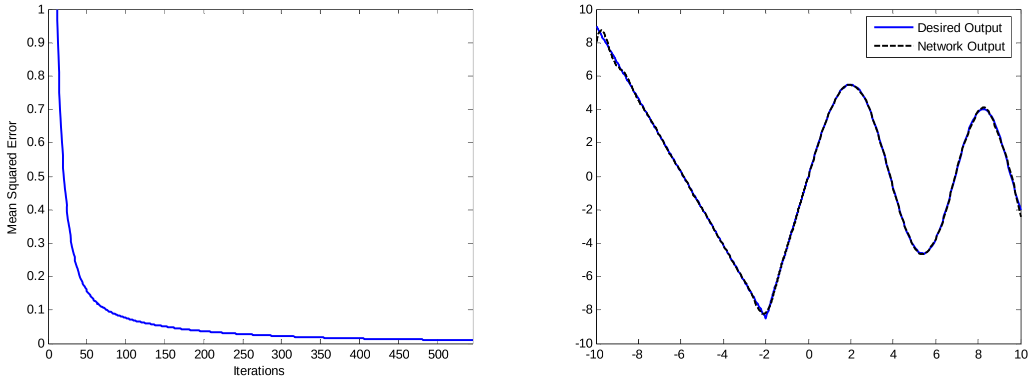
**Figure 3.8 (a)** MSE and **(b)** WN Output (Heuristic Method)

The second wavelet network was initialized based on the selection method. A library of 73 wavelets was generated from the set of dilations and translations. The output bias and direct connection weights were estimated using least squares. The wavelet neuron output weights were initialized to small random numbers. The learning rate and momentum term were set to 0.5 and 0.5 respectively. The simulation was run to achieve a mean squared error of $1x10^{-2}$. The statistics are shown in Table 3.2 (simulation time includes library generation time). Plots of the MSE and wavelet network output are shown in Figure 3.9 (a) and (b) respectively.

Table 3.2

Simulation Statistics for One Input Function Approximation using Selection Method

| MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1x10^{-2}$ | 545 | 2 seconds |



(a) (b)

**Figure 3.9 (a)** MSE and **(b)** WN Output (Selection Method)

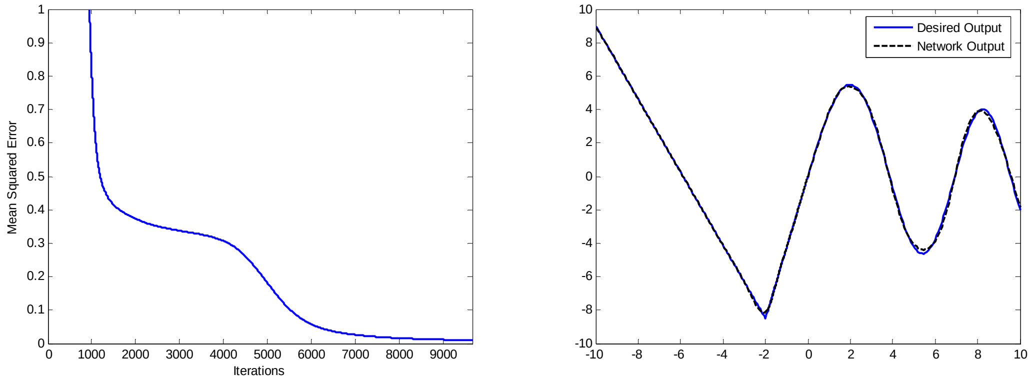
The simulation results showed that the selection method enabled the wavelet network to achieve faster convergence than the heuristic method.

Finally, a traditional feedforward neural network based on the backpropagation algorithm was used. It has three layers (input layer, hidden layer, output layer) with 10 hidden neurons, employing the sigmoid activation function. The network weights were initialized to small random values. The learning rate was set to $5x10^{-2}$ and the momentum coefficient to 0.9. The simulation was run to achieve a mean squared error of $1x10^{-2}$. The statistics are shown in Table 3.3. Plots of the MSE and neural network output are shown in Figure 3.10 (a) and (b) respectively.

Table 3.3

Simulation Statistics for One Input Function Approximation using Feedforward Neural Network

| MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1x10^{-2}$ | 9,651 | 58 seconds |



(a) (b)

**Figure 3.10 (a)** MSE and **(b)** NN Output

The simulation results showed that the neural network exhibited slower convergence than both wavelet networks.

The nonlinear function of the second example is as follows, and a plot of the function is shown in Figure 3.11:

$$f(x_1,x_2)=1.335[1.5(1-x_1)]+\exp(2x_1-1)\sin\left(3\pi(x_1-0.6)^2\right)+\exp(3(x_2-0.5))\sin\left(4\pi(x_2-0.9)^2\right)$$

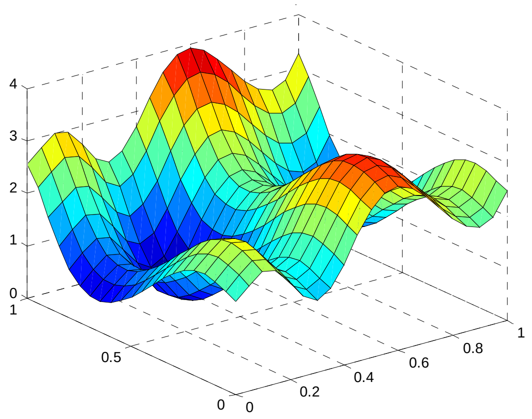


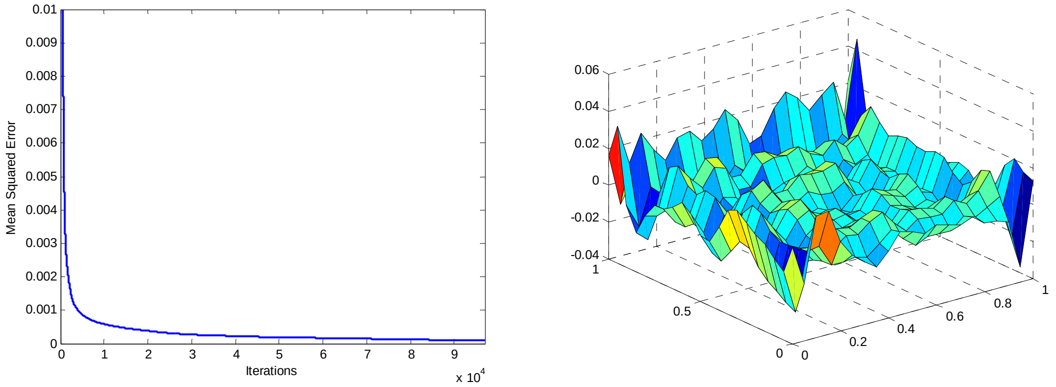
**Figure 3.11** Desired Output of Second Example

The first wavelet network was initialized based on the heuristic method. The number of wavelets used was 20. The weights were initialized to small random numbers. The learning rate and momentum term were set to 0.1 and 0.8 respectively. The simulation was run to achieve a mean squared error of 1x10$^{-4}$. The statistics are shown in Table 3.4. Plots of the MSE and wavelet network error are shown in Figure 3.12 (a) and (b) respectively.

Table 3.4

Simulation Statistics for Two Input Function Approximation using Heuristic Method

| MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1x10^{-4}$ | 96,961 | 1 hour 17 minutes |



(a) (b)

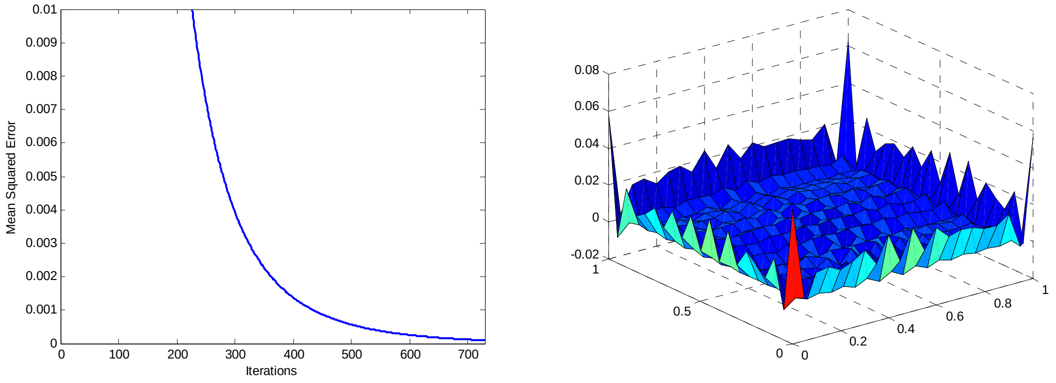
**Figure 3.12 (a)** MSE and **(b)** WN Error (Heuristic Method)

The second wavelet network was initialized based on the selection method. A library of 3,481 wavelets was generated from the sets of dilations and translations. The output bias and direct connection weights were estimated using least squares. The wavelet neuron output weights were initialized to small random numbers. The learning rate and momentum term were set to 0.5 and 0.5 respectively. The simulation was run to achieve a mean squared error of $1x10^{-4}$. The statistics are shown in Table 3.5 (simulation time includes library generation time). Plots of the MSE and wavelet network error are shown in Figure 3.13 (a) and (b) respectively.

Table 3.5

Simulation Statistics for Two Input Function Approximation using Selection Method

| MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1x10^{-4}$ | 729 | 8 minutes 17 seconds |



(a) (b)

**Figure 3.13 (a)** MSE and **(b)** WN Error (Selection Method)

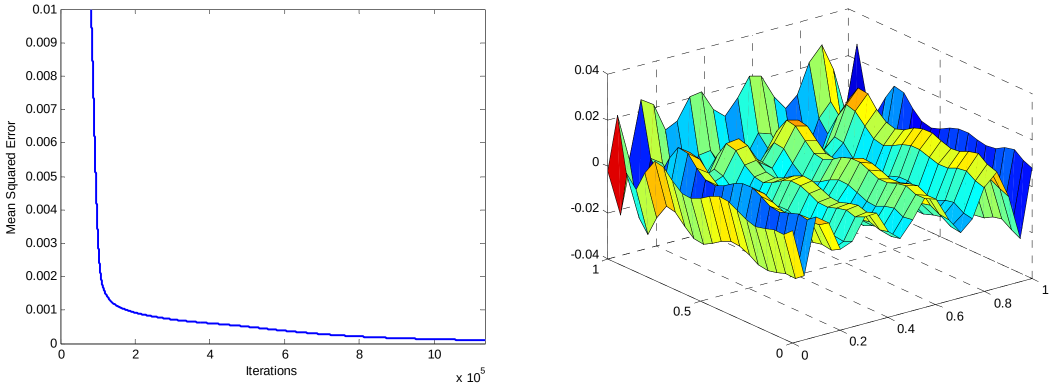
The simulation results showed that the selection method enabled the wavelet network to achieve faster convergence than the heuristic method.

Finally, a traditional feedforward neural network based on the backpropagation algorithm was used. It has three layers (input layer, hidden layer, output layer) with 20 hidden neurons, employing the sigmoid activation function. The network weights were initialized to small random values. The learning rate was set to $5x10^{-2}$ and the momentum coefficient to 0.9. The simulation was run to achieve a mean squared error of $1x10^{-4}$. The statistics are shown in Table 3.6. Plots of the MSE and neural network error are shown in Figure 3.14 (a) and (b) respectively.

Table 3.6

Simulation Statistics for One Input Function Approximation using Feedforward Neural Network

| MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1x10^{-4}$ | 1,136,105 | 18 hours 56 minutes |



(a) (b)

**Figure 3.14 (a)** MSE and **(b)** NN Error

The simulation results showed that the neural network exhibited much slower convergence than both wavelet networks.

In the two examples shown, the wavelet libraries generated with the selection method were not reduced in size by further ranking and selecting the most relevant wavelets. This process would have increased the total simulation time for the selection method and was shown to be unnecessary.

## 3.5 Conclusion

A wavelet network approximates any desired signal by generalizing a linear combination of a set of daughter wavelets generated by the dilation and translation of a mother wavelet. Conventional wavelet networks used for modeling multi-input processes use multidimensional wavelets which can be constructed as the product of scalar wavelets. The conventional wavelet network model can be used for both static and dynamic nonlinear system modeling. The Dynamic Wavelet Network (DWN) was designed for the dynamic modeling of nonlinear single input single output (SISO) systems. Wavelet network training is based on minimizing a quadratic cost function. The minimization is performed based on the gradient descent algorithm. The training is performed to achieve a specified mean squared error value (MSE).

There are two methods for initializing wavelet networks using the Gaussian wavelet of order 1 as mother wavelet, namely the heuristic method and the selection method The heuristic initialization method is a general one, used for both static and dynamic modeling of nonlinear systems, whereas the initialization by selection method is used for static modeling only. The selection method produces large libraries with a high modeling ability; however, the selection process becomes lengthy when dealing with multidimensional wavelets. Wavelet networks, initialized using the heuristic method, require all network parameters to be optimized during training, while those initialized using the selection method require only the wavelet neuron weights to be optimized during training. Hence, these two initialization methods enable the design of two different wavelet networks in terms of size as well as parameter optimization.

In static modeling the wavelet network performs a mapping between a multivariable input and the output of a nonlinear function. In the two examples shown on nonlinear function approximation, training time and number of iterations were significantly less with the selection method than with the heuristic method. This was because with the selection method, the network only needed to optimize one parameter, whereas with the heuristic method, all the network parameters needed to be optimized. Furthermore, with the selection method, the minimization of error between the wavelet network output and the desired output was more uniform across the function's domain than with the heuristic method. Therefore, the selection method was found to be superior to the heuristic method in the area of static modeling. When compared to traditional feedforward neural networks, both wavelet networks were shown to be superior.

# CHAPTER 4

# DYNAMIC MODELING

When wavelet networks are used in the framework of dynamic modeling, network training is performed in a recursive manner. There are two configurations used to train conventional wavelet networks, namely the feedforward wavelet predictor and the feedback wavelet predictor [14]. Based on the assumptions about noise, either feedforward or feedback predictors are used. If it is assumed that noise is acting on the states of the system, a feedforward predictor is used, whereas if noise is assumed to be acting on the output of the system, a feedback predictor is used. In the absence of noise, either configuration can be used.

In the feedforward wavelet predictor configuration, the wavelet network input consists of delayed samples of the system input *u(k)* and the system output *y(k)*. The number of inputs to the wavelet network increases with the order of the system being modeled. Hence, the wavelet network output is computed as

$$\hat{y}(k) = \sum_{j=1}^{Nw} c_j \Phi_j + \sum_{m=N+1}^{2N} a_m y(k-m+N) + \sum_{n=1}^{N} a_n u(k-n) + a_0 \tag{4.1}$$

Where $N$ is the order of the system being modeled.

A block diagram of the dynamic modeling of an $N^{th}$ order nonlinear system using a feedforward wavelet predictor is shown in Figure 4.1.

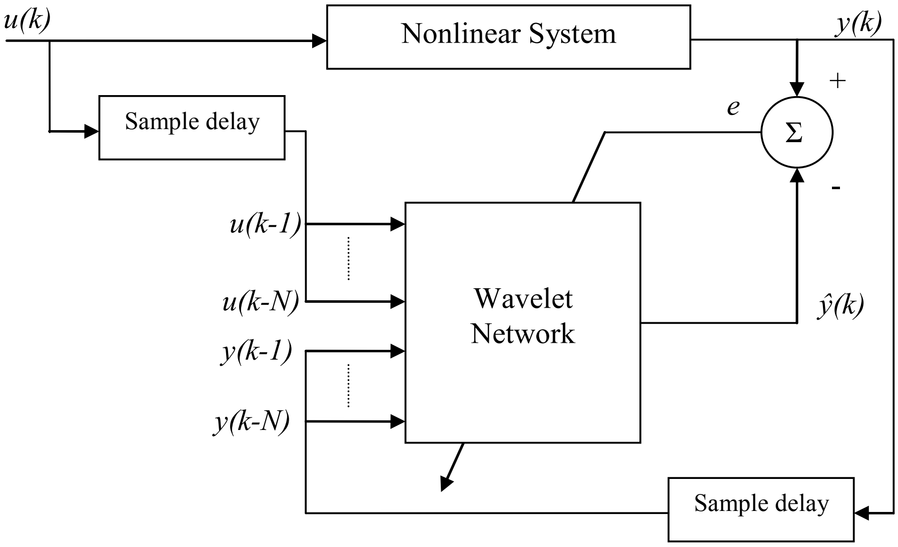


**Figure 4.1** Feedforward Wavelet Predictor

Dynamic modeling with the feedforward configuration can be similar to static modeling, since the wavelet network input data is known prior to training. Therefore, input variables are treated as vectors rather than scalars, and training is performed in batch mode, resulting in shorter simulation times.

In the feedback wavelet predictor configuration, the wavelet network input consists of delayed samples of the system input *u(k)* and the wavelet network output *ŷ(k)*. The number of inputs to the wavelet network increases with the order of the system being modeled. Hence, the wavelet network output is computed as

$$\hat{y}(k) = \sum_{j=1}^{Nw} c_j \Phi_j + \sum_{m=N+1}^{2N} a_m \hat{y}(k - m + N) + \sum_{n=1}^{N} a_n u(k - n) + a_0 \tag{4.2}$$

A block diagram of the dynamic modeling of an $N^{th}$ order nonlinear system using the feedback wavelet predictor is shown in Figure 4.2.

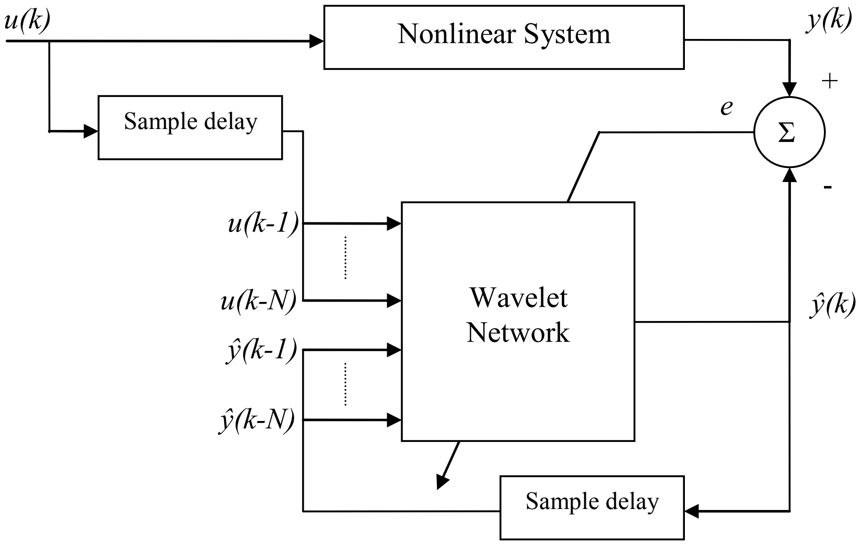


**Figure 4.2** Feedback Wavelet Predictor

Since the network input data is not known prior to training, input variables are treated as scalars and training must be performed in iterative mode.

Dynamic modeling with the DWN involves adjusting the size of the recursive filter section to accommodate the order of the system being modeled, as opposed to adjusting the number of inputs to the wavelet network with conventional dynamic modeling methods. This method enables the user to model higher order systems without the need for using multidimensional wavelets and thus avoiding the added complexity to the network. The network output is computed as

$$\hat{y}(k)=\sum_{n=0}^{N}\beta_n u_1(k-n)+\sum_{m=1}^{N}\alpha_m \hat{y}(k-m) \tag{4.3}$$

where

$$u_1(k)=\sum_{j=1}^{Nw} c_j\phi_j(k)+a_1 u(k)+a_0 \tag{4.4}$$

A block diagram of the dynamic modeling of an $N^{th}$ order nonlinear system using the DWN is shown in Figure 4.3.

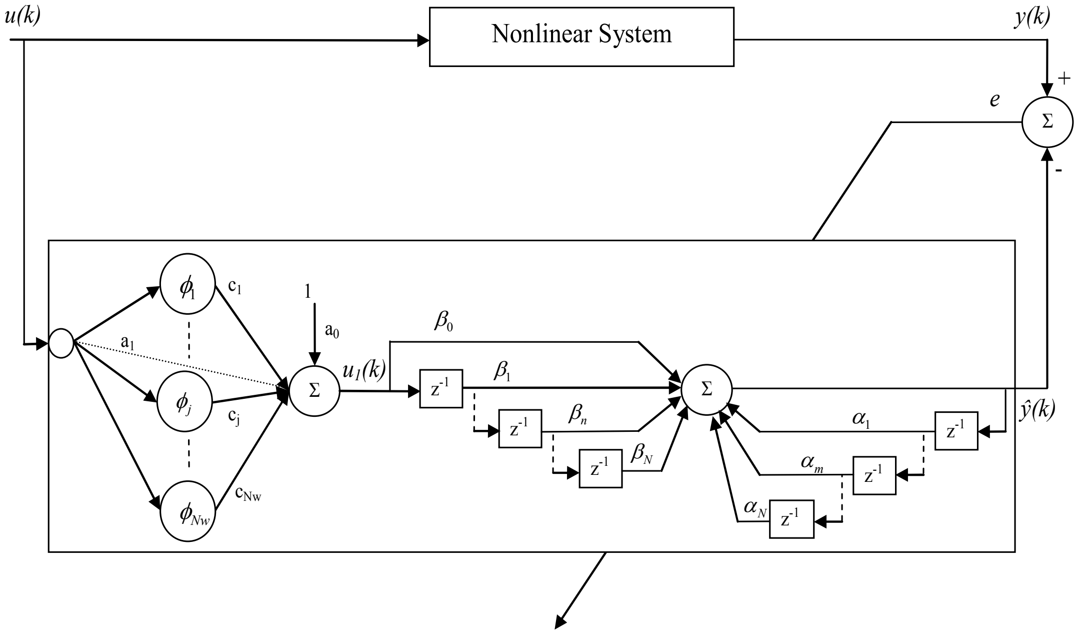


**Figure 4.3** Dynamic Modeling using DWN

Due to the sample delays in the filter section of the network, training must be performed in iterative mode.

## 4.1 First Order System

A first order system can be represented as

$$\frac{K}{1+\dfrac{s}{\omega_n}} \tag{4.5}$$

$K$ is the low frequency (or DC) gain and $\omega_n$ is the corner (or natural) frequency.

This representation is equivalent to that of a first order low pass filter, which implies that the system will respond in a similar manner to a given input. Inputs with frequencies below $\omega_n$ will be multiplied by the system's low frequency gain $K$, while those above $\omega_n$ will be attenuated by the system at a rate of 20 dB/dec. The system's natural frequency is at the point where the DC gain is less by 3dB. A Bode plot shows the frequency response of a first order system in Figure 4.4.

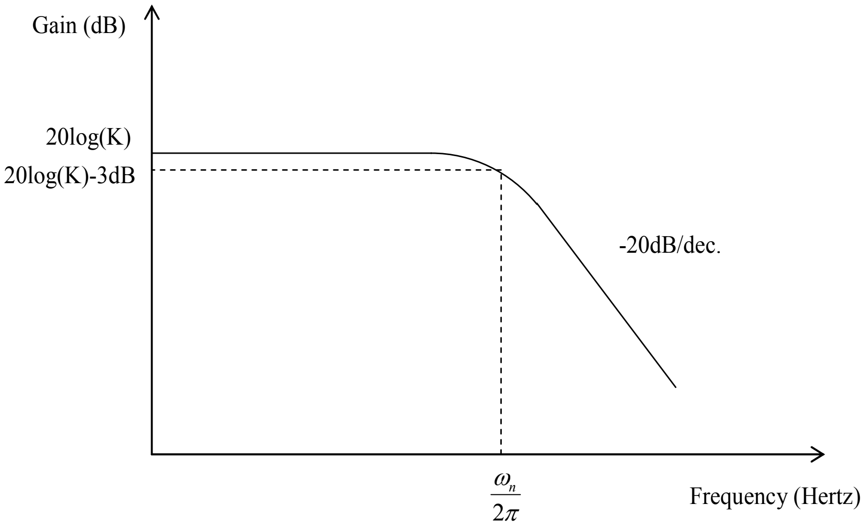


**Figure 4.4** First Order System Bode Plot

The system response was examined for the following first order system:

$$\frac{0.5}{1+0.5s} \tag{4.6}$$

Where $K = 0.5$ and $\omega_n = 2$ rad/sec.

A step response is shown in Figure 4.5. The system output stabilized at 0.5, which was half the amplitude of the input.

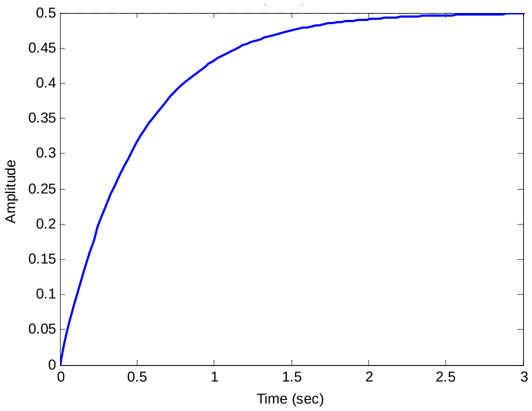
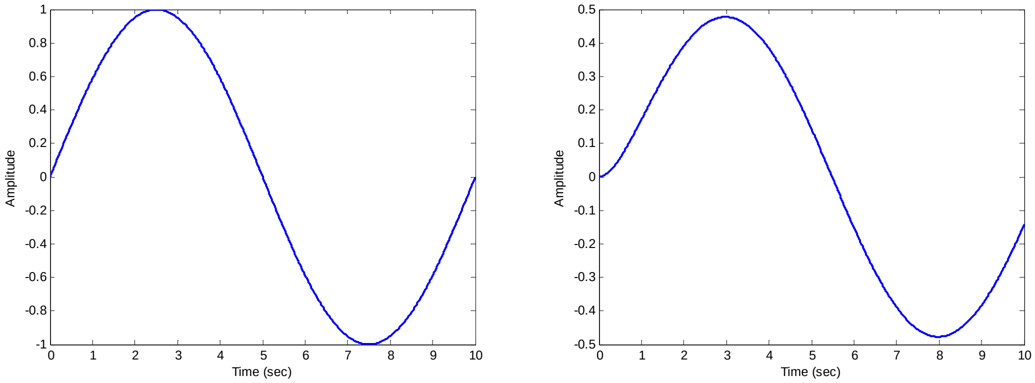


**Figure 4.5** First Order System Step Response

The second system response was to a sinusoid signal $u(\mathrm{k})=sin(0.2\pi \mathrm{k})$, shown in Figure 4.6 (a). The system output was a sinusoid with a smaller peak and shifted to the right, shown in Figure 4.6 (b).



(a) (b)

**Figure 4.6** First Order Linear System **(a)** Input and **(b)** Output

The frequency of the input was 0.1 Hertz and its maximum amplitude was 1, while the output had the same frequency but its maximum amplitude was about half the input, at 0.48, with a shift of about half a second or 17.4°. The sampling interval was set to 0.01 seconds.

## 4.2 First Order System with Saturation

When input signal saturation is introduced into a first order linear system, the system's response becomes nonlinear. This is similar to having a limit set on the system input. A block diagram of the nonlinear system is shown in Figure 4.7.

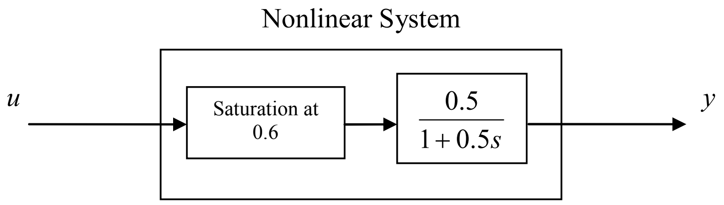


**Figure 4.7** First Order Nonlinear System

Figure 4.8 shows the representation of saturation at 0.6.

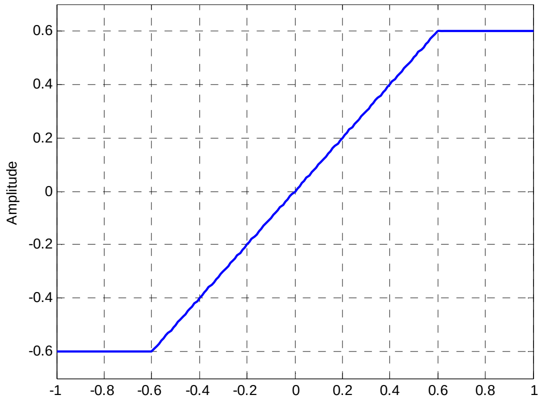
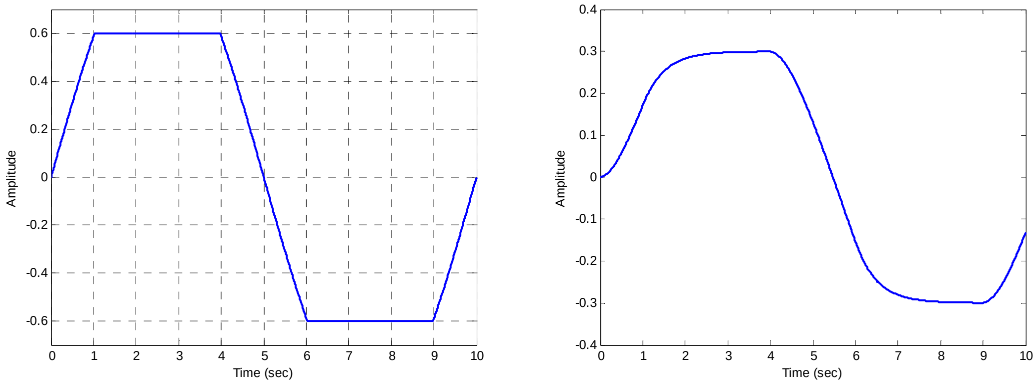


**Figure 4.8** Saturation at 0.6

To demonstrate the effect of saturation, the system was excited by a sinusoidal input as shown in Figure 4.6 (a). The input after saturation is shown in Figure 4.9 (a) and the system output is shown in Figure 4.9 (b).



(a) (b)

**Figure 4.9 (a)** Saturated Input and **(b)** System Output (First Order System)

It was seen that the maximum amplitude of the system output had decreased to 0.3, which was 60% of the maximum amplitude when there was no saturation. This was directly related to the level of saturation applied. As a result, the system response was considered to be nonlinear.

The system was modeled using both a conventional wavelet network and a dynamic wavelet network. In the first case, a feedforward wavelet predictor, shown in Figure 4.10, was used to model the system. The number of wavelet neurons used was 4. The initial values of the wavelet neuron weights, $c_j$, output bias, $a_0$ and direct connections, $a_k$, were initialized to small random values. The dilation and translation parameters were initialized using the heuristic method. The learning rate was set to $1x10^{-4}$ and the momentum coefficient to 0.7. The simulation was run to achieve a normalized mean squared error of $1x10^{-4}$. Normalized MSE provides a universal criterion for measuring WN modeling performance since it is not affected by the magnitude of the system output. The equation for normalized MSE is:

$$MSE_{Normalized} = \frac{MSE}{(y_{\max})^2} \tag{4.7}$$

The statistics are shown in Table 4.1. Plots of the normalized MSE and wavelet network output are shown in Figure 4.11 (a) and (b) respectively.

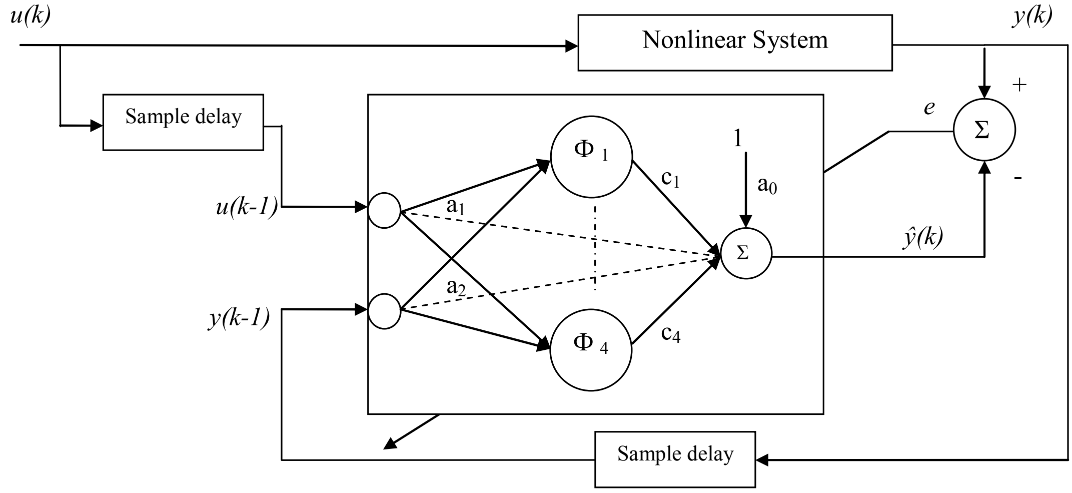
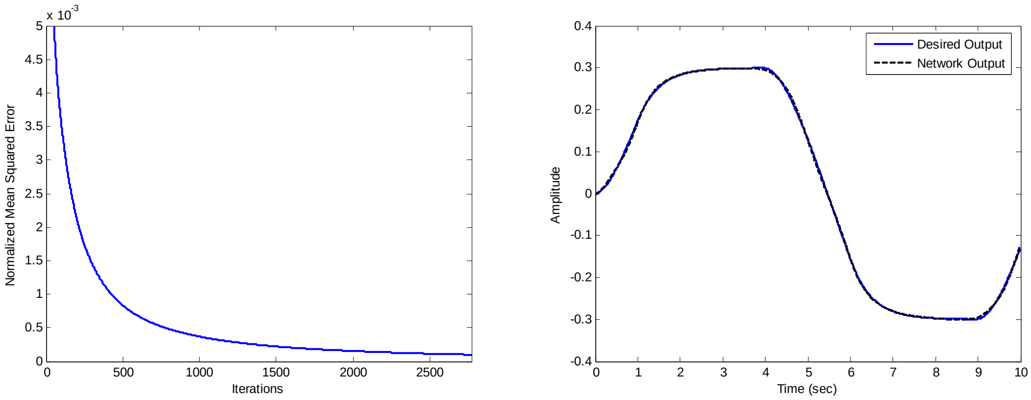


**Figure 4.10** Feedforward Wavelet Predictor for First Order Nonlinear System

Table 4.1

Simulation Statistics for Dynamic Modeling of First Order Nonlinear System using Conventional WN

| Normalized MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1x10^{-4}$ | 2,771 | 52 seconds |



(a) (b)

**Figure 4.11 (a)** Normalized MSE and **(b)** WN Output (Conventional WN)

The simulation results showed that the network was able to model the system by utilizing four wavelets.

If we consider the system model without saturation, the system becomes purely linear and the transfer function is

$$\frac{0.5}{1+0.5s} \tag{4.8}$$

The discrete form of the system's transfer function for a sampling interval of 0.01 seconds and using a zero-order hold is

$$H(z)=\frac{0.009901}{\mathrm{z}-0.9802} \tag{4.9}$$

It was observed that the coefficients of the discrete transfer function were nearly equivalent to the direct connections of the wavelet network when estimated using the least squares method:

$a_0 = 1.6620 \times 10^{-6}$
$a_1 = 0.0100$
$a_2 = 0.9800$

Thus, for a first order linear system, the discrete transfer function could be approximated using the direct connections of the wavelet network as

$$\frac{a_0+a_1}{z-a_2} \tag{4.10}$$

In the second case, the dynamic wavelet network, shown in Figure 4.12, was used to model the system. The number of wavelet neurons used was 4. The initial values of the wavelet neuron weights, $c_j$, output bias, $a_0$ and direct connections, $a_k$, were initialized to small random values. The dilation and translation parameters were initialized using the heuristic method. The linear coefficients, $\beta_0$, $\beta_1$, $\alpha_1$ were initialized using least-squares estimation. Four learning rates were used in training the wavelet network. The learning rate for the wavelet neuron weights, output bias, and direct connections, was set to $1\text{x}10^{-3}$. The learning rate for the dilation and translation parameters was set to $1\text{x}10^{-4}$. The learning rates for the linear coefficients, $\beta_n$ and $\alpha_m$ were set to $1\text{x}10^{-7}$ and $1\text{x}10^{-6}$ respectively. The momentum coefficient was set to 0.5. The simulation was run to achieve a normalized mean squared error of $1\text{x}10^{-4}$. The statistics are shown in Table 4.2. Plots of the normalized MSE and wavelet network output are shown in Figure 4.13 (a) and (b) respectively.

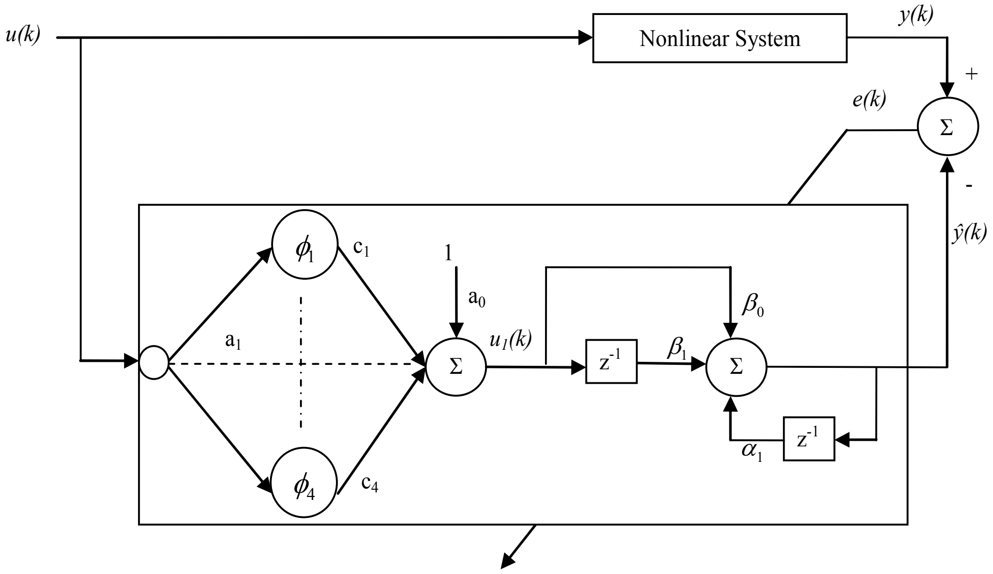
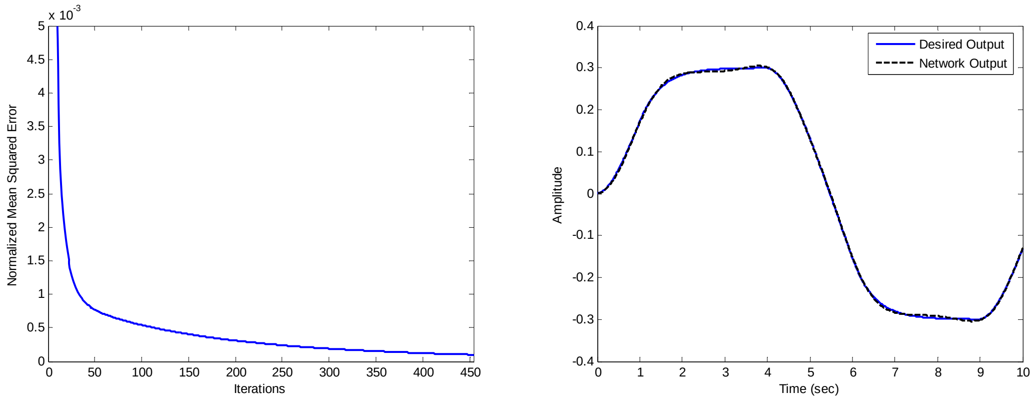


**Figure 4.12** Modeling of First Order Nonlinear System using DWN

Table 4.2

Simulation Statistics for Dynamic Modeling of First Order Nonlinear System using Dynamic WN

| Normalized MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1\text{x}10^{-4}$ | 454 | 84 seconds |



(a) (b)

**Figure 4.13 (a)** Normalized MSE and **(b)** WN Output (Dynamic WN)

The simulation results showed that the DWN was able to model the system within a smaller number of iterations compared to the conventional wavelet network and by utilizing the same number of wavelets. However, the simulation time was longer, due to the difference in training modes between the two schemes. The feedforward wavelet predictor was trained in batch mode, while the DWN was trained in iterative mode.

Furthermore, the intermediate output $u_1$, shown in Figure 4.14 against the saturated input, was used to identify the level of saturation in the system.

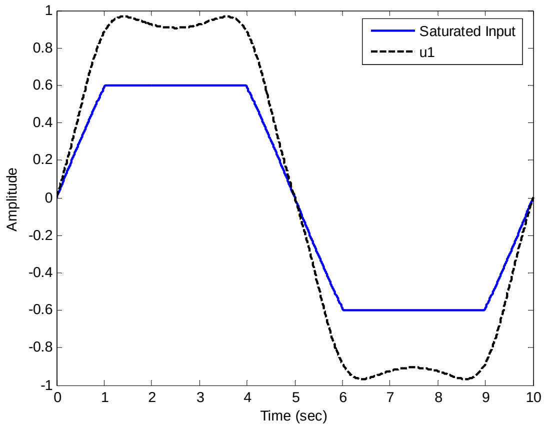


**Figure 4.14** Intermediate Output $u_1$

If we consider the system of Figure 4.7 without saturation being applied, the linear system transfer function is as shown in Equation 4.7 and its DC gain is equal to 0.5. Based on the results obtained, the following equation was derived

$$Saturation = \frac{u_{1_{\max}} \times G}{K} \tag{4.11}$$

Where $u_{1max}$ is the maximum value of $u_1$, $K$ is the DC gain of the linear system, and

$$G = \frac{\beta_0 + \beta_1}{1 - \alpha_1} \tag{4.12}$$

Substituting values into Equation 4.11, we get

$$Saturation = \frac{0.9666 \times 0.3232}{0.5} = 0.6248$$

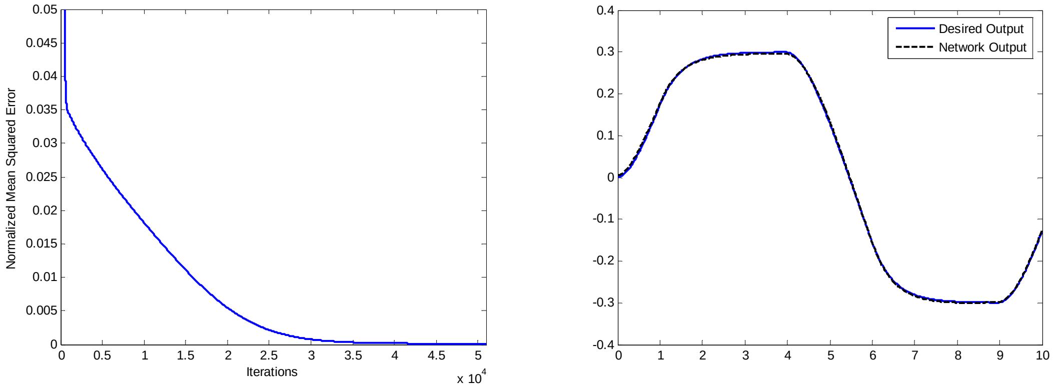
The percentage error of the calculated saturation level was 4.13% of the actual value 0.6.

Finally, the system was modeled using a traditional feedforward neural network based on the backpropagation algorithm. A three-layer network (input layer, hidden layer, output layer) with 4 hidden neurons, employing the sigmoid activation function, was used in the feedforward predictor configuration. The network weights were initialized to small random values. The learning rate was set to $5 \times 10^{-2}$ and the momentum coefficient to 0.5. The simulation was run to achieve a normalized mean squared error of $1 \times 10^{-4}$. The statistics are shown in Table 4.3. Plots of the normalized MSE and neural network output are shown in Figure 4.15 (a) and (b) respectively.

Table 4.3

Simulation Statistics for Dynamic Modeling of First Order Nonlinear System using Feedforward Neural Network

| Normalized MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1 \times 10^{-4}$ | 50,944 | 5 minutes 56 seconds |



(a) (b)

**Figure 4.15 (a)** Normalized MSE and **(b)** NN Output

The simulation results showed that the neural network modeled the system with a much larger number of iterations compared to the conventional wavelet network. The feedforward neural network predictor was also trained in batch mode.

## 4.3 Second Order System

A second order system can be represented as

$$\frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4.13}$$

$K$ is the low frequency (or DC) gain, $\omega_n$ is the corner (or natural) frequency and $\zeta$ is the damping ratio.

In the frequency domain, the system is equivalent to a second order low pass filter. Inputs with frequencies below $\omega_n$ will be multiplied by the system's low frequency gain $K$, while those above $\omega_n$ will be attenuated by the system at a rate of 40 dB/dec. There is a peak in the frequency response at the system's natural frequency $\omega_n$. This peak is dependant upon the value of the damping ratio $\zeta$. The smaller the damping ratio, the higher the gain is at the natural frequency. A Bode plot shows the frequency response of a second order system with different damping ratios in Figure 4.16.

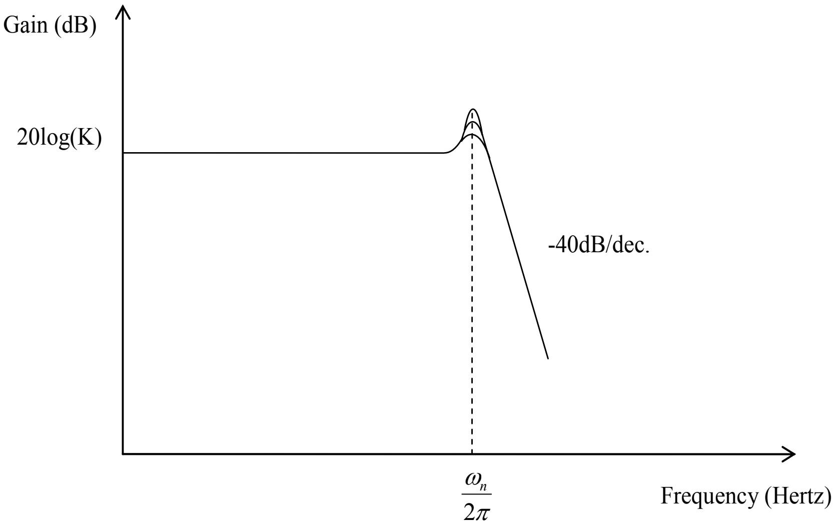


**Figure 4.16** Bode Plot For Second Order System

The system response was examined for the following second order system:

$$\frac{2}{s^2 + 2s + 4} \tag{4.14}$$

Where $K = 0.5$, $\omega_n = 2$ rad/sec and $\zeta = 0.5$.

A step response is shown in Figure 4.17. The system output stabilized at 0.5, which was half the amplitude of the input. The system also displayed an under damped response.

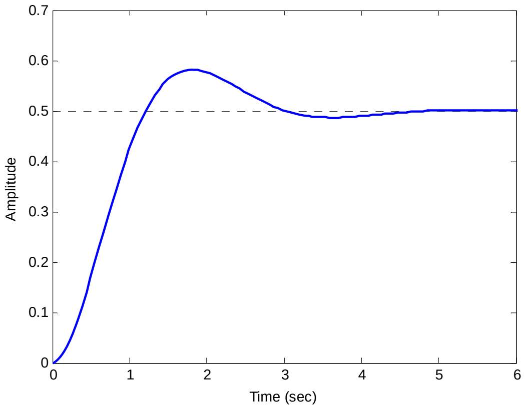
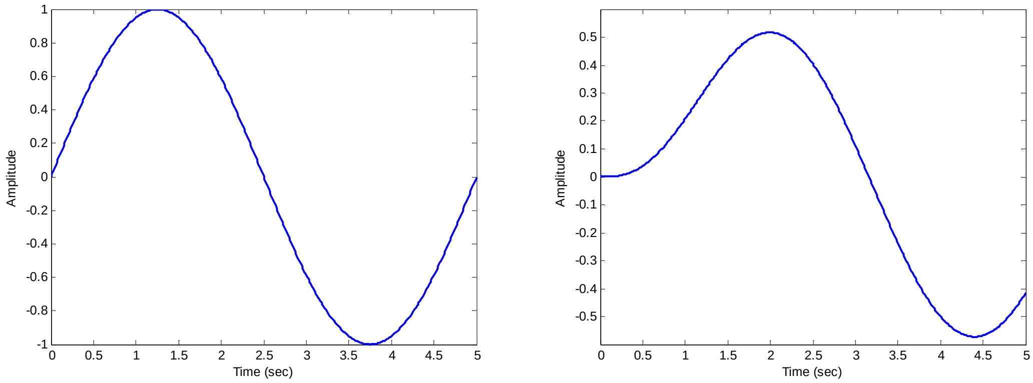


**Figure 4.17** Second Order System Step Response

The second system response was to a sinusoidal signal $u(k)=sin(0.4\pi k)$, shown in Figure 4.18 (a). The system output was a sinusoid with a smaller peak and shifted to the right, shown in Figure 4.18 (b).



(a) (b)

**Figure 4.18** Second Order Linear System **(a)** Input and **(b)** Output

The frequency of the input was 0.2 Hertz and its amplitude was 1, while the output had the same frequency but its amplitude was about half the input, at 0.52, with a shift of about 0.64 seconds or 46.1°. The sampling interval was set to 0.005 seconds.

## 4.4 Second Order System with Saturation

When input signal saturation is introduced into a second order linear system, the system response becomes nonlinear. Figure 4.19 shows the considered second order nonlinear system with input saturation at 0.6.

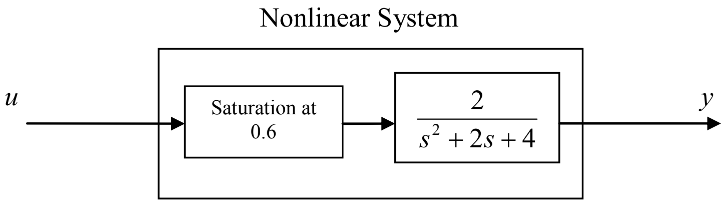
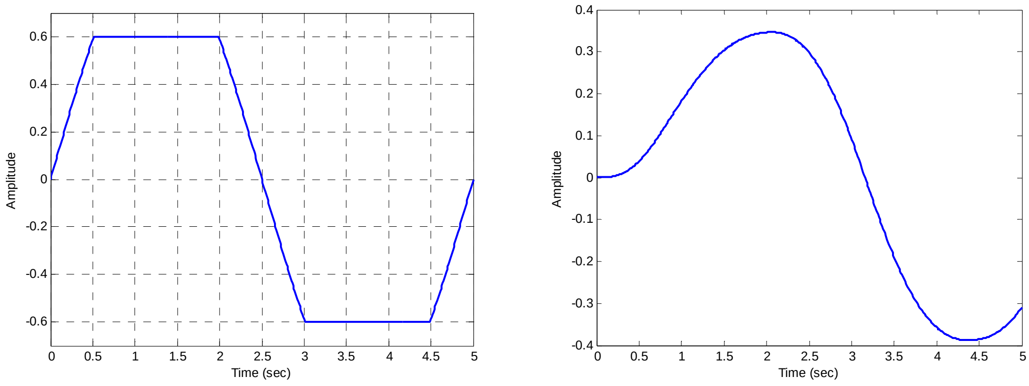


**Figure 4.19** Second Order Nonlinear System

To demonstrate the effect of saturation, the system was excited by the same sinusoidal input shown in Figure 4.18 (a). The input after saturation is shown in Figure 4.20 (a) and the system output is shown in Figure 4.20 (b).



(a) (b)

**Figure 4.20 (a)** Saturated Input and **(b)** System Output (Second Order System)

It was seen that the maximum amplitude of the system output had decreased to 0.35, which was 70% of the maximum amplitude when there was no saturation of the input signal. This was directly related to the level of saturation applied and the damping ratio of the system. As a result, the system response was considered to be nonlinear.

The system was modeled using both a conventional wavelet network and a dynamic wavelet network. In the first case, a feedforward wavelet predictor, shown in Figure 4.21, was used to model the system. The number of wavelet neurons used was 8. The initial values of the wavelet neuron weights, $c_j$, output bias, $a_0$ and direct connections, $a_k$, were initialized to small random values. The dilation and translation parameters were initialized using the heuristic method. The learning rate was set to $1\text{x}10^{-4}$ and the momentum coefficient to 0.6. The simulation was run to achieve a normalized mean squared error of $1\text{x}10^{-4}$. The statistics are shown in Table 4.4. Plots of the normalized MSE and wavelet network output are shown in Figure 4.22 (a) and (b) respectively.

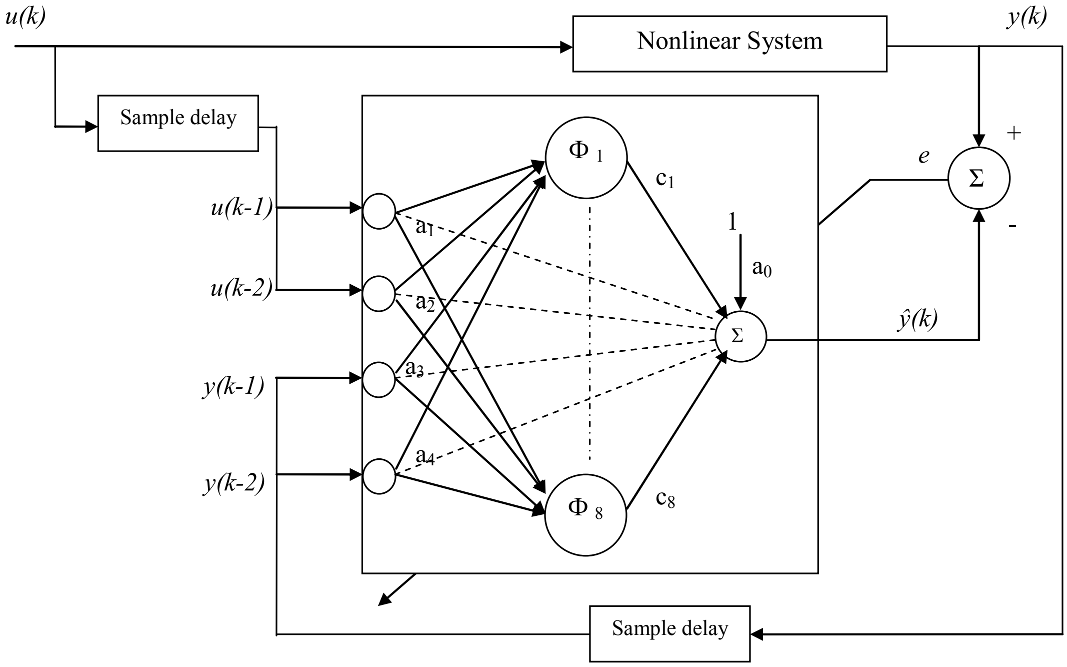
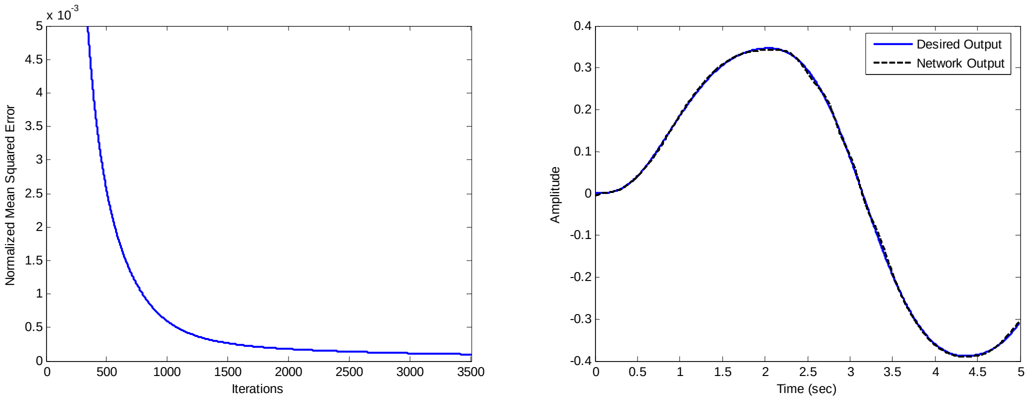


**Figure 4.21** Feedforward Wavelet Predictor for Second Order Nonlinear System

Table 4.4

Simulation Statistics for Dynamic Modeling of Second Order Nonlinear System using Conventional WN

| Normalized MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1\text{x}10^{-4}$ | 3,508 | 9 minutes 7 seconds |



(a) (b)

**Figure 4.22 (a)** Normalized MSE and **(b)** WN Output (Conventional WN)

The simulation results showed that the network was able to model the system by utilizing eight wavelets.

If we consider the system model of Figure 4.19 without saturation, the system is purely linear and the transfer function is

$$\frac{2}{s^2 + 2s + 4} \tag{4.15}$$

The discrete form of the system transfer function for a sampling interval of 0.005 seconds and using a zero-order hold is

$$H(z) = \frac{2.492\text{x}10^{-5} z + 2.483\text{x}10^{-5}}{z^2 - 1.99z + 0.99} \tag{4.16}$$

It was observed that the coefficients of the discrete transfer function were not all similar to the direct connections of the wavelet network when estimated using the least squares method:

$a_0 = 2.5567\text{x}10^{-14}$
$a_1 = 4.9791\text{x}10^{-5}$
$a_2 = -4.1231\text{x}10^{-8}$
$a_3 = 1.9900$
$a_4 = -0.9900$

Only the coefficients in the denominator were similar to $a_3$ and $a_4$. Thus, for a second order linear system, the discrete transfer function can not be approximated using the direct connections of the wavelet network.

In the second case, the dynamic wavelet network, shown in Figure 4.23, was used to model the system. The number of wavelet neurons used was 8. The initial values of the wavelet neuron weights, $c_j$, output bias, $a_0$ and direct connections, $a_k$, were initialized to small random values. The dilation and translation parameters were initialized using the heuristic method. The linear coefficients, $\beta_0$, $\beta_1$, $\beta_2$, $\alpha_1$, $\alpha_2$ were initialized using the least-squares estimation method. Four learning rates were used in training the wavelet network. The learning rate for the wavelet neuron weights, output bias, and direct connections, was set to $1\text{x}10^{-3}$. The learning rate for the dilation and translation parameters was set to $1\text{x}10^{-4}$. The learning rates for the linear coefficients, $\beta_n$ and $\alpha_m$ were set to $1\text{x}10^{-12}$ and $1\text{x}10^{-11}$ respectively. The momentum coefficient was set to 0.6. The simulation was run to achieve a normalized mean squared error of $1\text{x}10^{-4}$. The statistics are shown in Table 4.5. Plots of the normalized MSE and wavelet network output are shown in Figure 4.24 (a) and (b) respectively.

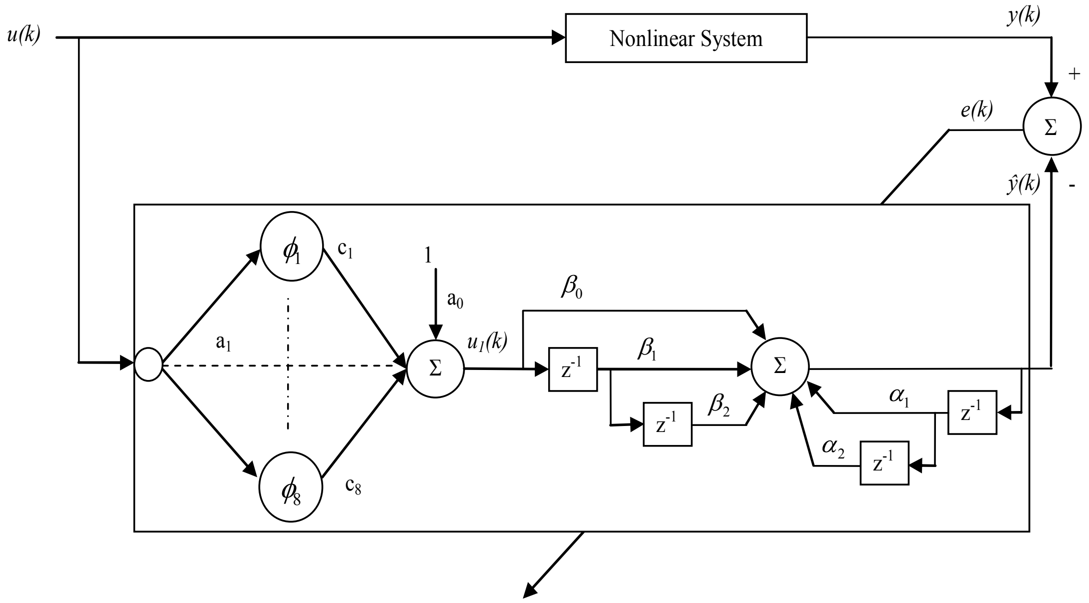
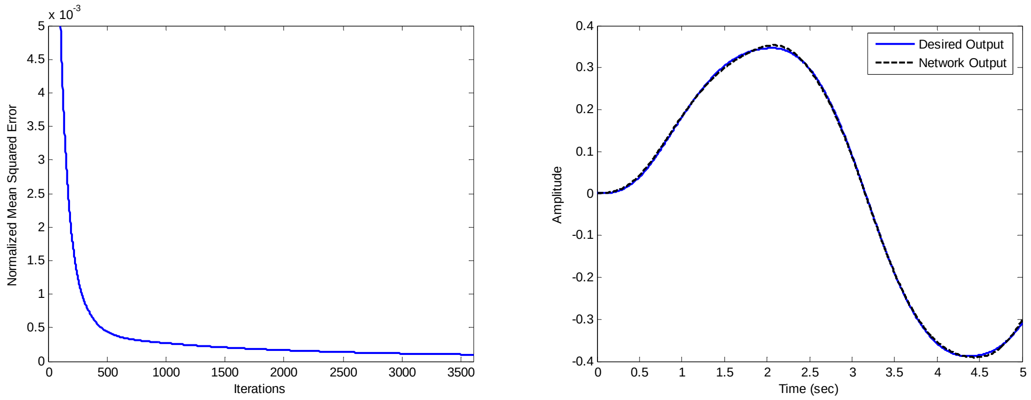


**Figure 4.23** Modeling of Second Order Nonlinear System using DWN

Table 4.5

Simulation Statistics for Dynamic Modeling of Second Order Nonlinear System using Dynamic WN

| Normalized MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1\text{x}10^{-4}$ | 3,602 | 16 minutes 29 seconds |



(a) (b)

**Figure 4.24 (a)** Normalized MSE and **(b)** WN Output (Dynamic WN)

The simulation results showed that the DWN was able to model the system within a slightly larger number of iterations than the conventional wavelet network and by utilizing the same number of wavelets. However, the simulation time was longer, due to the difference in training modes between the two schemes. The feedforward wavelet predictor was trained in batch mode, while the DWN was trained in iterative mode.

Furthermore, the intermediate output $u_1$, shown in Figure 4.25 against the saturated input, was used to identify the level of saturation in the system.

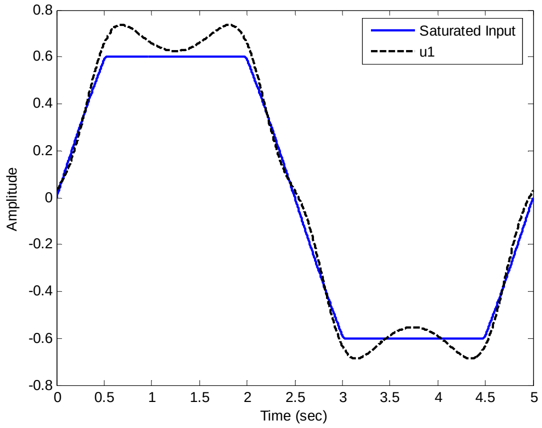


**Figure 4.25** Intermediate Output $u_1$

If we consider the system of Figure 4.19 without saturation being applied, the linear system transfer function is as shown in Equation 4.14 and its DC gain is equal to 0.5. Again, we use derived Equation 4.11 with

$$G = \frac{\beta_0 + \beta_1 + \beta_2}{1 - \alpha_1 - \alpha_2} \tag{4.17}$$

Substituting values into Equation 4.11, we get

$$Saturation = \frac{0.7362 \times 0.4239}{0.5} = 0.6242$$

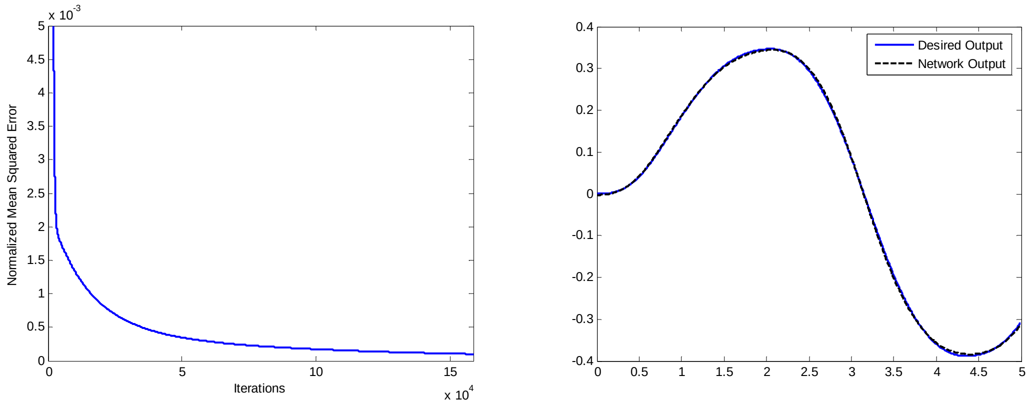
The percentage error of the calculated saturation level was 4.03% of the actual value 0.6.

Finally, the system was modeled using a traditional feedforward neural network based on the backpropagation algorithm. A three-layer network (input layer, hidden layer, output layer) with 8 hidden neurons, employing the sigmoid activation function, was used in the feedforward predictor configuration. The network weights were initialized to small random values. The learning rate was set to 5x$10^{-2}$ and the momentum coefficient to 0.5. The simulation was run to achieve a normalized mean squared error of $1x10^{-4}$. The statistics are shown in Table 4.6. Plots of the normalized MSE and neural network output are shown in Figure 4.26 (a) and (b) respectively.

Table 4.6

Simulation Statistics for Dynamic Modeling of First Order Nonlinear System using Feedforward Neural Network

| Normalized MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1x10^{-4}$ | 158,460 | 46 minutes 7 seconds |



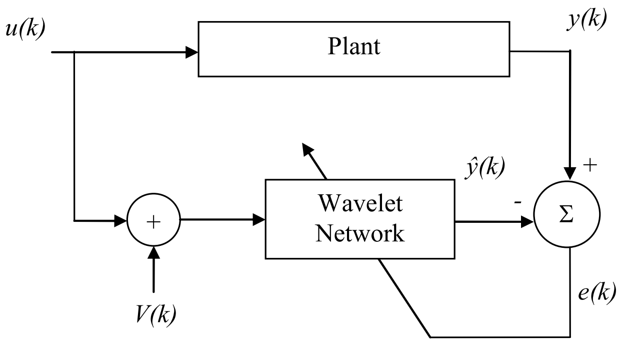
(a) (b)

**Figure 4.26 (a)** Normalized MSE and **(b)** NN Output

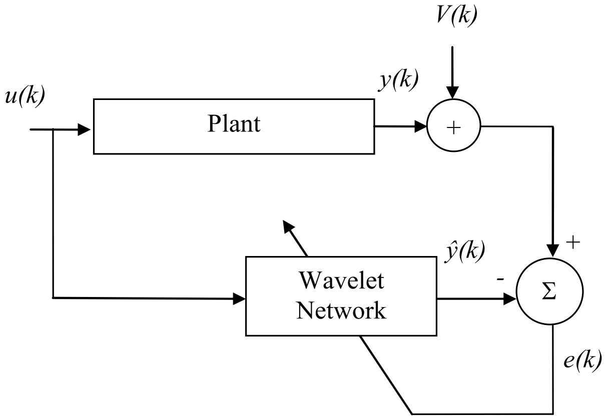
The simulation results showed that the neural network modeled the system with a much larger number of iterations compared to the conventional wavelet network. The feedforward neural network predictor was also trained in batch mode.

## 4.5 Effect of Noise on DWN

In order to study the effect of noise on the dynamic wavelet network, it was trained in the two cases of noise contamination. The first case was input noise training, where the input of the wavelet network model was contaminated by a noise source, as shown in Figure 4.27 (a). The second case was output noise training, where the output of the modeled plant was contaminated by a noise source, as shown in Figure 4.27 (b).



(a)



(b)

**Figure 4.27 (a)** Input Noise Training and **(b)** Output Noise Training

In both cases, three increasing noise levels were successively added. The noise signal was a random noise with a uniform distribution and its mean squared value was calculated as:

$$V_{rms}^2 = \frac{1}{N}\sum_{k=1}^{N} V(k)^2 \tag{4.18}$$

Where $N$ is the number of data samples.

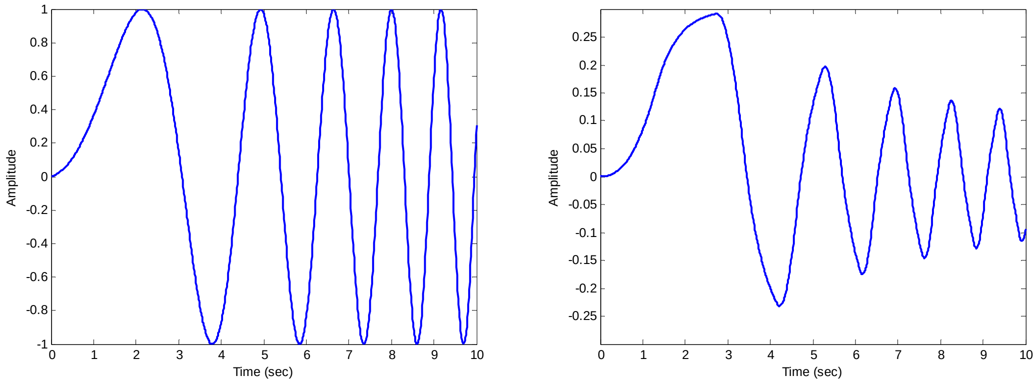
The resulting signal to noise ratio (SNR) in dB of the contaminated input/output signal was then calculated as:

$$SNR = 10\log_{10}\left(\frac{S_{rms}^2}{V_{rms}^2}\right) \tag{4.19}$$

Where $S^2_{rms}$ is the mean squared value of the clean input/output signal.

### 4.5.1 First Order System with Saturation

The DWN was trained on the same system of Figure 4.7. The system was excited by a chirp signal with a frequency range of 0.01 Hertz to 1 Hertz and amplitude of 1, shown in Figure 4.28 (a). The system output is shown in Figure 4.28 (b). The sampling interval was set to 0.01 seconds.



(a) (b)

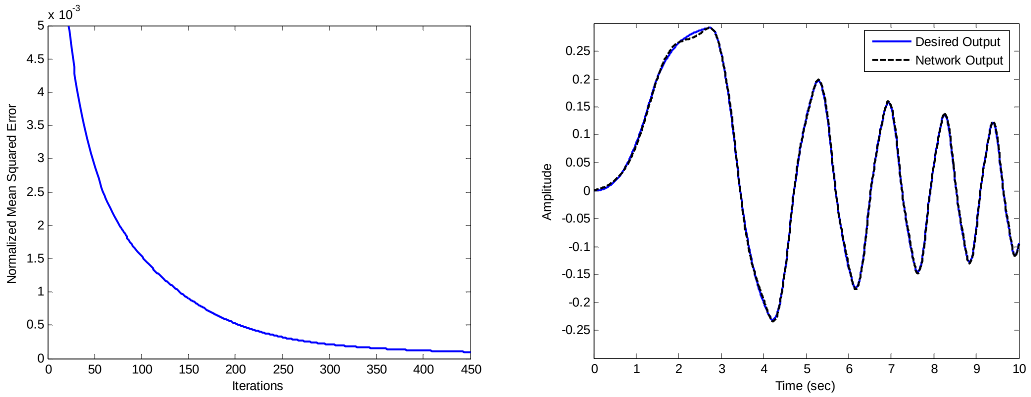
**Figure 4.28 (a)** System Input and **(b)** System Output (First Order Response)

The system was modeled using a dynamic wavelet network with 4 wavelets. The initial values of the wavelet neuron weights, $c_j$, output bias, $a_0$ and direct connections, $a_k$, were initialized to small random values. The dilation and translation parameters were initialized using the heuristic method. The linear coefficients, $\beta_0$, $\beta_1$, $\alpha_1$ were initialized using the least-squares estimation method as explained in Chapter 4. Four learning rates were used in training the wavelet network. The learning rate for the wavelet neuron weights, output bias, and direct connections, was set to $1\text{x}10^{-3}$. The learning rate for the dilation and translation parameters was set to $1\text{x}10^{-3}$. The learning rates for the linear coefficients, $\beta_n$ and $\alpha_m$ were set to $1\text{x}10^{-7}$ and $1\text{x}10^{-6}$ respectively. The momentum coefficient was set to 0.6. The simulation was run to achieve a normalized mean squared error of $1\text{x}10^{-4}$. The statistics are shown in Table 4.7. Plots of the normalized MSE and wavelet network output are shown in Figure 4.29 (a) and (b) respectively. Plots of the wavelet neuron weights, direct connections, dilations, translations and linear coefficients are shown in Figure 4.30 (a), (b), (c), (d) and (e) respectively.

Table 4.7

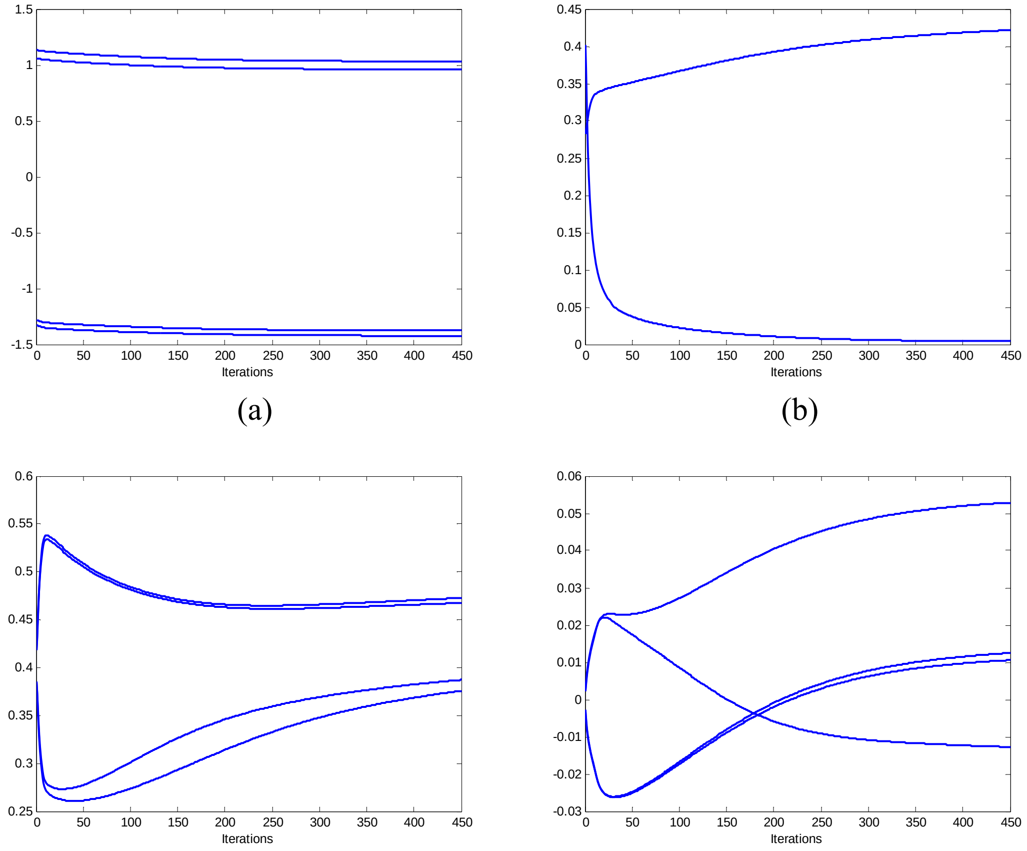
Simulation Statistics for Dynamic Modeling of First Order Nonlinear System without Noise

| Normalized MSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1\text{x}10^{-4}$ | 450 | 69 seconds |



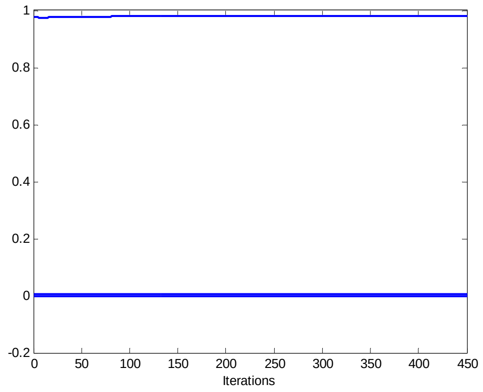
(a) (b)

**Figure 4.29 (a)** Normalized MSE and **(b)** DWN Output (Noise-free)



(a) (b)

(c) (d)



(e)

**Figure 4.30 (a)** Wavelet Neuron Weights, **(b)** Direct Connections, **(c)** Dilations, **(d)** Translations and **(e)** Linear Coefficients

The simulation results showed the network was able to model the system by utilizing four wavelets.

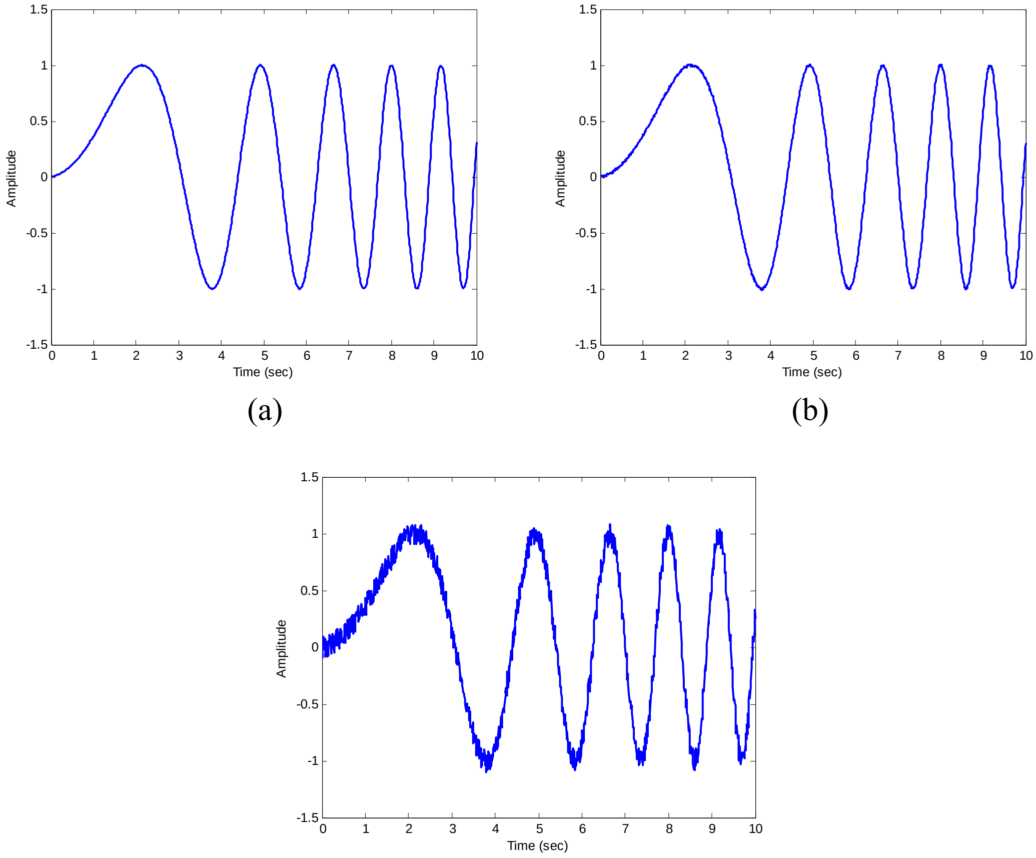
For input noise training, the simulation was run for the same number of iterations as in the noise-free case. The statistics are shown in Table 4.8.

Table 4.8

Simulation Statistics for Dynamic Modeling of First Order Nonlinear System with Input Noise

| SNR (dB) | Number of Iterations | NMSE |
|---|---|---|
| 61.66 | 450 | $9.98 \times 10^{-5}$ |
| 41.66 | 450 | $9.97 \times 10^{-5}$ |
| 21.66 | 450 | $1.30 \times 10^{-4}$ |

Figure 4.31 (a)-(c) shows the network input with successively increasing noise.

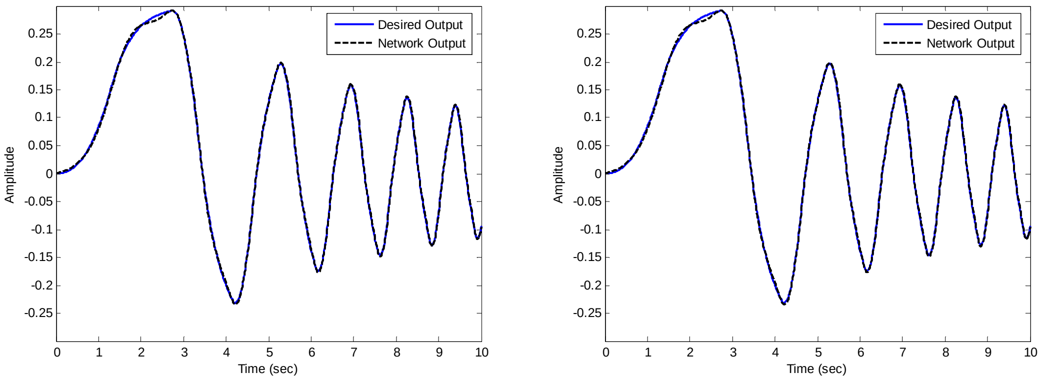


(a) (b)

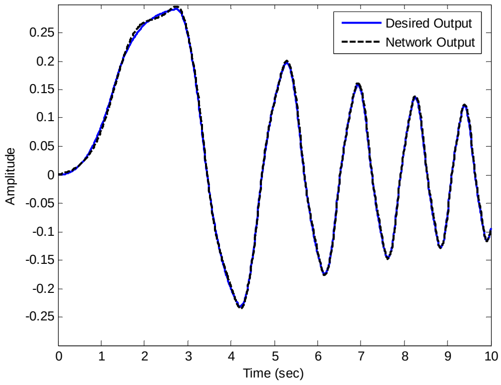
(c)

**Figure 4.31 (a)-(c)** DWN Input with Successively Increasing Noise

Figure 4.32 (a)-(c) shows the corresponding wavelet network output after training.



(a) (b)



(c)

**Figure 4.32 (a)-(c)** DWN Output with Successively Increasing Input Noise

The simulation results showed the DWN performed as good as the noise-free case at the lowest and intermediate noise levels. At the highest noise level the performance was slightly poorer. Thus, input noise had a slight effect on training the DWN on a first order nonlinear system when the noise had a magnitude of 3.15 x$10^{-3}$.

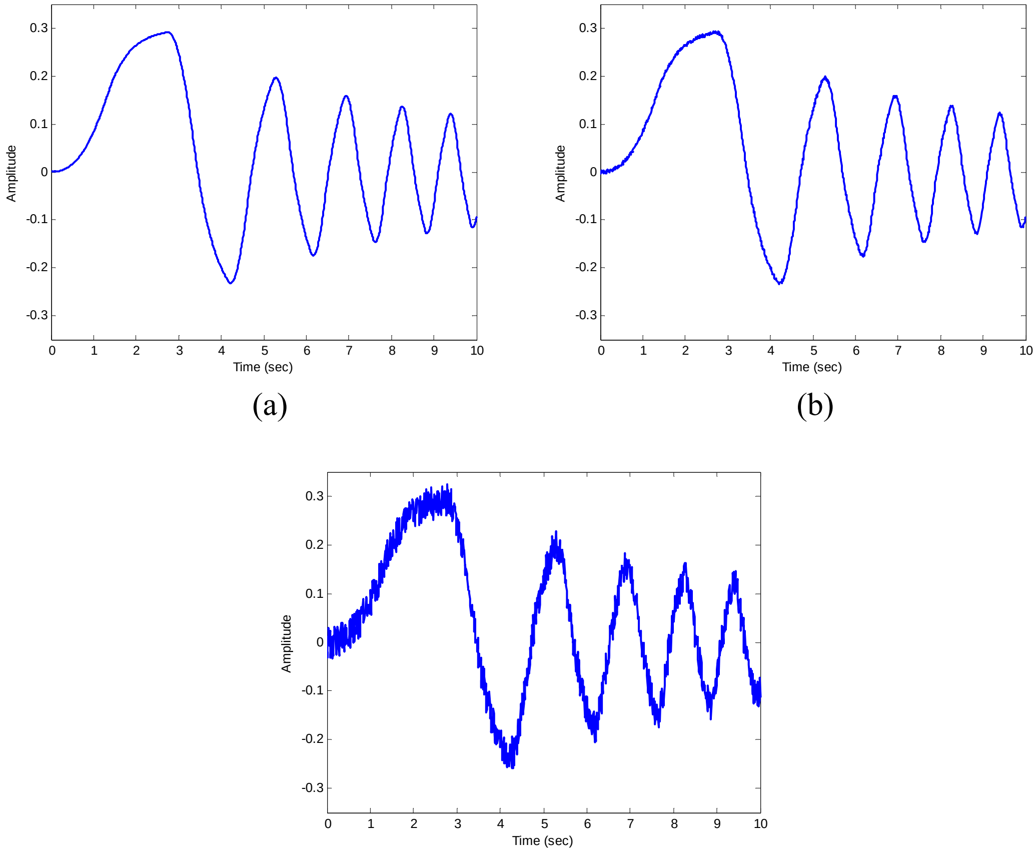
For output noise training, the simulation was run for the same number of iterations as in the noise-free case. The statistics are shown in Table 4.9. The values of NMSE are for the normalized error between the clean plant output and the wavelet network output.

Table 4.9

Simulation Statistics for Dynamic Modeling of First Order Nonlinear System with Output Noise

| SNR (dB) | Number of Iterations | NMSE |
|---|---|---|
| 59.19 | 450 | 9.92 x$10^{-5}$ |
| 39.19 | 450 | 9.65 x$10^{-5}$ |
| 19.19 | 450 | 5.85 x$10^{-5}$ |

Figure 4.33 (a)-(c) shows the plant output with successively increasing noise.

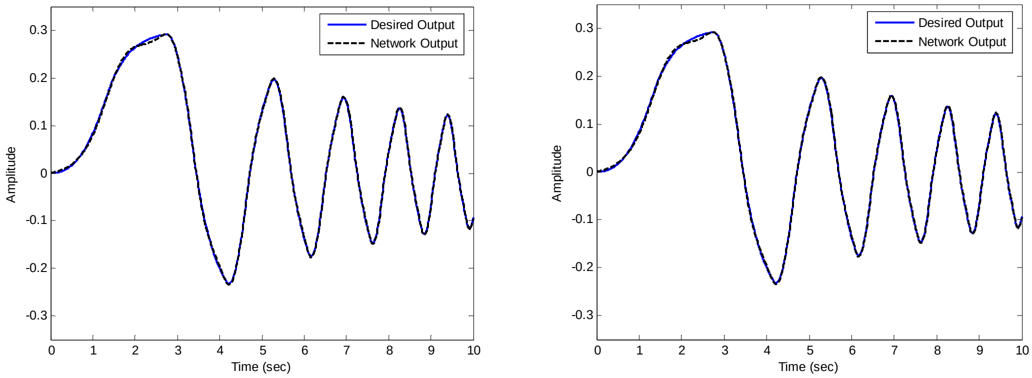


(a) (b)

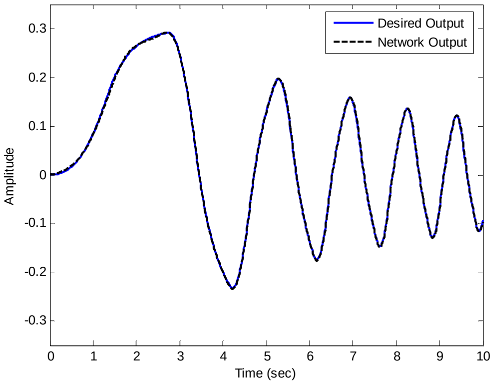
(c)

**Figure 4.33 (a)-(c)** Plant Output with Successively Increasing Noise

Figure 4.34 (a)-(c) shows the corresponding wavelet network output after training.



(a) (b)



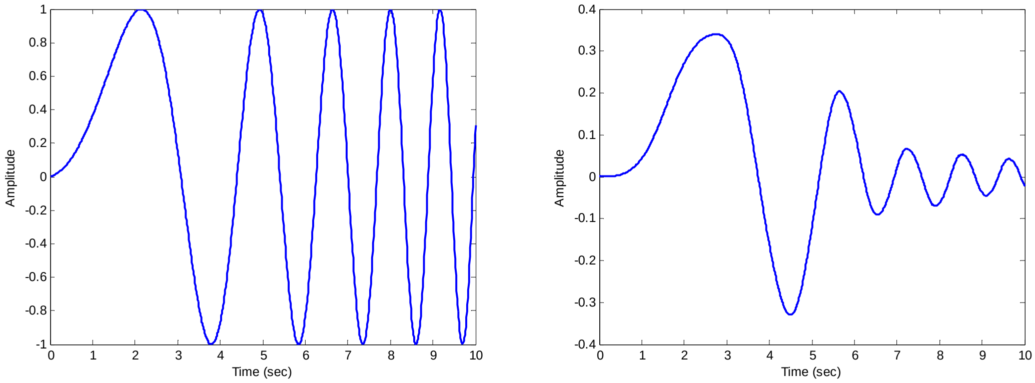
(c)

**Figure 4.34 (a)-(c)** DWN Output with Successively Increasing Output Noise

The simulation results showed the DWN performed as good as the noise-free case at the lowest and intermediate noise levels. At the highest noise level the performance was slightly better. Thus, output noise had no effect on training the DWN on a first order nonlinear system.

### 4.5.2 Second Order System with Saturation

The DWN was trained on the same system of Figure 4.19. The system was excited by a chirp signal with a frequency range of 0.01 Hertz to 1 Hertz and amplitude of 1, shown in Figure 4.35 (a). The system output is shown in Figure 4.35 (b). The sampling interval was set to 0.01 seconds.



(a) (b)

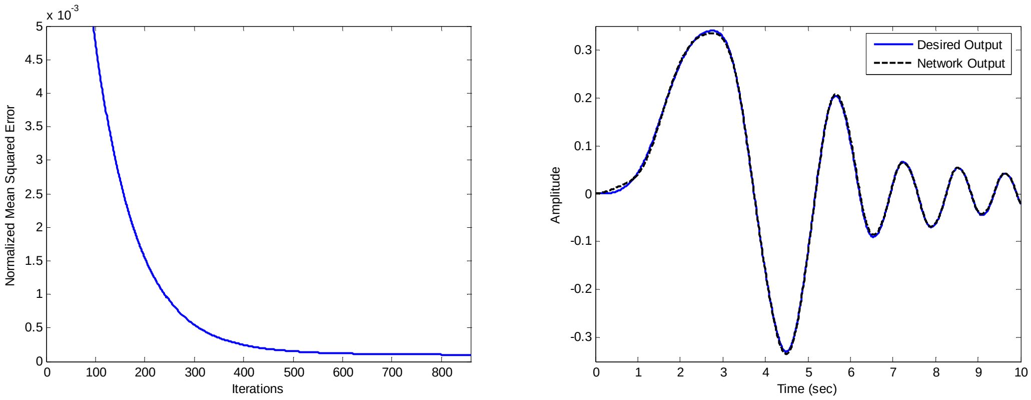
**Figure 4.35 (a)** System Input and **(b)** System Output (Second Order Response)

The system was modeled using a dynamic wavelet network with 4 wavelets. The initial values of the wavelet neuron weights, $c_j$, output bias, $a_0$ and direct connections, $a_k$, were initialized to small random values. The dilation and translation parameters were initialized using the heuristic method. The linear coefficients, $\beta_0$, $\beta_1$, $\beta_2$, $\alpha_1$, $\alpha_2$ were initialized using the least-squares estimation method as explained in Chapter 4. Four learning rates were used in training the wavelet network. The learning rate for the wavelet neuron weights, output bias, and direct connections, was set to $1\text{x}10^{-3}$. The learning rate for the dilation and translation parameters was set to $1\text{x}10^{-4}$. The learning rates for the linear coefficients, $\beta_n$ and $\alpha_m$ were set to $1\text{x}10^{-12}$ and $1\text{x}10^{-11}$ respectively. The momentum coefficient was set to 0.6. The simulation was run to achieve a normalized mean squared error of $1\text{x}10^{-4}$. The statistics are shown in Table 4.10. Plots of the normalized MSE and wavelet network output are shown in Figure 4.36 (a) and (b) respectively. Plots of the wavelet neuron weights, direct connections, dilations, translations and linear coefficients are shown in Figure 4.37 (a), (b), (c), (d) and (e) respectively.

Table 4.10

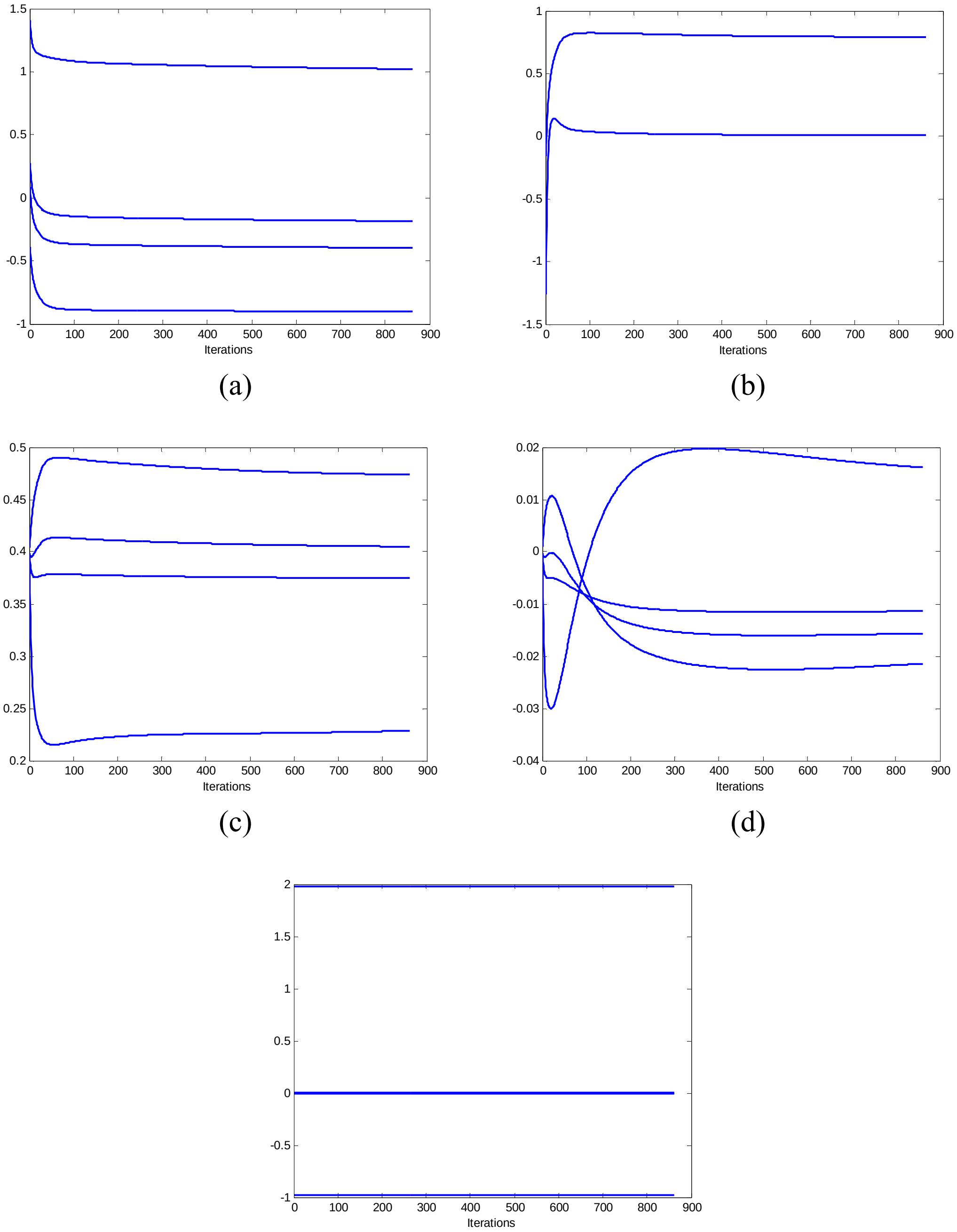
Simulation Statistics for Dynamic Modeling of Second Order Nonlinear System without Noise

| NMSE | Number of Iterations | Simulation Time |
|---|---|---|
| $1\text{x}10^{-4}$ | 861 | 140 seconds |



(a) (b)

**Figure 4.36 (a)** Normalized MSE and **(b)** DWN Output (Noise-free)



(a) (b)

(c) (d)

(e)

**Figure 4.37 (a)** Wavelet Neuron Weights, **(b)** Direct Connections, **(c)** Dilations, **(d)** Translations and **(e)** Linear Coefficients

The simulation results showed the network was able to model the system by utilizing four wavelets.

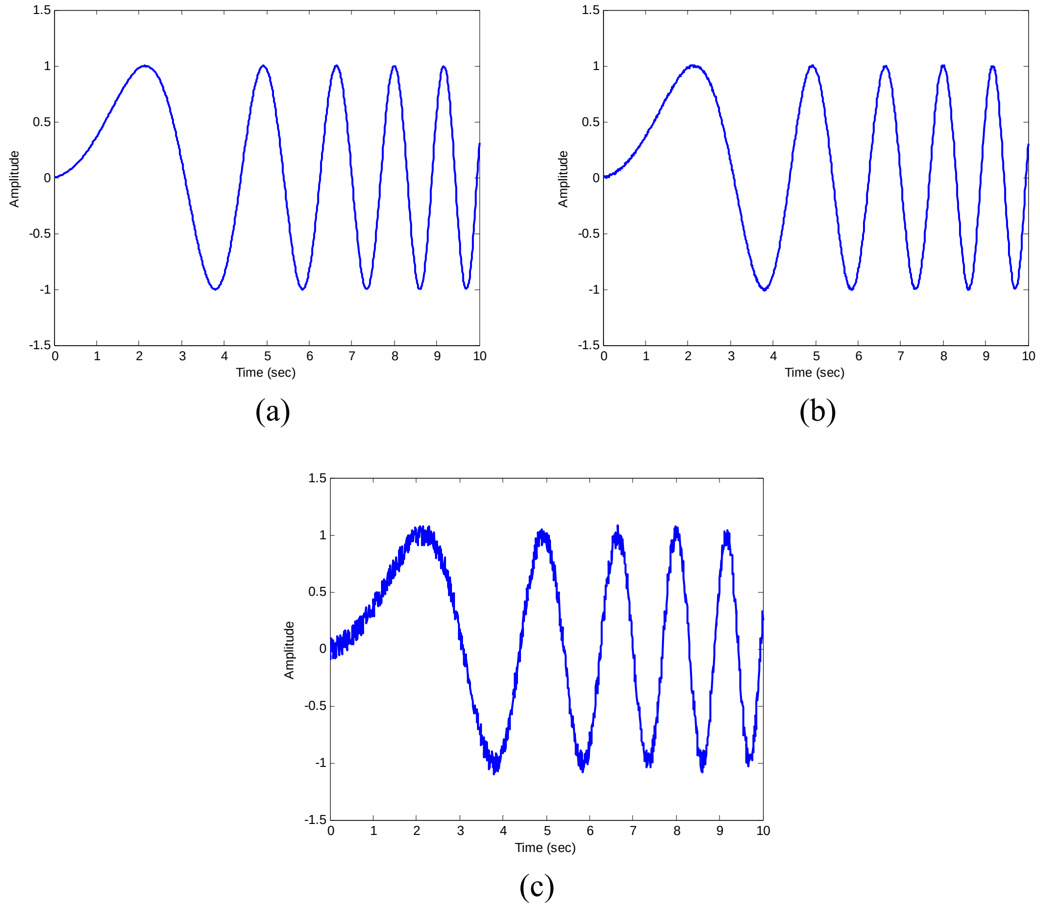
For input noise training, the simulation was run for the same number of iterations as in the noise-free case. The statistics are shown in Table 4.11.

Table 4.11

Simulation Statistics for Dynamic Modeling of Second Order Nonlinear System with Input Noise

| SNR (dB) | Number of Iterations | NMSE |
|---|---|---|
| 61.66 | 861 | 9.95 x$10^{-5}$ |
| 41.66 | 861 | 9.11 x$10^{-5}$ |
| 21.66 | 861 | 8.54 x$10^{-5}$ |

Figure 4.38 (a)-(c) shows the network input with successively increasing noise.



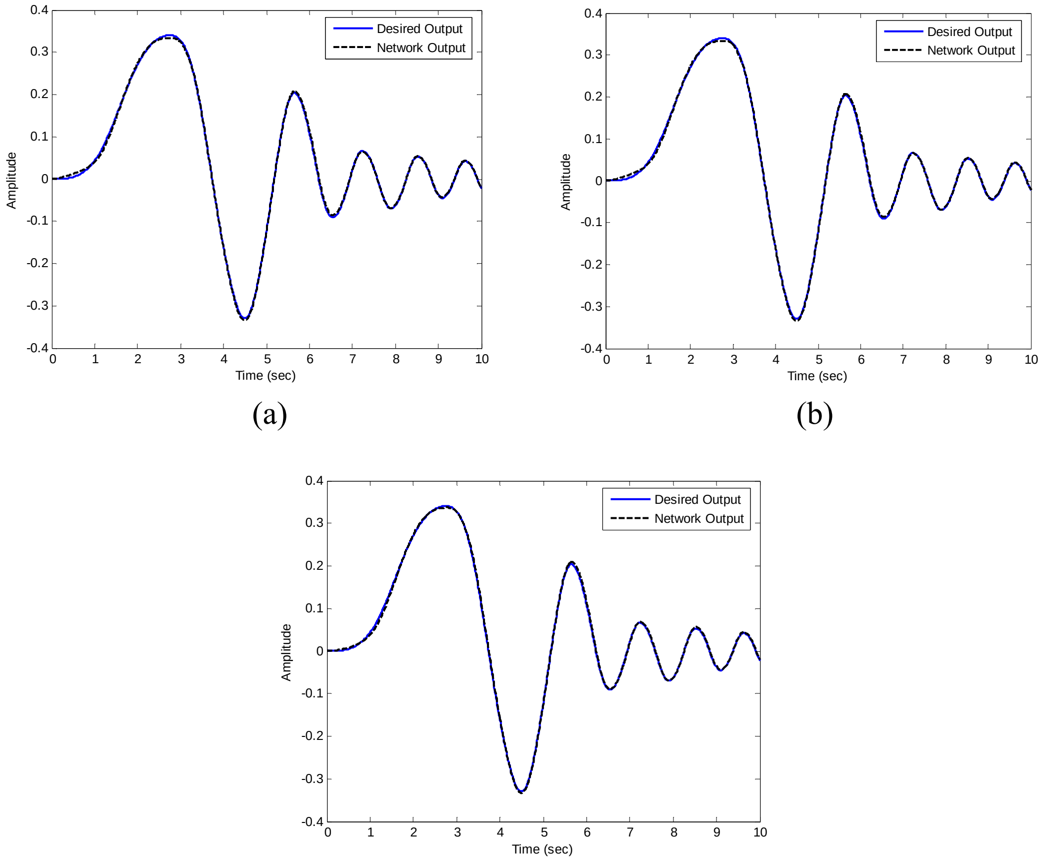
(a)

(b)

(c)

**Figure 4.38 (a)-(c)** DWN Input with Successively Increasing Noise

Figure 4.39 (a)-(c) shows the corresponding wavelet network output after training.



(a) (b)

(c)

**Figure 4.39 (a)-(c)** DWN Output with Successively Increasing Input Noise

The simulation results showed the DWN performed as good as the noise-free case at all the noise levels. Thus, input noise had no effect on the training of a DWN on a second order nonlinear system.

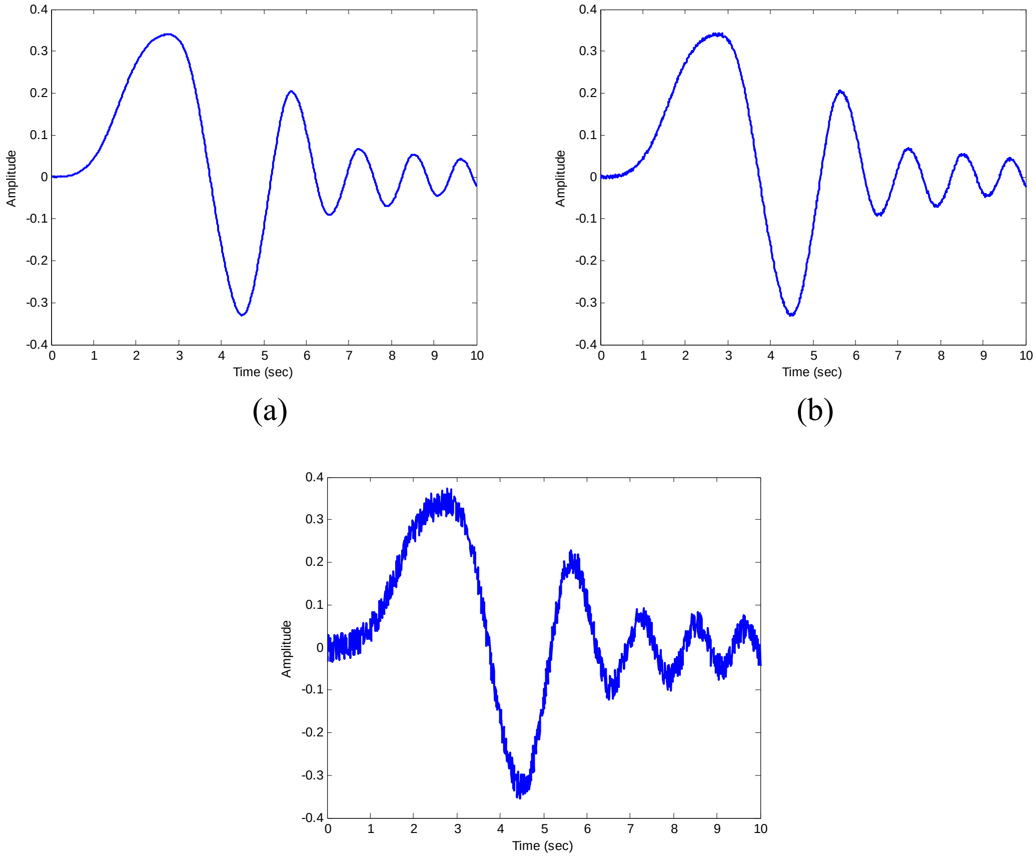
For output noise training, the simulation was run for the same number of iterations as in the noise-free case. The statistics are shown in Table 4.12. The values of NMSE are for the normalized error between the clean plant output and the wavelet network output.

Table 4.12

Simulation Statistics for Dynamic Modeling of Second Order Nonlinear System with Output Noise

| SNR (dB) | Number of Iterations | NMSE |
|---|---|---|
| 58.74 | 861 | $1.39\ \text{x}10^{-2}$ |
| 38.74 | 861 | $8.07\ \text{x}10^{-2}$ |
| 18.74 | 861 | $4.48\ \text{x}10^{-2}$ |

Figure 4.40 (a)-(c) shows the plant output with successively increasing noise.



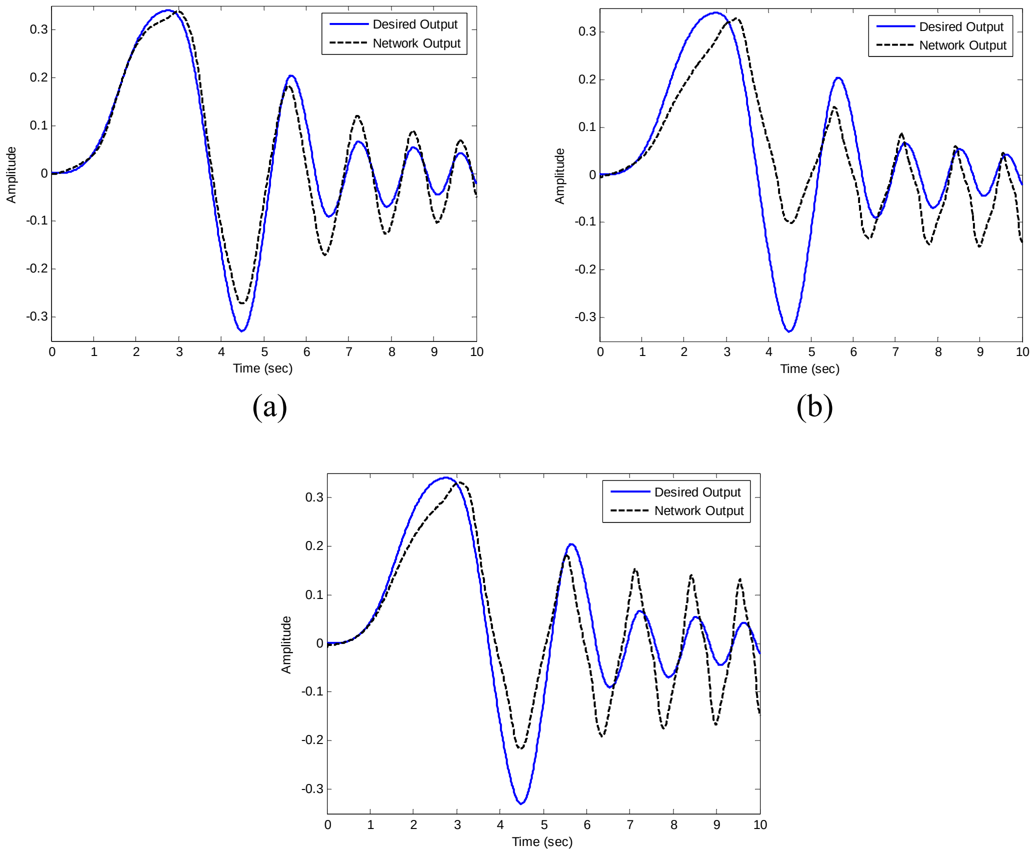
(a)

(b)

(c)

**Figure 4.40 (a)-(c)** Plant Output with Successively Increasing Noise

Figure 4.41 (a)-(c) shows the corresponding wavelet network output after training.



(a) (b)

(c)

**Figure 4.41 (a)-(c)** DWN Output with Successively Increasing Output Noise

The simulation results showed the DWN performed worse than the noise-free case at all the noise levels. Thus, output noise had a major effect on the training of a DWN on a second order nonlinear system.

## 4.6 Conclusion

When wavelet networks are used in the framework of dynamic modeling, network training is performed in a recursive manner. There are two configurations used to train conventional wavelet networks, namely the feedforward wavelet predictor and the feedback wavelet predictor. In the feedforward wavelet predictor configuration, the wavelet network input consists of delayed samples of the system input and the system output. In the feedback wavelet predictor configuration, the wavelet network input consists of delayed samples of the system input and the wavelet network output. The number of inputs to the wavelet network increases with the order of the system being modeled. Dynamic modeling with the DWN involves adjusting the size of the recursive filter section to accommodate the order of the system being modeled, as opposed to adjusting the number of inputs to the wavelet network with conventional dynamic modeling methods. This method enables the user to model higher order systems without the need for using multidimensional wavelets and thus avoiding the added complexity to the network.

The DWN was shown to be as effective as the conventional wavelet network in the dynamic modeling of first and second order nonlinear systems. The advantage of the DWN was shown in the case of input saturation, where the intermediate output $u_1$ was used to identify the level of saturation. The disadvantage of the DWN was in the longer simulation times compared to the feedforward wavelet predictor. When compared to traditional feedforward neural networks, both wavelet network models were shown to be superior.

Dynamic modeling of first and second order nonlinear systems with the DWN was shown to be unaffected by input noise resulting in SNR values of 61.66 dB and 41.66 dB. However, with input noise resulting in a SNR of 21.66 dB, only dynamic modeling of a second order nonlinear system was unaffected, while it slightly affected dynamic modeling of a first order nonlinear system. Nevertheless, the result can be improved by increasing the training time of the DWN. In the case of output noise, dynamic modeling of a first order nonlinear system was shown to be unaffected by output noise resulting in SNR values of 59.19 dB, 39.19 dB and 19.19 dB, while dynamic modeling of a second order nonlinear system was greatly affected by output noise resulting in SNR values of 58.74 dB, 38.74 dB and 18.74 dB. Thus, the DWN was found to be highly sensitive to output noise in the dynamic modeling of second order nonlinear systems. In order to avoid this outcome, the output data must be filtered prior to modeling the system.

# CHAPTER 5

# ADAPTIVE PID-DWN CONTROLLER

The Proportional-Integral-Derivative (PID) controller is one of the simplest of the traditional feedback control schemes. Nevertheless, the linear PID algorithm might have difficulties dealing with processes with complex dynamics such as those with large dead time, inverse response and highly nonlinear characteristics. To improve the control performance, an adaptive PID algorithm is proposed by utilizing the simple PID controller structure based on self-tuning schemes of the proportional, integral and derivative parameters. The basic idea of PID control is that the control action $u$(k) should be proportional to the error, the integral of the error over time, and the temporal derivative of the error. However, limited performance can be a disadvantage of the linear PID controller i.e., the PD action is used to accelerate the speed of the response and the PI mode is used to eliminate the steady-state offset, which sometimes can cause excessive overshoot due to direct implementation of the integral action, etc. The proposed adaptive variable PID controller can help to improve the limited performance of the static PID controller dealing with conflict in nature between static accuracy (steady-state error) and dynamic responsiveness (speed of response).

## 5.1 Structure and Algorithm

Figure 5.1 depicts the block diagram of the network topology based on the PID controller paradigm.

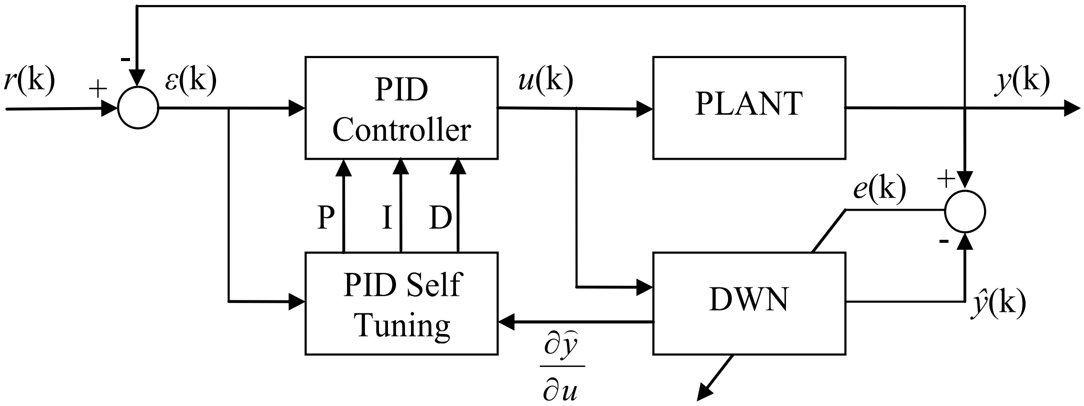


**Figure 5.1** Adaptive PID-DWN Controller

Several tuning components determine the contribution of the weights of the error that suits a cost function

$$J = \frac{1}{2}\sum_{k=1}^{T}\left(r(k) - \hat{y}(k)\right)^2 = \frac{1}{2}\sum_{k=1}^{T}\hat{e}(k)^2 \tag{5.1}$$

Where *r(k)* is the desired set point and *ŷ(k)* is the wavelet network output. The digital PID controller can be expressed in discrete time as follows [15, 16, 17]:

$$u(k) = u(k-1) + P[\varepsilon(k) - \varepsilon(k-1)] + I\varepsilon(k) + D[\varepsilon(k) - 2\varepsilon(k-1) + \varepsilon(k-2)] \tag{5.2}$$

Where *P*, *I* and *D* are proportional, integral and derivative terms, *u(k)* is the plant input at *kT*, where *T* is the sampling interval, and

$$\varepsilon(k) = r(k) - y(k) \tag{5.3}$$

The *P*, *I* and *D* parameters are considered part of the cost function *J*, and are optimized and updated according to the gradient descent algorithm. The PID controller parameters are represented by the set *θ*, i.e. *θ* = {*P*, *I*, *D*}. The partial derivative of the cost function with respect to *θ* is:

$$\frac{\partial J}{\partial \theta} = -\sum_{k=1}^{T}\hat{e}(k)\frac{\partial \hat{y}(k)}{\partial \theta} \tag{5.4}$$

From Equation 4.3, the derivative equations of the above vector with respect to the DWN output, *ŷ(k)*, are:

$$\frac{\partial \hat{y}(k)}{\partial P} = \sum_{n=0}^{N}\beta_n \frac{\partial u_1(k-n)}{\partial P} + \sum_{m=1}^{N}\alpha_m \frac{\partial \hat{y}(k-m)}{\partial P} \tag{5.5}$$

$$\frac{\partial \hat{y}(k)}{\partial I} = \sum_{n=0}^{N}\beta_n \frac{\partial u_1(k-n)}{\partial I} + \sum_{m=1}^{N}\alpha_m \frac{\partial \hat{y}(k-m)}{\partial I} \tag{5.6}$$

$$\frac{\partial \hat{y}(k)}{\partial D} = \sum_{n=0}^{N} \beta_n \frac{\partial u_1(k-n)}{\partial D} + \sum_{m=1}^{N} \alpha_m \frac{\partial \hat{y}(k-m)}{\partial D} \tag{5.7}$$

From Equation 4.4, the derivative equations of the above vector with respect to the DWN intermediate output, $u_1(k)$, are:

$$\frac{\partial u_1(k)}{\partial P} = \frac{\partial u(k)}{\partial P}\left[a_1 + \sum_{j=1}^{Nw} c_j \left(z_j(k)^2 - 1\right) e^{\frac{-z_j(k)^2}{2}} \left(\frac{1}{d_j}\right)\right] \tag{5.8}$$

$$\frac{\partial u_1(k)}{\partial I} = \frac{\partial u(k)}{\partial I}\left[a_1 + \sum_{j=1}^{Nw} c_j \left(z_j(k)^2 - 1\right) e^{\frac{-z_j(k)^2}{2}} \left(\frac{1}{d_j}\right)\right] \tag{5.9}$$

$$\frac{\partial u_1(k)}{\partial D} = \frac{\partial u(k)}{\partial D}\left[a_1 + \sum_{j=1}^{Nw} c_j \left(z_j(k)^2 - 1\right) e^{\frac{-z_j(k)^2}{2}} \left(\frac{1}{d_j}\right)\right] \tag{5.10}$$

Where $\phi_j(k) = -z_j(k) e^{-\frac{z_j(k)^2}{2}}$, and $z_j(k) = \frac{u(k) - m_j}{d_j}$

The derivative equations of the above vector with respect to the plant input, $u(k)$, are:

$$\frac{\partial u(k)}{\partial P} = \frac{\partial u(k-1)}{\partial P} + \varepsilon(k) - \varepsilon(k-1) \tag{5.11}$$

$$\frac{\partial u(k)}{\partial I} = \frac{\partial u(k-1)}{\partial I} + \varepsilon(k) \tag{5.12}$$

$$\frac{\partial u(k)}{\partial D} = \frac{\partial u(k-1)}{\partial D} + \varepsilon(k) - 2\varepsilon(k-1) + \varepsilon(k-2) \tag{5.13}$$

The controller parameters are updated every sampling interval by

$$\Delta P(k) = -\mu_P \frac{\partial J}{\partial P} \tag{5.14}$$

$$\Delta I(k) = -\mu_I \frac{\partial J}{\partial I} \tag{5.15}$$

$$\Delta D(k) = -\mu_D \frac{\partial J}{\partial D} \tag{5.16}$$

Where $\mu_P$, $\mu_I$, $\mu_D$ are the proportional, integral and derivative parameter learning rates, respectively.

## 5.2 Simulation Analysis

The proposed adaptive PID-DWN controller was implemented in Matlab/Simulink environment and utilized in the control of two nonlinear dynamic systems. The first was a first order system with saturation and the second was a second order system with saturation. The DWN models in both cases were trained on the nonlinear systems off-line. Step and sine inputs were used to test the effectiveness of the proposed control system. The effects of input noise and output noise were also investigated.

### 5.2.1 Simulink Model

The Simulink model of the control system is shown in Figure 5.2. The contents of the DWN + PID Tuning block and the PID Controller block are shown in Figure 5.3 and 5.4, respectively.

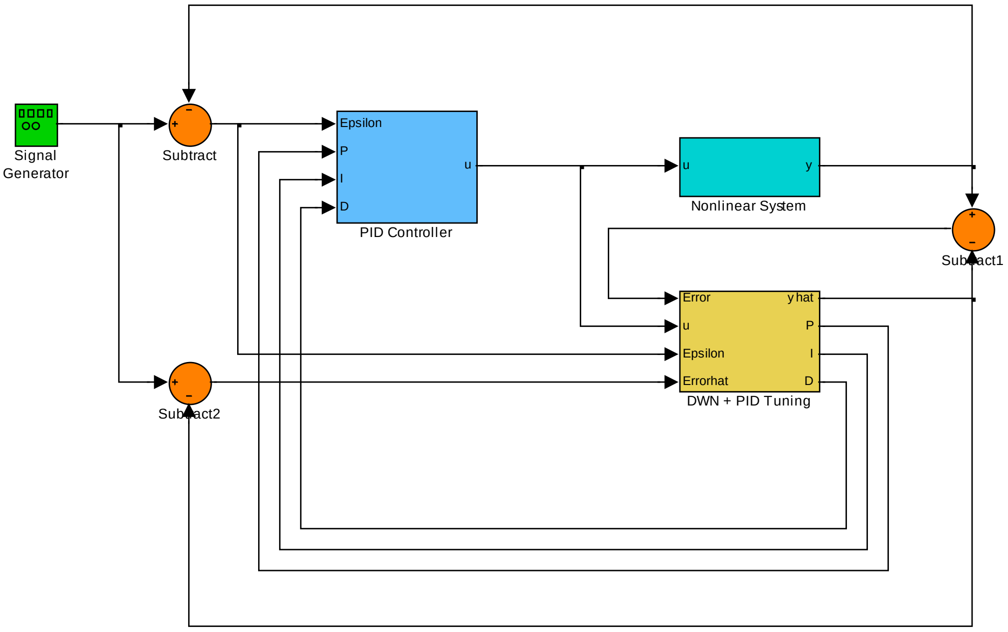


**Figure 5.2** Adaptive PID-DWN Control System (Simulink Model)

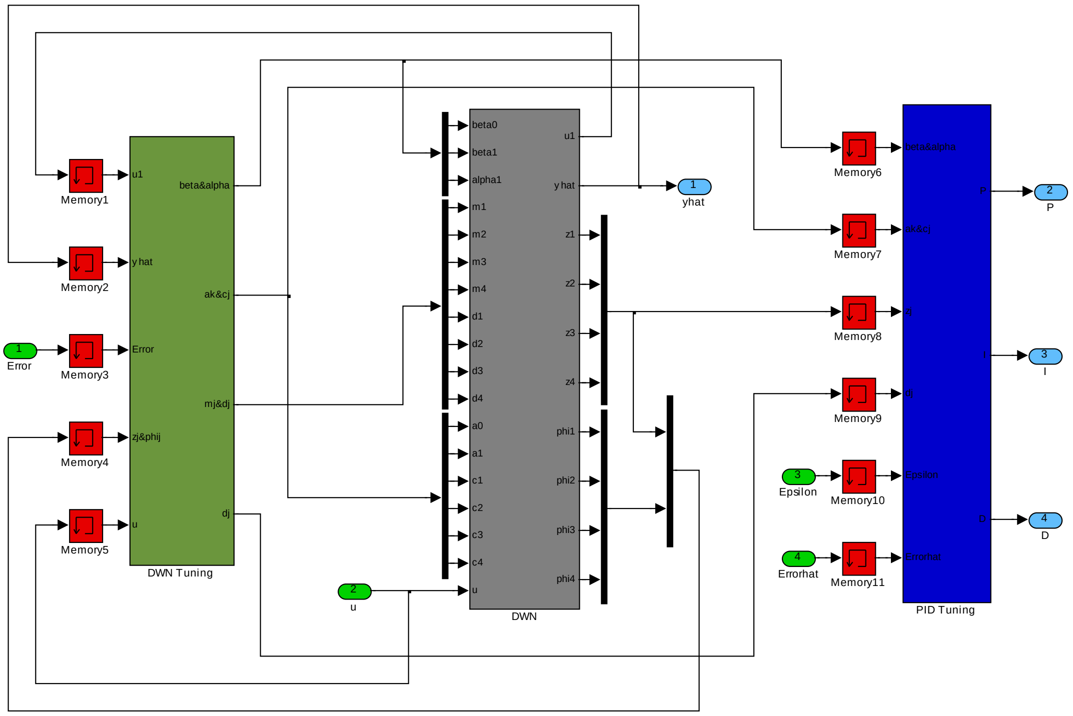


**Figure 5.3** DWN + PID Tuning Block

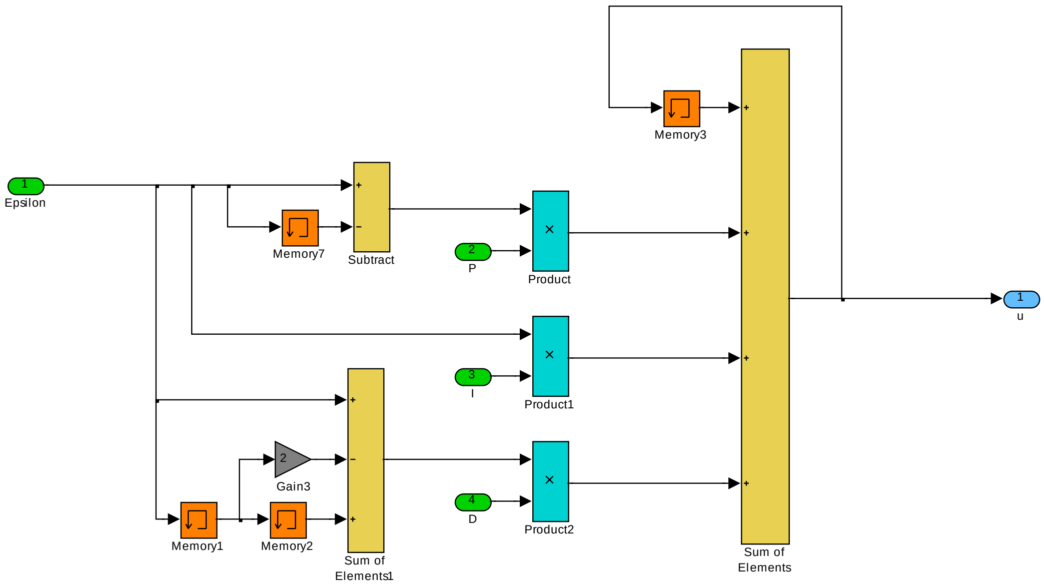


**Figure 5.4** PID Controller Block

### 5.2.2 First Order System with Saturation

The same system studied in subsection 4.2, shown in Figure 5.5, was to be controlled by the proposed PID-DWN controller.

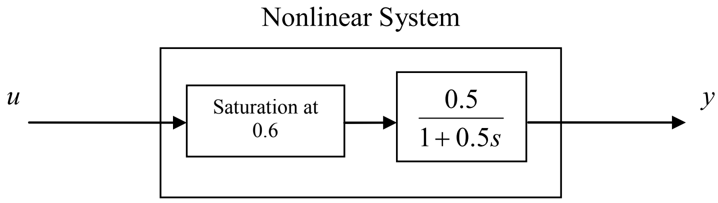
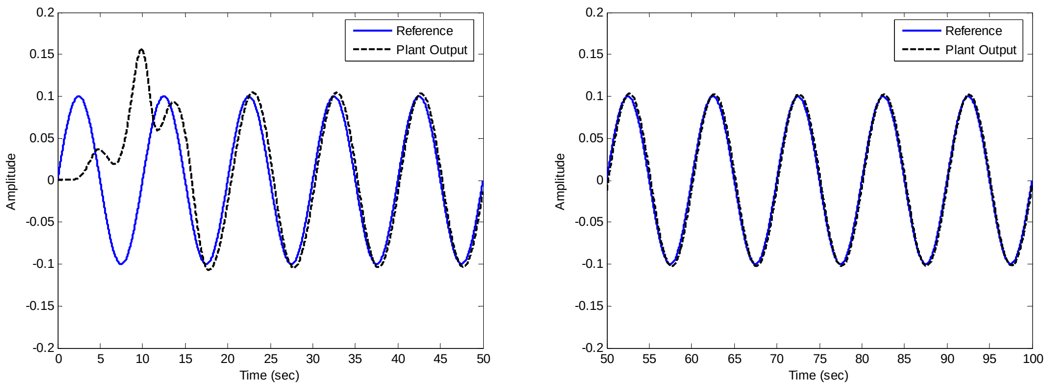


**Figure 5.5** First Order System with Saturation

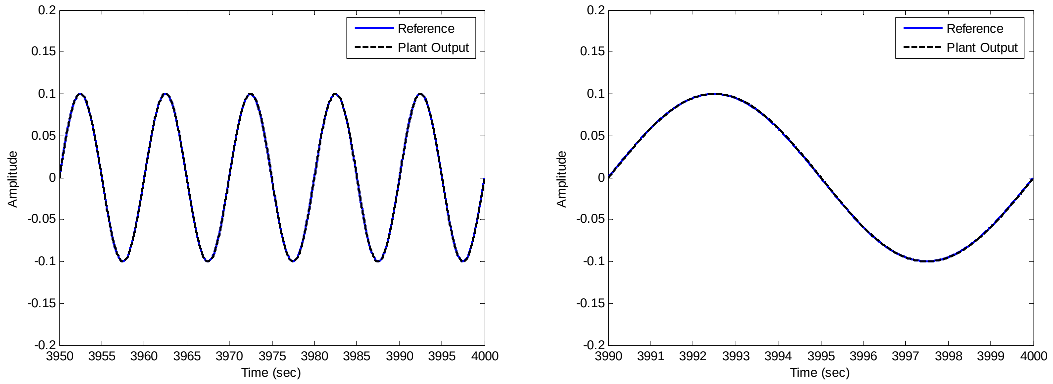
The DWN was first trained on the nonlinear system off-line. In order for the DWN to be properly trained and learn the system dynamics, a chirp signal, as in subsection 4.5.1, was used for the identification. The system was modeled, using the wavelet network, initialization method and learning rates, as in subsection 4.5.1.

The PID-DWN controller was tested in tracking two reference signals: a sinusoid and a step.

The sinusoid signal had a frequency of 0.1Hz and maximum amplitude of 0.1. The initial values of the PID terms were $K_P = 0$, $K_I = 0$, $K_D = 0$.The learning rates for the wavelet network weights, dilations, translations and linear coefficients were set to $3x10^{-6}$. The momentum coefficient was set to 0. The proportional, integral and derivative learning rates were set to $1x10^{-3}$, $1x10^{-4}$ and $1x10^{-8}$, respectively. The sampling interval was set to 0.01 seconds. The simulation was run for 400 cycles. Figure 5.6 (a)-(c) shows the tracking performance of the controller in 5 cycle long plots. Figure 5.6 (d) shows the final cycle in the simulation.



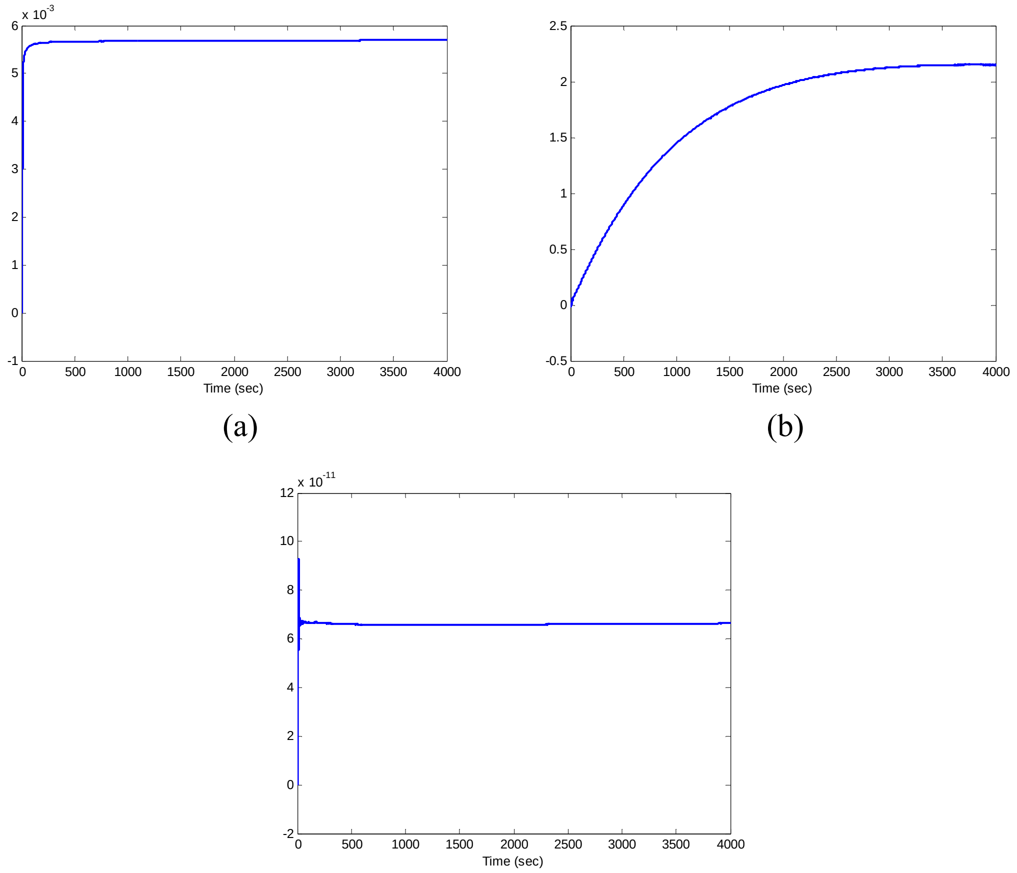
(a) (b)



(c) (d)

**Figure 5.6 (a)-(c)** Tracking Plots and **(d)** Final Cycle (Sine Ref.)

The final values of the PID terms were $K_P = 0.0057$, $K_I = 2.149$, $K_D = 6.64 \times 10^{-11}$. Figure 5.7 (a)-(c) shows the plots of the PID terms during the simulation.



(a) (b)

(c)

**Figure 5.7 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

Figure 5.8 shows the plot of the derivative term for the first ten cycles.

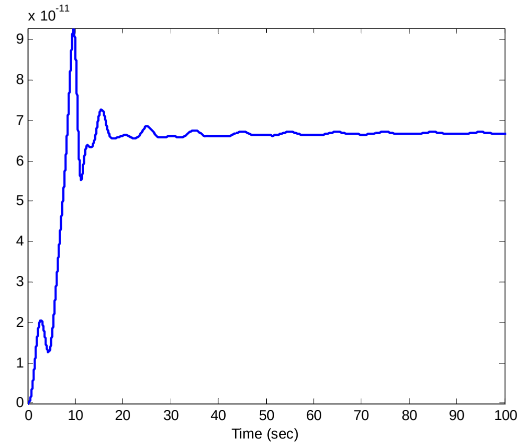
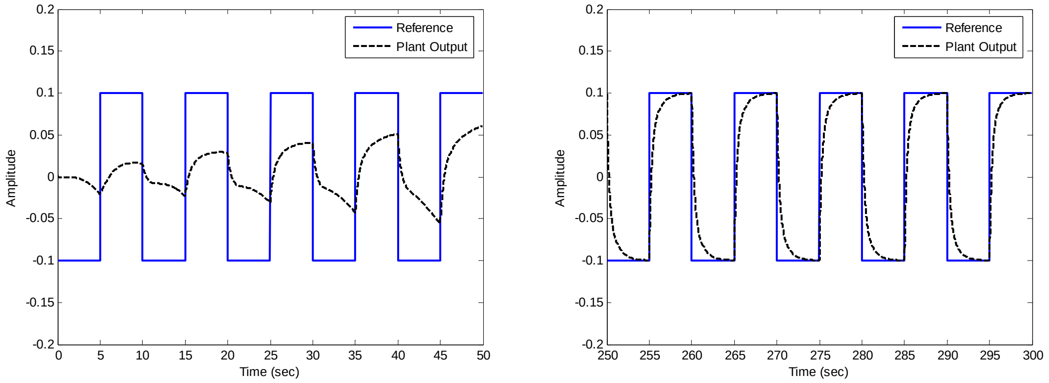


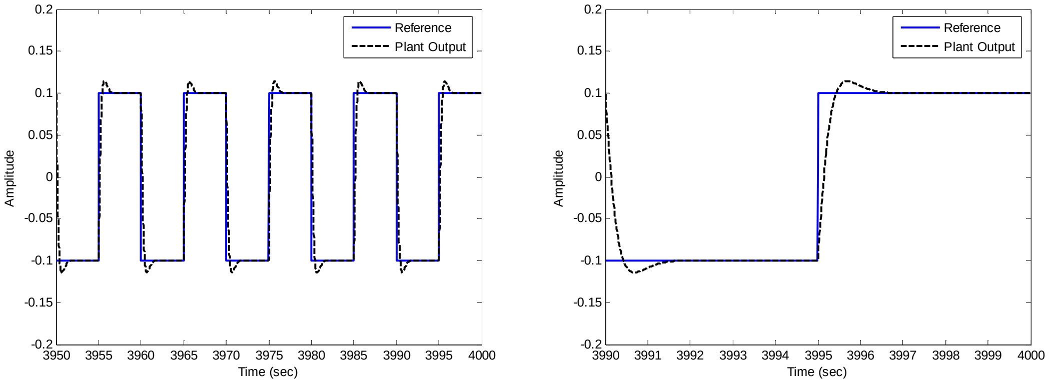
**Figure 5.8** Derivative Term during First Ten Cycles

The controller was able to track the reference signal and the PID terms stabilized by the end of the simulation. The plant output started from zero and began to follow the reference signal. By the 10th cycle they were very close and by the 20th cycle the two signals were almost identical.

The step signal had a frequency of 0.1Hz and maximum amplitude of 0.1. The initial values of the PID terms were $K_P = 0$, $K_I = 0$, $K_D = 0$. The learning rates for the wavelet network weights, dilations, translations and linear coefficients were all set to $3\times10^{-6}$. The momentum coefficient was set to 0. The proportional, integral and derivative learning rates were set to $1\times10^{-1}$, $1\times10^{-5}$ and $1\times10^{-8}$, respectively. The sampling interval was set to 0.01 seconds. The simulation was run for 400 cycles. Figure 5.9 (a)-(c) shows the tracking performance of the controller in five cycle long plots. Figure 5.9 (d) shows the final cycle in the simulation.



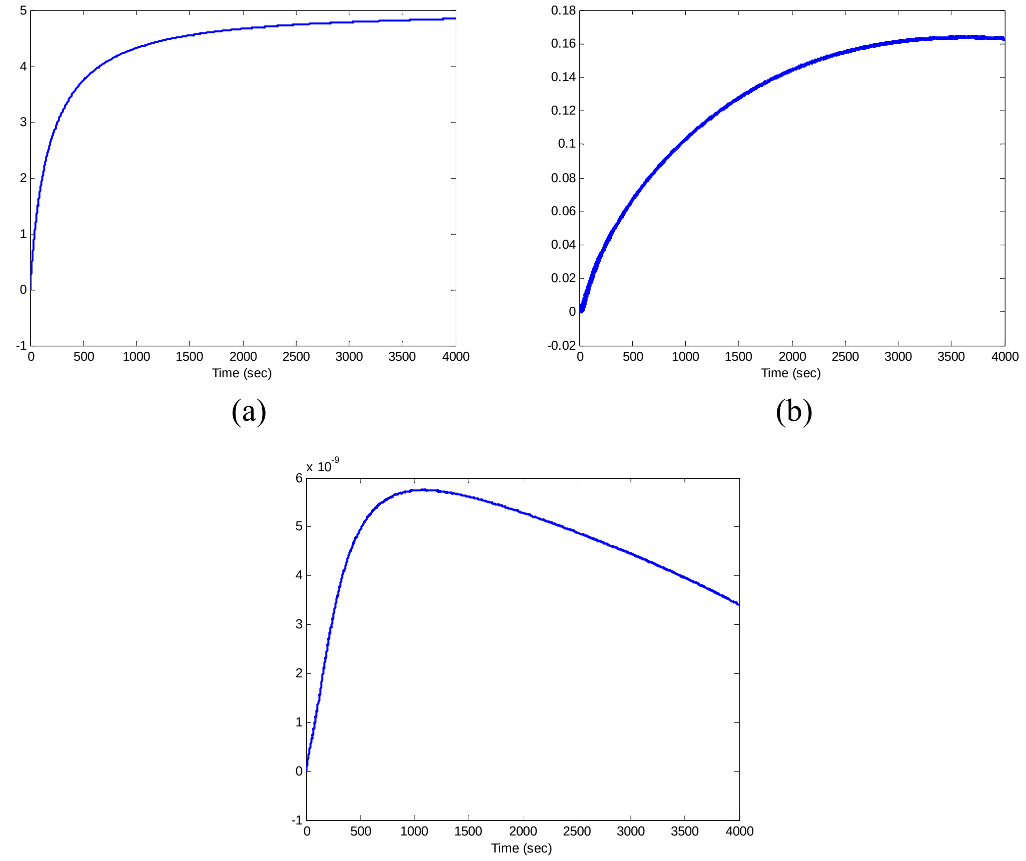
(a) (b)



(c) (d)

**Figure 5.9 (a)-(c)** Tracking Plots and **(d)** Final Cycle (Step Ref.)

The final values of the PID terms were $K_P = 4.848$, $K_I = 0.163$, $K_D = 3.42\text{x}10^{-9}$. Figure 5.10 (a)-(c) shows the plots of the PID terms during the simulation.

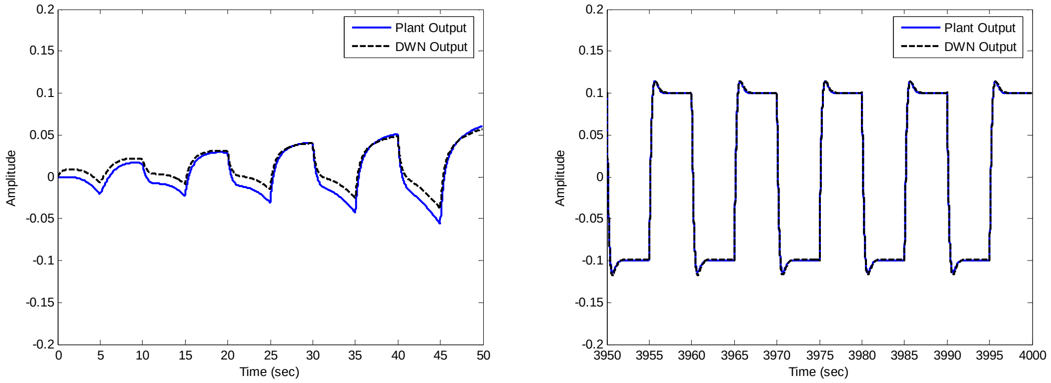


(a) (b)

(c)

**Figure 5.10 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

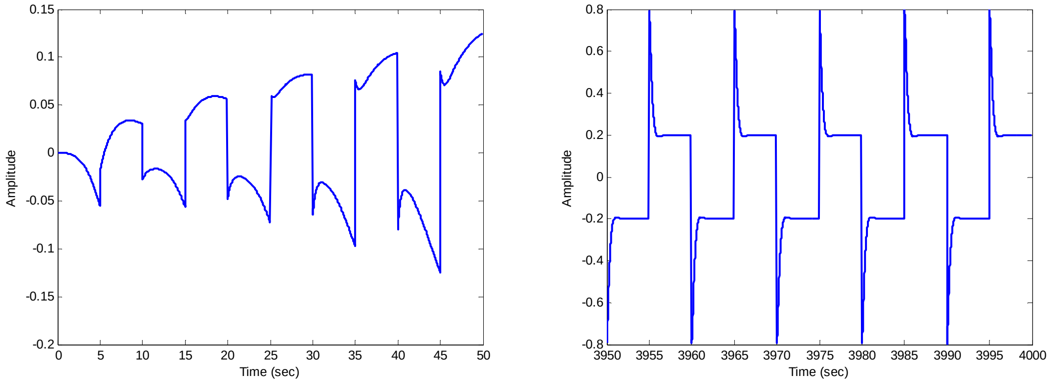
Figure 5.11 (a) shows the DWN output during the first five cycles and 5.11 (b) shows the DWN output during the last five cycles.



(a) (b)

**Figure 5.11 (a)-(b)** DWN Output during Simulation

Figure 5.12 (a) shows the controller output during the first five cycles and 5.12 (b) shows the controller output during the last five cycles.



(a) (b)

**Figure 5.12 (a)-(b)** Controller Output during Simulation

The controller was able to track the reference signal and the proportional and integral terms stabilized by the end of the simulation. While the derivative term had not stabilized, its value was around zero during the simulation. The plant output started from zero and began to follow the reference signal. By the 100th cycle the plant output was close to the reference signal and by the 150th cycle the plant output was close in shape to the final cycle, the difference being in the overshoot. The larger value of the control signal's maximum amplitude relative to the reference signal's maximum amplitude was due to the proportional and derivative parts of the discrete PID control equation.

#### 5.2.2.1 Effect of Noise

The PID-DWN controller was tested in tracking the step signal, as in subsection 5.2.2, with the addition of noise. As in subsection 4.5.1, two cases of noise contamination were considered: input noise and output noise. The first involved the addition of noise to the input of the DWN, while the second involved the addition of noise to the output of the plant. The noise signal was a random noise with a uniform distribution and its mean squared value was calculated as in Equation 4.18. In both cases, two increasing noise levels were successively added.

The DWN was trained on the nonlinear system off-line under similar noisy conditions as in subsection 4.5.1. The initial PID values, as well as the learning rates of the DWN parameters and PID terms were set as in subsection 5.2.2. The sampling interval was set to 0.01 seconds. The simulation was run for 400 cycles.

For the case of input noise, the first noise signal had a mean squared value of 4.09 $x10^{-8}$, shown in Figure 5.13. This resulted in the input signal having a SNR of 61.63.

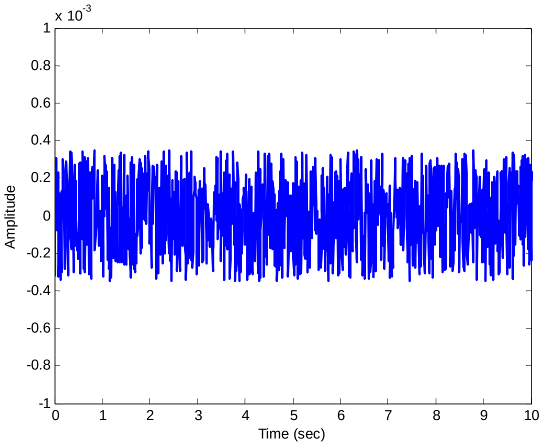
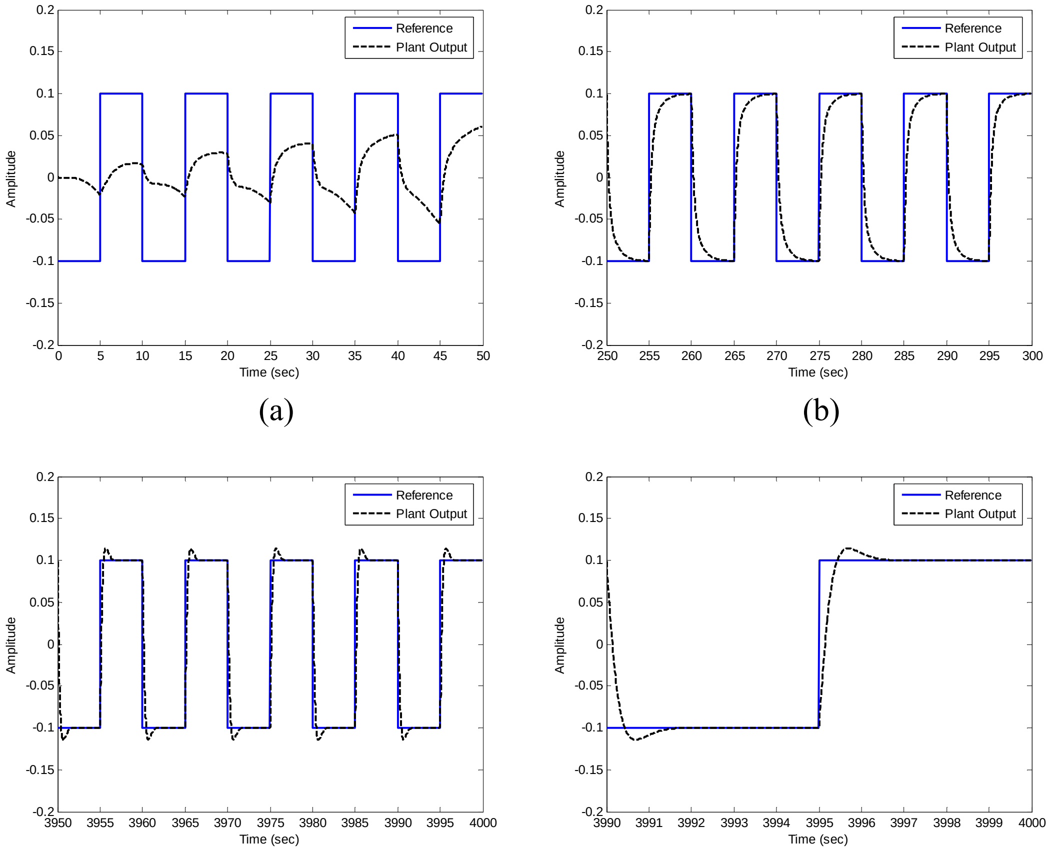


**Figure 5.13** Input Noise (MS = 4.09 $x10^{-8}$)

Figure 5.14 (a)-(c) shows the tracking performance of the controller in five cycle long plots. Figure 5.14 (d) shows the final cycle in the simulation.

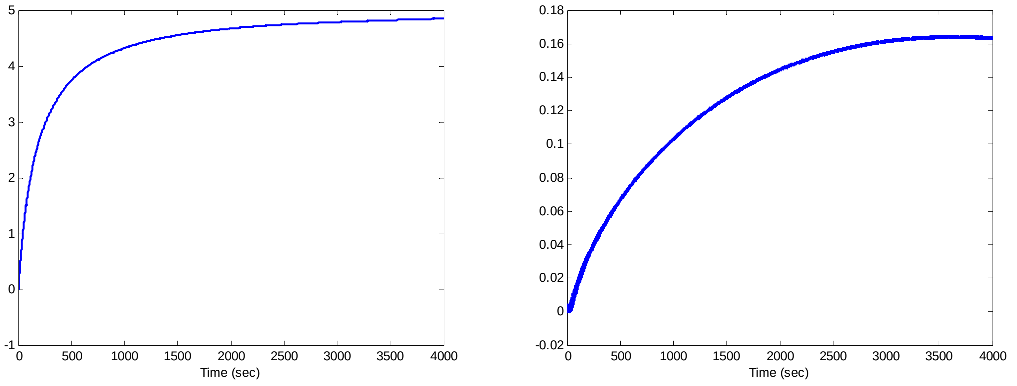


(a) (b)

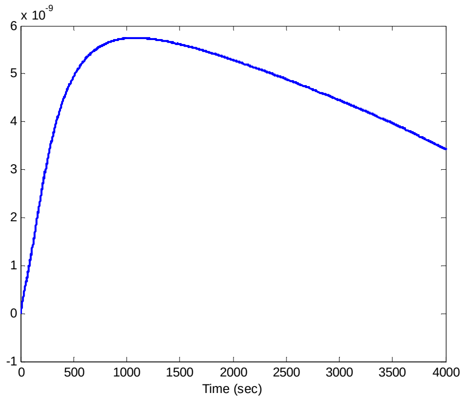
(c) (d)

**Figure 5.14 (a)-(c)** Tracking Plots and **(d)** Final Cycle (Input SNR = 61.63)

The final values of the PID terms were $K_P = 4.848$, $K_I = 0.163$, $K_D = 3.43 \times 10^{-9}$. Figure 5.15 (a)-(c) shows the plots of the PID terms during the simulation.



(a) (b)

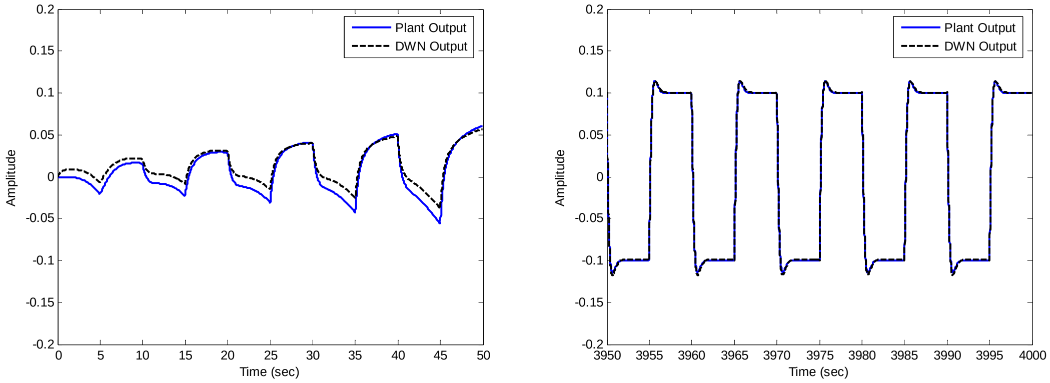


(c)

**Figure 5.15 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

(Input SNR = 61.63)

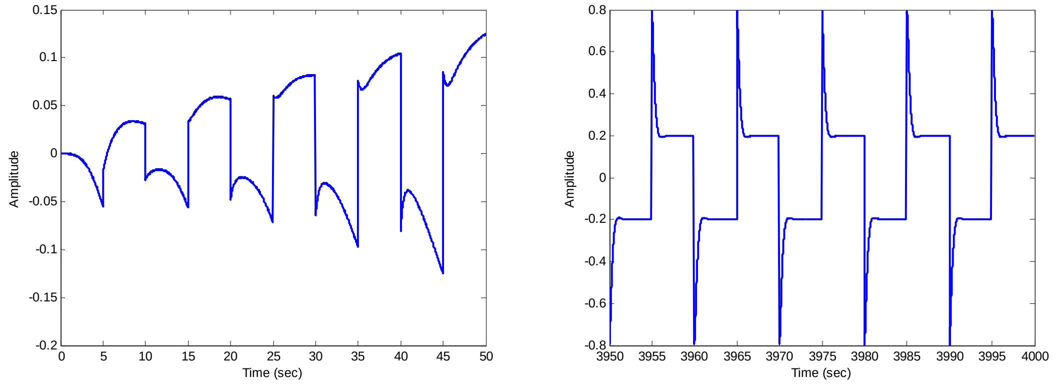
Figure 5.16 (a) shows the DWN output during the first five cycles and 5.15 (b) shows the DWN output during the last five cycles.



(a) (b)

**Figure 5.16 (a)-(b)** DWN Output during Simulation (Input SNR = 61.63)

Figure 5.17 (a) shows the controller output during the first five cycles and 5.17 (b) shows the controller output during the last five cycles.



(a) (b)

**Figure 5.17 (a)-(b)** Controller Output during Simulation (Input SNR = 61.63)

The controller displayed similar behavior to the noise-free case with no differences in the final values of the PID terms. Thus, an input signal having a SNR of 61.63 had no effect on the performance of the PID-DWN controller.

The second noise signal had a mean squared value of 4.09 x10$^{-6}$, shown in Figure 5.18. This resulted in the input signal having a SNR of 41.63.

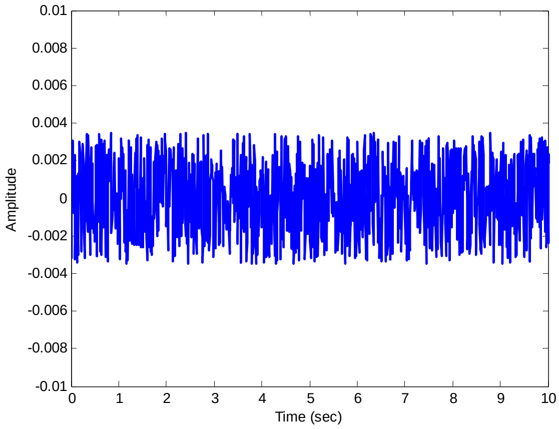
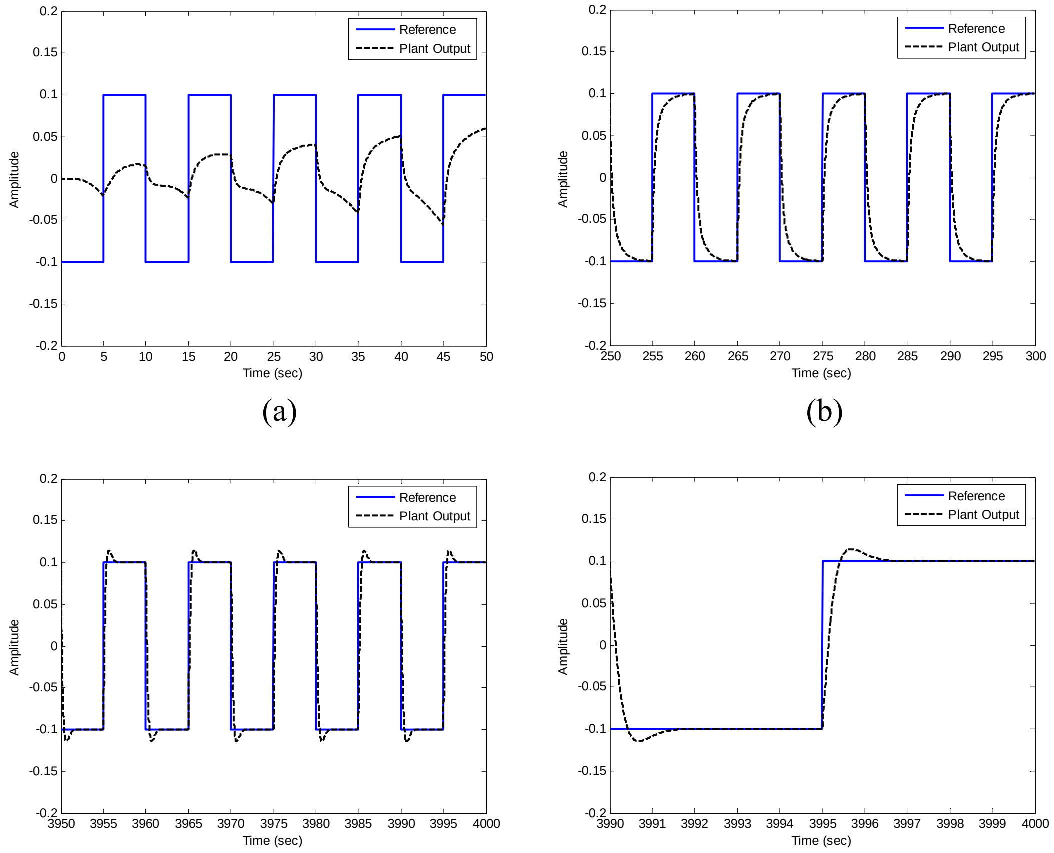


**Figure 5.18** Input Noise (MS = 4.09 x10$^{-6}$)

Figure 5.19 (a)-(c) shows the tracking performance of the controller in five cycle long plots. Figure 5.19 (d) shows the final cycle in the simulation.

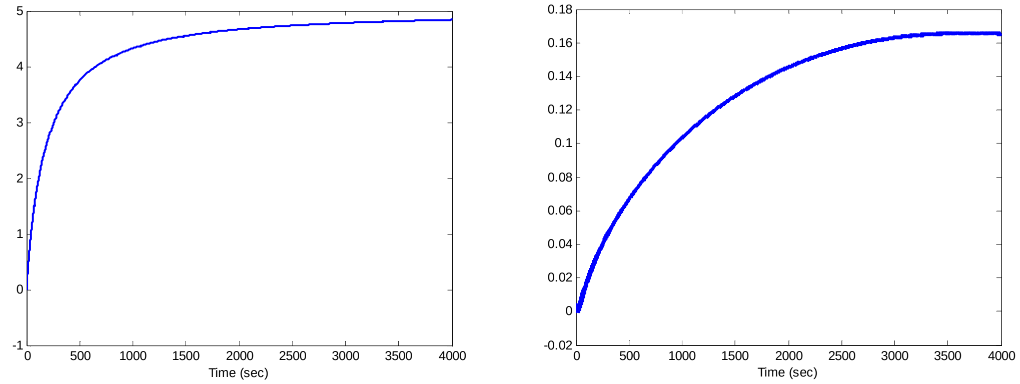


(a) (b)

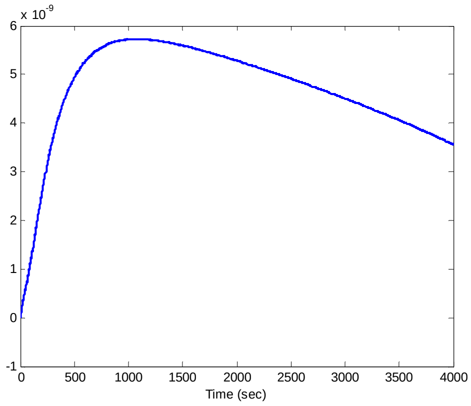
(c) (d)

**Figure 5.19 (a)-(c)** Tracking Plots and **(d)** Final Cycle (Input SNR = 41.63)

The final values of the PID terms were $K_P = 4.842$, $K_I = 0.165$, $K_D = 3.56\text{x}10^{-9}$. Figure 5.20 (a)-(c) shows the plots of the PID terms during the simulation.



(a) (b)

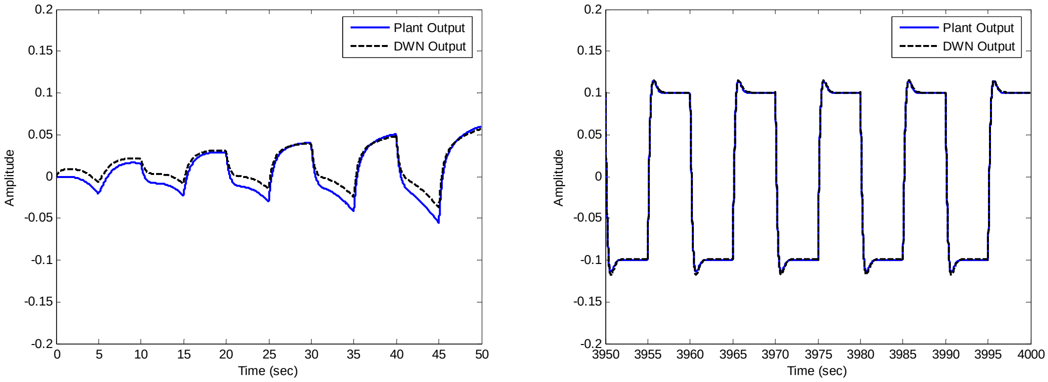


(c)

**Figure 5.20 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

(Input SNR = 41.63)

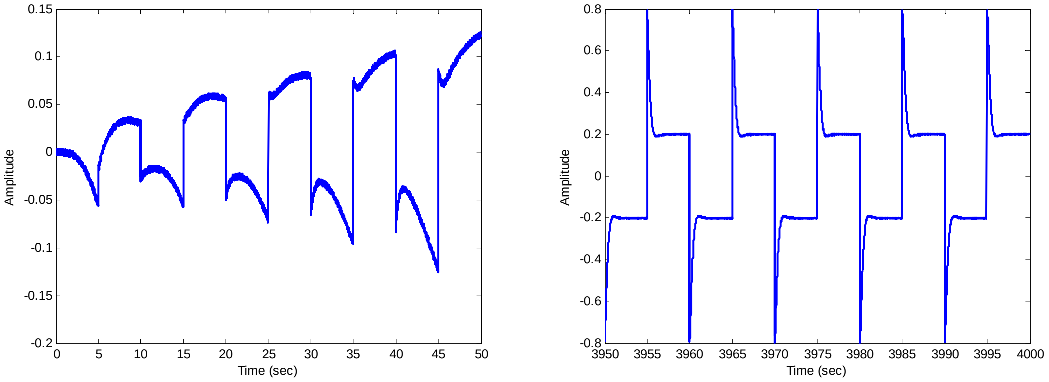
Figure 5.21 (a) shows the DWN output during the first five cycles and 5.21 (b) shows the DWN output during the last five cycles.



(a) (b)

**Figure 5.21 (a)-(b)** DWN Output during Simulation (Input SNR = 41.63)

Figure 5.22 (a) shows the controller output during the first five cycles and 5.22 (b) shows the controller output during the last five cycles.



(a) (b)

**Figure 5.22 (a)-(b)** Controller Output during Simulation (Input SNR = 41.63)

The controller displayed similar behavior to the noise-free case with minor differences in the final values of the PID terms. Thus, an input signal having a SNR of 41.63 had no effect on the performance of the PID-DWN controller.

For the case of output noise, the first noise signal had a mean squared value of 1.08 x10$^{-8}$, shown in Figure 5.23. This resulted in the output signal having a SNR of 59.35.

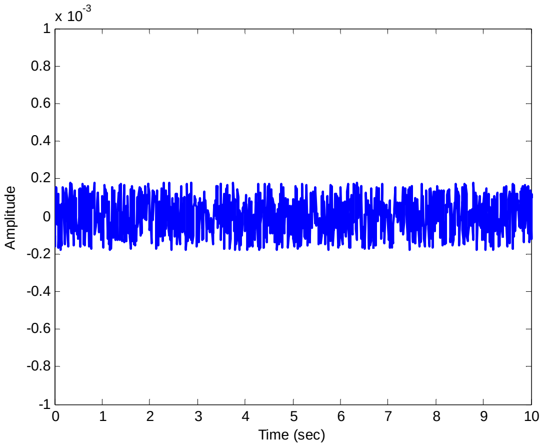
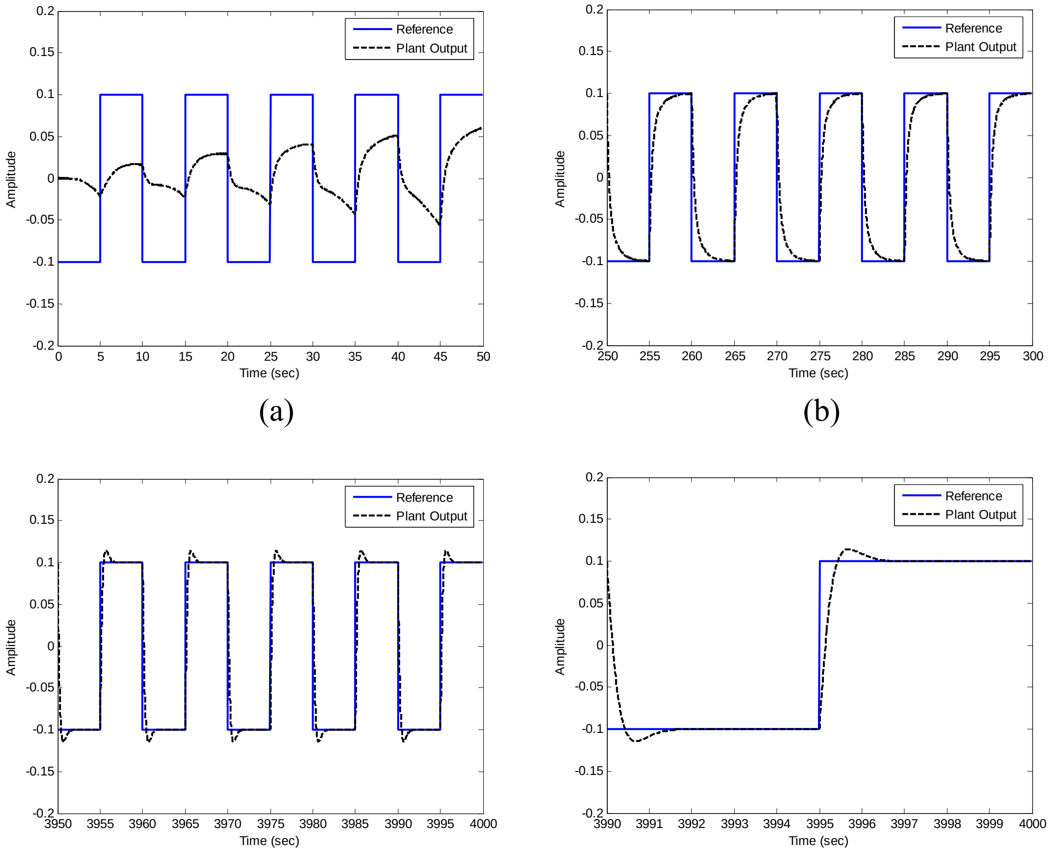


**Figure 5.23** Output Noise (MS = 1.08 x10$^{-8}$)

Figure 5.24 (a)-(c) shows the tracking performance of the controller in five cycle long plots. Figure 5.24 (d) shows the final cycle in the simulation.

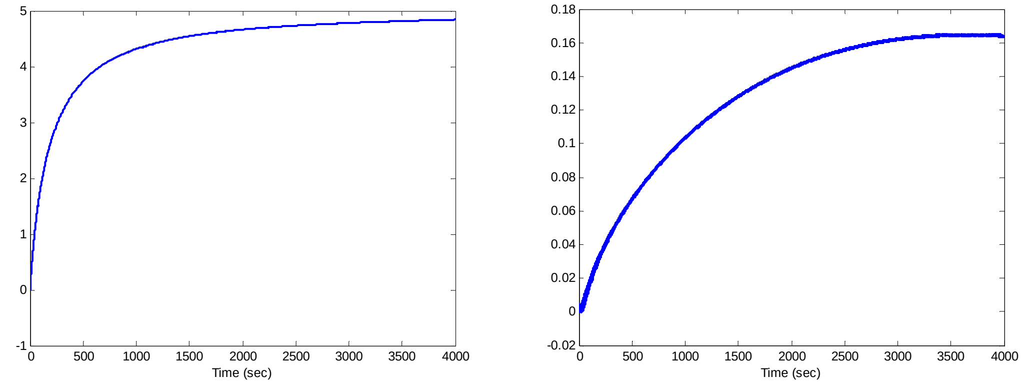


(a) (b)

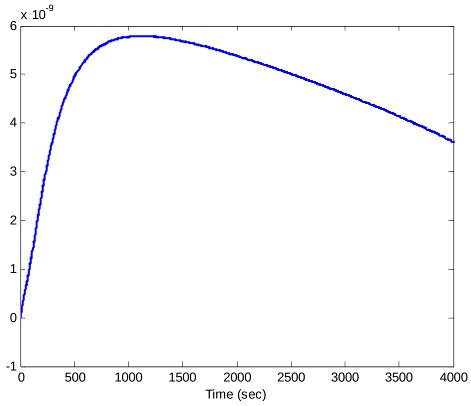
(c) (d)

**Figure 5.24 (a)-(c)** Tracking Plots and **(d)** Final Cycle (Output SNR = 59.35)

The final values of the PID terms were $K_P = 4.842$, $K_I = 0.164$, $K_D = 3.62\text{x}10^{-9}$. Figure 5.25 (a)-(c) shows the plots of the PID terms during the simulation.



(a) (b)

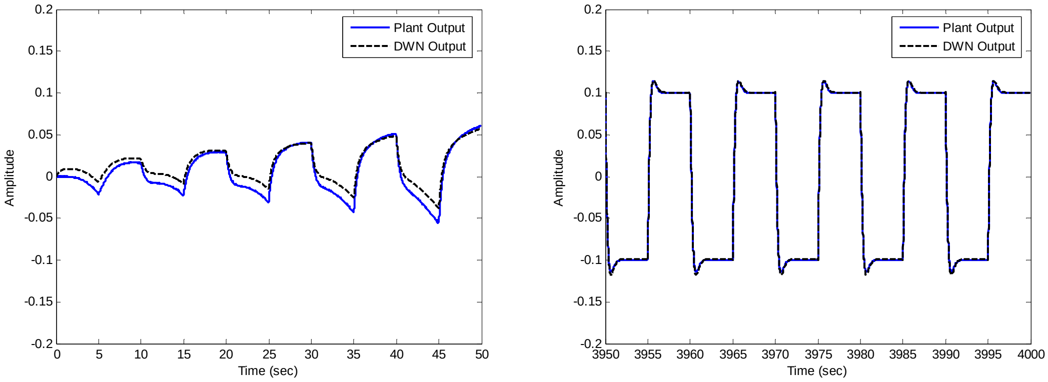


(c)

**Figure 5.25 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

(Output SNR = 59.35)

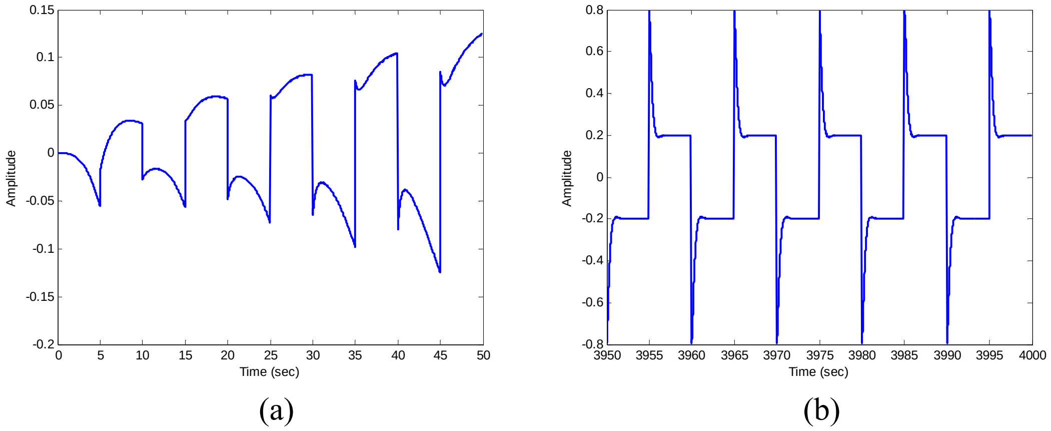
Figure 5.26 (a) shows the DWN output during the first five cycles and 5.26 (b) shows the DWN output during the last five cycles.



(a) (b)

**Figure 5.26 (a)-(b)** DWN Output during Simulation (Output SNR = 59.35)

Figure 5.27 (a) shows the controller output during the first five cycles and 5.27 (b) shows the controller output during the last five cycles.



(a) (b)

**Figure 5.27 (a)-(b)** Controller Output during Simulation (Output SNR = 59.35)

The controller displayed similar behavior to the noise-free case with minor differences in the final values of the PID terms. Thus, an output signal having a SNR of 59.35 had no effect on the performance of the PID-DWN controller.

The second noise signal had a mean squared value of 1.08 x$10^{-6}$, shown in Figure 5.28. This resulted in the output signal having a SNR of 39.35.

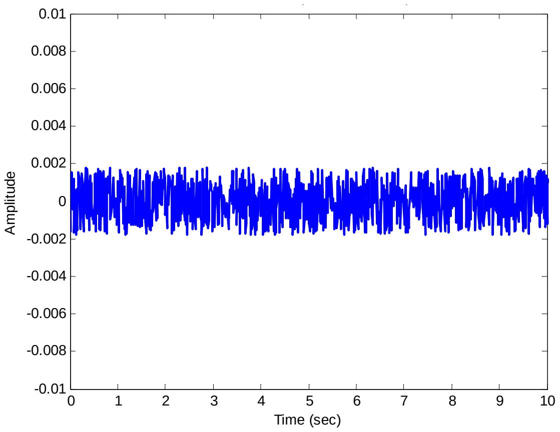
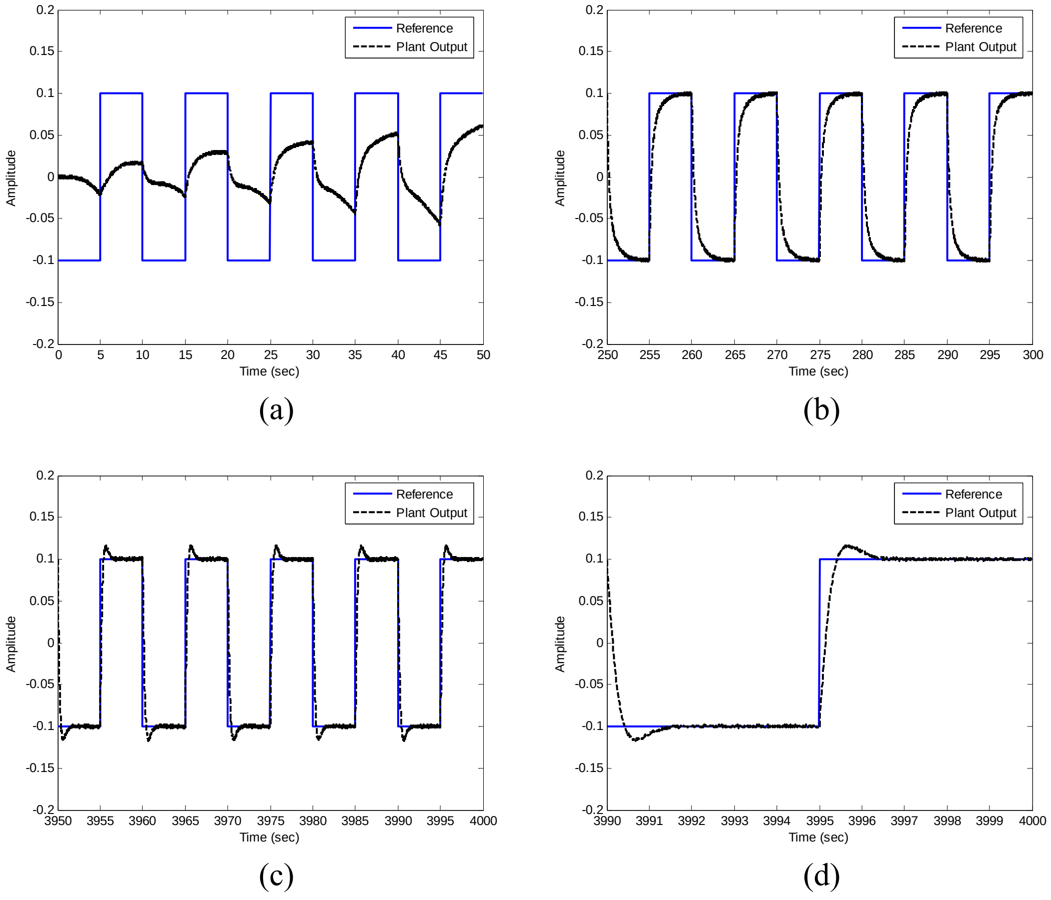


**Figure 5.28** Output Noise (MS = 1.08 x$10^{-6}$)

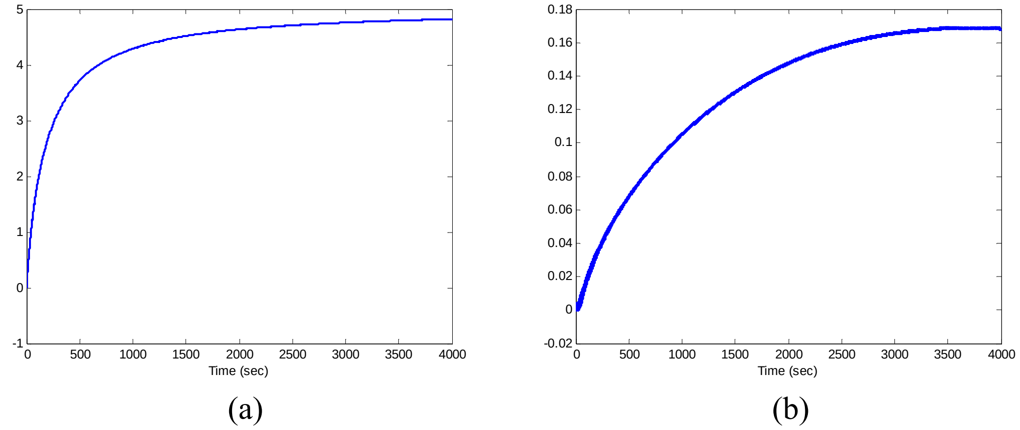
Figure 5.29 (a)-(c) shows the tracking performance of the controller in five cycle long plots. Figure 5.29 (d) shows the final cycle in the simulation.



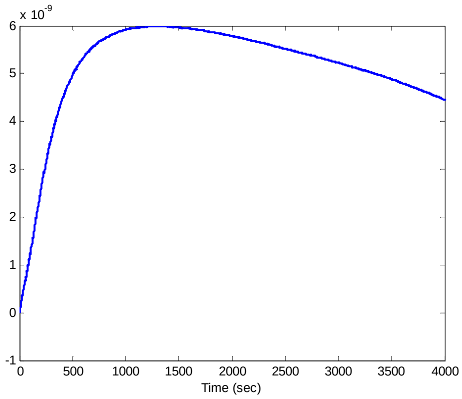
(a) (b) (c) (d)

**Figure 5.29 (a)-(c)** Tracking Plots and **(d)** Final Cycle (Output SNR = 39.35)

The final values of the PID terms were $K_P = 4.823$, $K_I = 0.168$, $K_D = 4.46x10^{-9}$. Figure 5.30 (a)-(c) shows the plots of the PID terms during the simulation.



(a) (b)

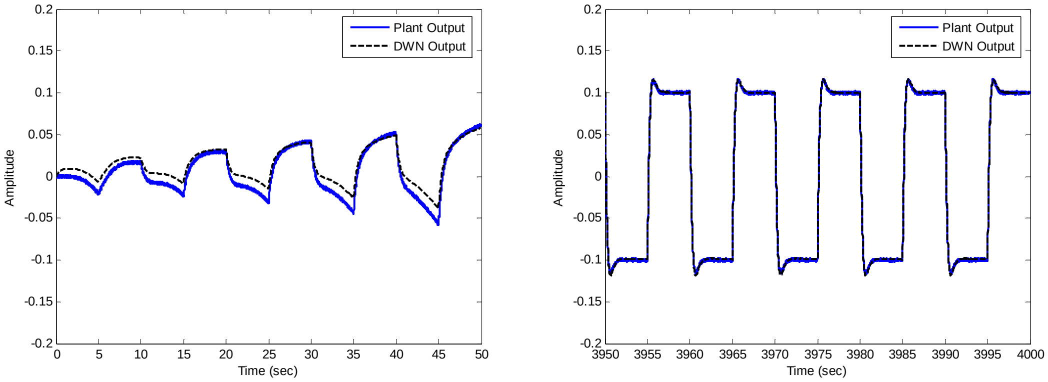


(c)

**Figure 5.30 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

(Output SNR = 39.35)

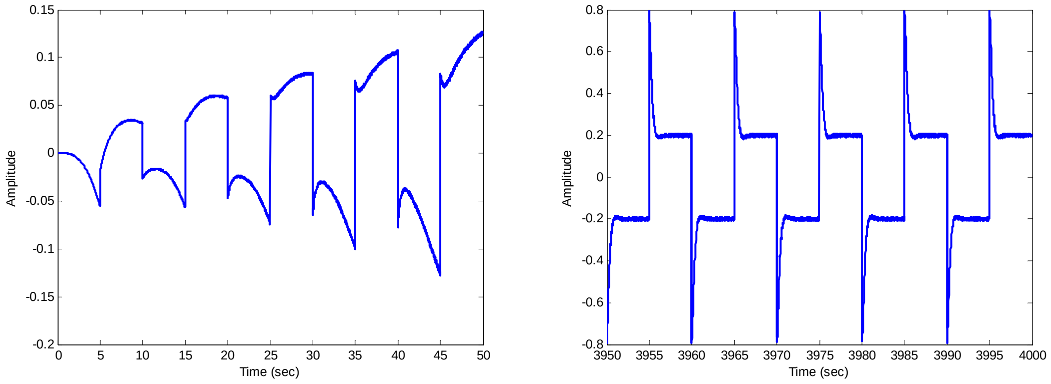
Figure 5.31 (a) shows the DWN output during the first five cycles and 5.31 (b) shows the DWN output during the last five cycles.



(a) (b)

**Figure 5.31 (a)-(b)** DWN Output during Simulation (Output SNR = 39.35)

Figure 5.32 (a) shows the controller output during the first five cycles and 5.32 (b) shows the controller output during the last five cycles.



(a) (b)

**Figure 5.32 (a)-(b)** Controller Output during Simulation (Output SNR = 39.35)

The controller displayed similar behavior to the noise-free case with minor differences in the final values of the PID terms. Thus, an output signal having a SNR of 39.35 had no effect on the performance of the PID-DWN controller.

### 5.2.3 Second Order System with Saturation

The same system studied in subsection 4.3, shown in Figure 5.33, was to be controlled by the proposed PID-DWN controller.

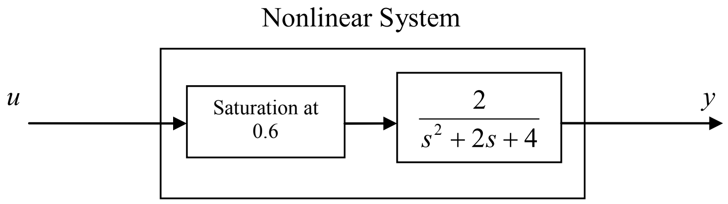
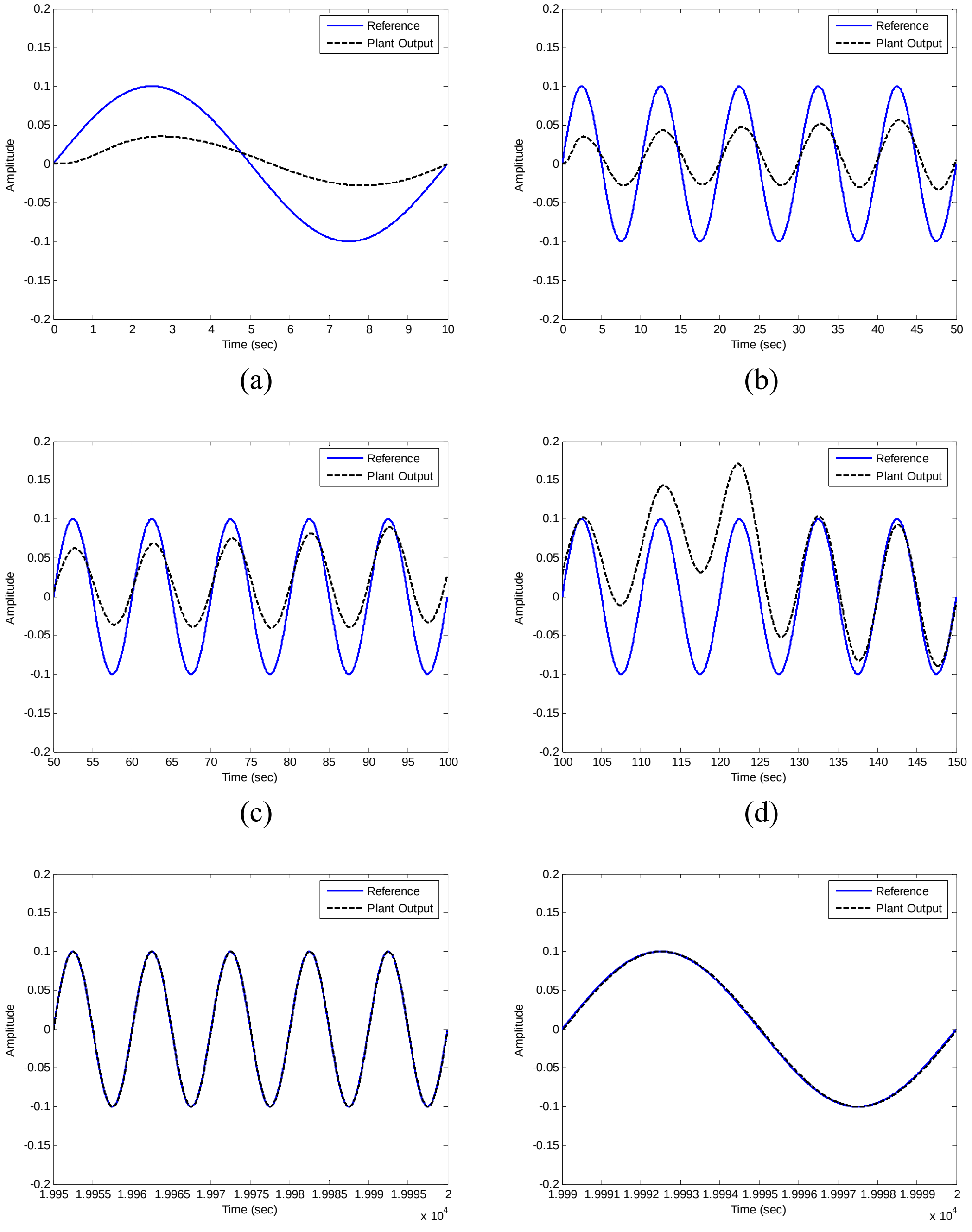


**Figure 5.33** Second Order System with Saturation

The DWN was first trained on the nonlinear system off-line. In order for the DWN to be properly trained and learn the system dynamics, a chirp signal, as in subsection 4.5.2, was used for the identification. The system was also modeled, using the wavelet network, initialization method and learning rates, as in subsection 4.5.2.

The PID-DWN controller was tested in tracking two reference signals: a sinusoid and a step.

The sinusoid signal had a frequency of 0.1Hz and maximum amplitude of 0.1. The initial values of the PID terms were $K_P = 1$, $K_I = 0$, $K_D = 0$. The learning rates for the wavelet network weights, dilations, and translations were set to $5\text{x}10^{-5}$. The learning rates for the linear coefficients were set to 5x10$^{-10}$. The momentum coefficient was set to 0. The proportional, integral and derivative learning rates were set to 2.8, 2.8x10$^{-6}$ and 1x10$^{-8}$, respectively. The sampling interval was set to 0.01 seconds. The simulation was run for 2000 cycles. Figure 5.34 (a) shows the initial cycle in the simulation. Figure 5.34 (b)-(e) shows the tracking performance of the controller in five cycle long plots. Figure 5.34 (f) shows the final cycle in the simulation.



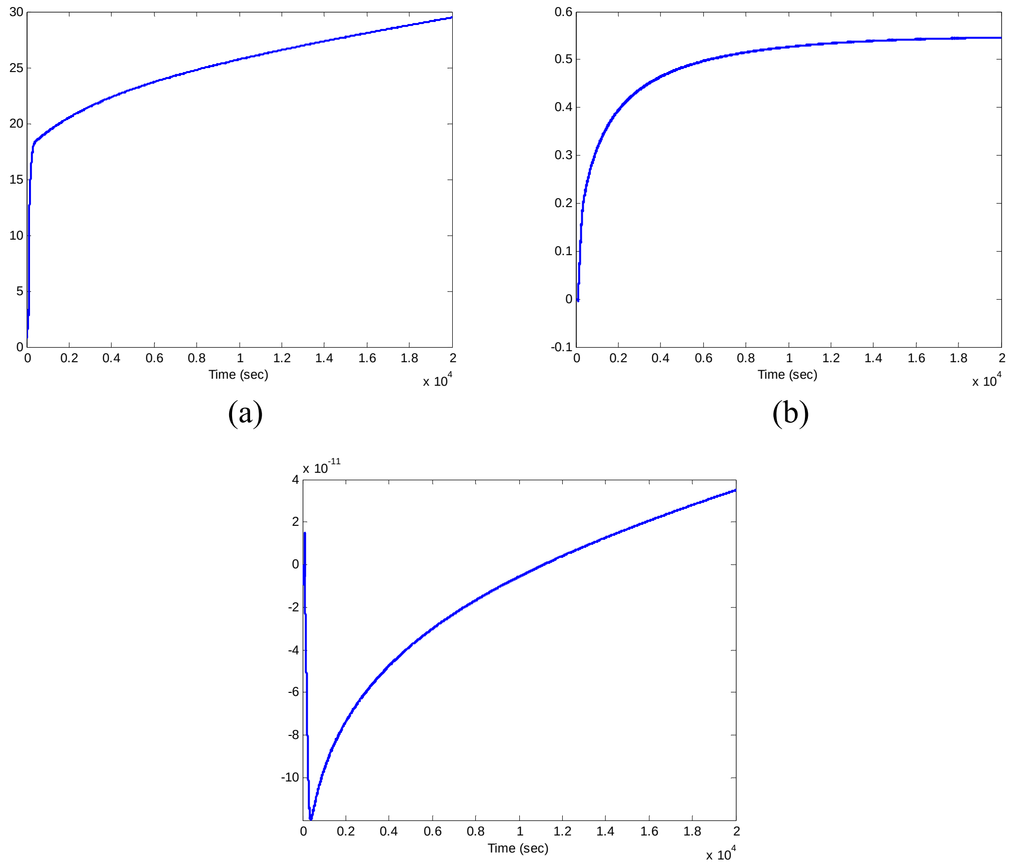
(a) (b)

(c) (d)

(e) (f)

**Figure 5.34 (a)** Initial Cycle, **(b)-(e)** Tracking Plots and **(f)** Final Cycle (Sine Ref.)

The final values of the PID terms were $K_P = 29.478$, $K_I = 0.546$, $K_D = 3.5\text{x}10^{-11}$. Figure 5.35 (a)-(c) shows the plots of the PID terms during the simulation.



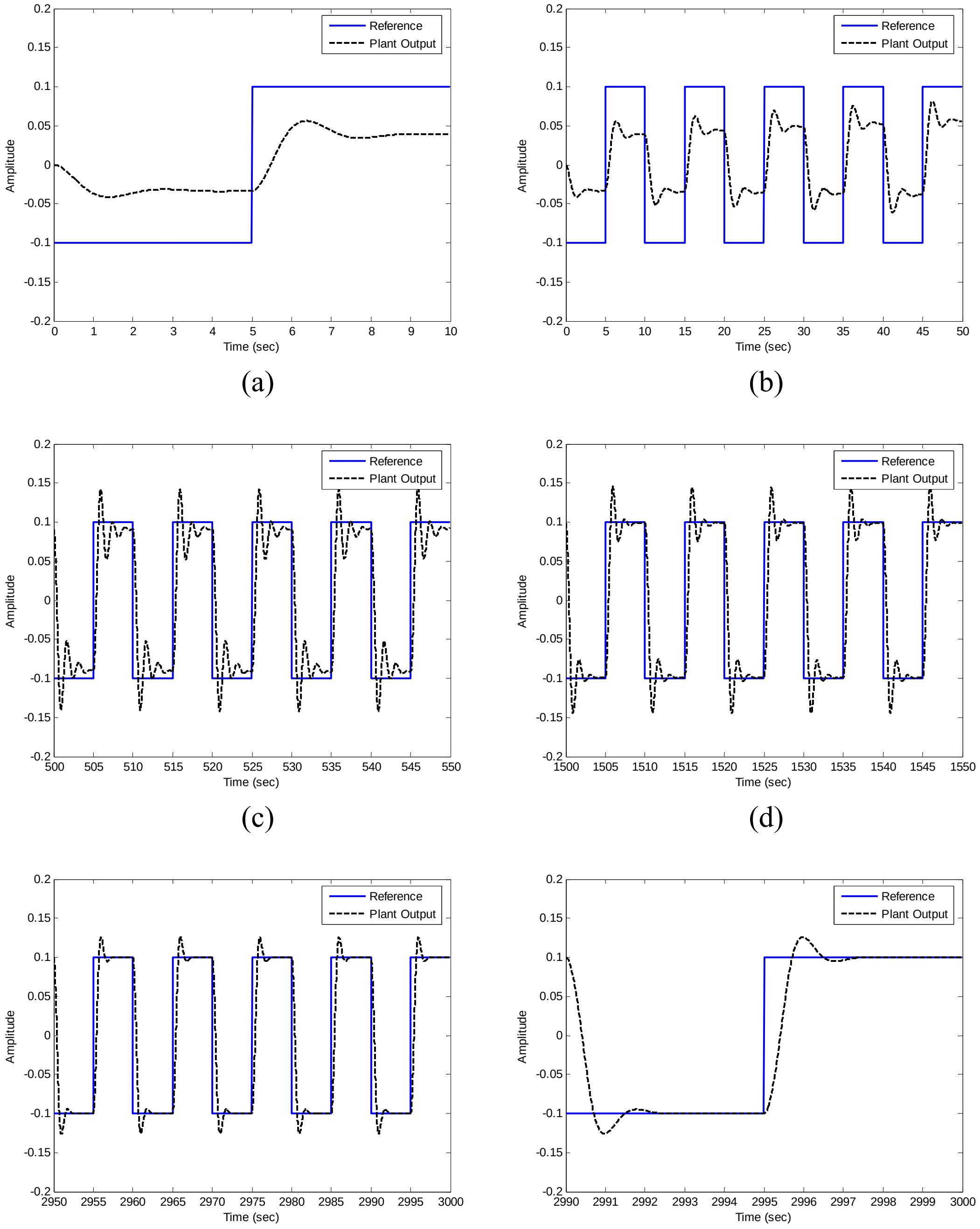
(a) (b)

(c)

**Figure 5.35 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

The controller was able to track the reference signal and by the end of the simulation the Integral term stabilized while the proportional and derivative terms were beginning to stabilize. By the 30th cycle the plant output was very close to the reference signal and by the end of the simulation the two signals were almost identical.

The step signal had a frequency of 0.1Hz and maximum amplitude of 0.1. The initial values of the PID terms were $K_P = 1$, $K_I = 0$, $K_D = 0$. The learning rates for the wavelet network weights, dilations, and translations were set to $3\text{x}10^{-5}$. The learning rates for the linear coefficients were set to $3\text{x}10^{-10}$. The momentum coefficient was set to 0. The proportional, integral and derivative learning rates were set to $2\text{x}10^{-1}$, $4\text{x}10^{-7}$ and $1.2\text{x}10^{2}$, respectively. The sampling interval was set to 0.01 seconds. The simulation was run for 300 cycles. Figure 5.36 (a) shows the initial cycle in the simulation. Figure 5.36 (b)-(e) shows the tracking performance of the controller in five cycle long plots. Figure 5.36 (f) shows the final cycle in the simulation.



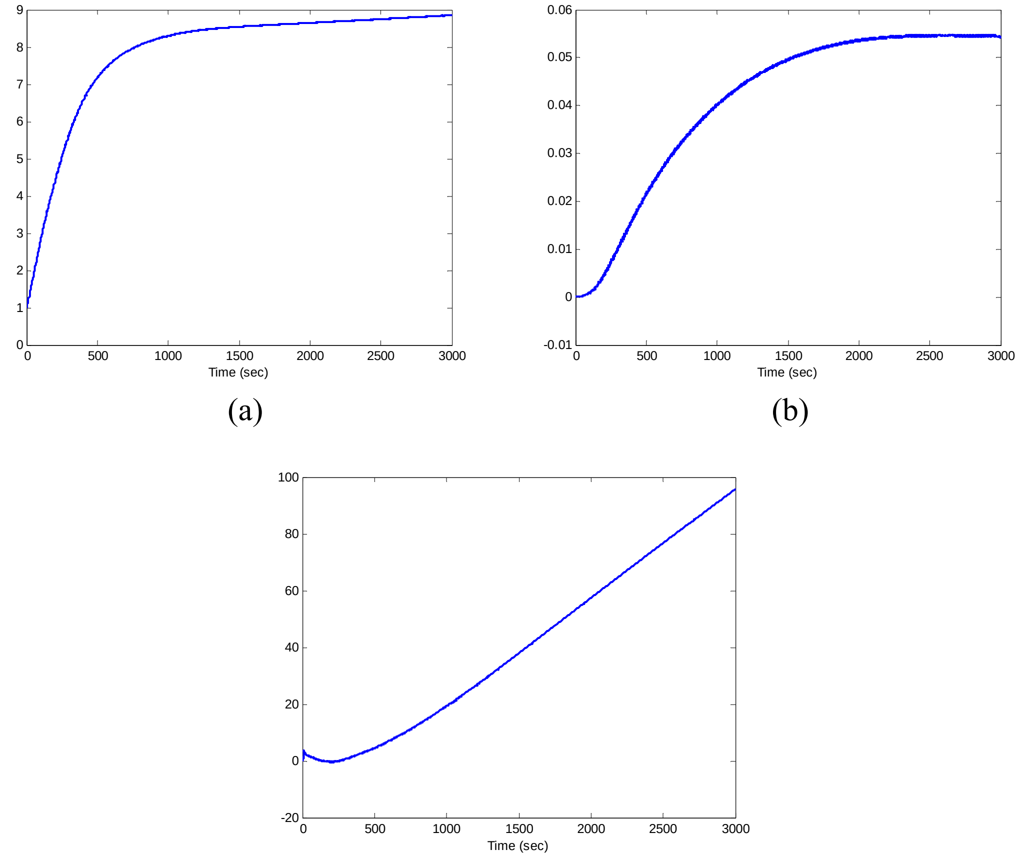
(a) (b)

(c) (d)

(e) (f)

**Figure 5.36 (a)** Initial Cycle, **(b)-(e)** Tracking Plots and **(f)** Final Cycle (Step Ref.)

The final values of the PID terms were $K_P = 8.854$, $K_I = 0.0543$, $K_D = 95.502$. Figure 5.37 (a)-(c) shows the plots of the PID terms during the simulation.

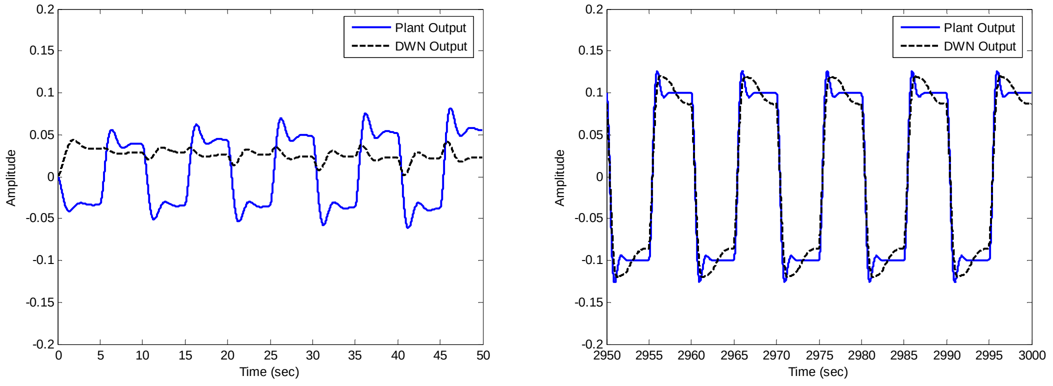


(a) (b)

(c)

**Figure 5.37 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

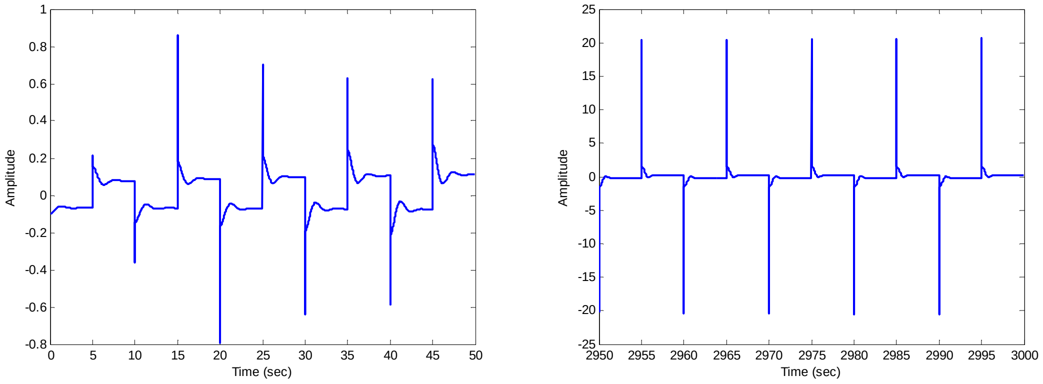
Figure 5.38 (a) shows the DWN output during the first five cycles and 5.38 (b) shows the DWN output during the last five cycles.



(a) (b)

**Figure 5.38 (a)-(b)** DWN Output during Simulation

Figure 5.39 (a) shows the controller output during the first five cycles and 5.39 (b) shows the controller output during the last five cycles.



(a) (b)

**Figure 5.39 (a)-(b)** Controller Output during Simulation

The controller was able to track the reference signal and the proportional and integral terms stabilized by the end of the simulation. Although the derivative term did not begin to stabilize, the simulation was ended based on the satisfactory tracking performance. By the 150th cycle, the plant output was close to the reference signal, but was having a big overshoot followed by oscillations. By the end of the simulation, the overshoot had decreased and the oscillations had been significantly minimized. The larger value of the control signal's maximum amplitude relative to the reference signal's maximum amplitude was due to the proportional and derivative parts of the discrete PID control equation.

#### 5.2.3.1 Effect of Noise

The PID-DWN controller was tested in tracking the same square wave input as in subsection 5.2.2 with the addition of noise. As in subsection 4.5.2, the two cases of noise contamination were considered: input noise and output noise. The first involved the addition of noise to the input of the DWN, while the second involved the addition of noise to the output of the plant. The noise signal was a random noise with a uniform distribution and its mean squared value was calculated as in Equation 4.18. In both cases, two increasing noise levels were successively added.

The DWN was trained on the nonlinear system off-line under similar noisy conditions as in subsection 4.5.2. The initial PID values, as well as the learning rates of the DWN parameters and PID terms were set as in subsection 5.2.3. The sampling interval was set to 0.01 seconds. The simulation was run for 300 cycles.

For the case of input noise, the first noise signal had a mean squared value of $3 \times 10^{-7}$, shown in Figure 5.40. This resulted in the input signal having a SNR of 61.44.

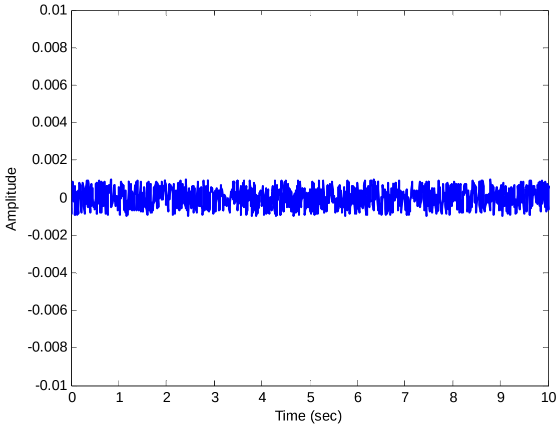
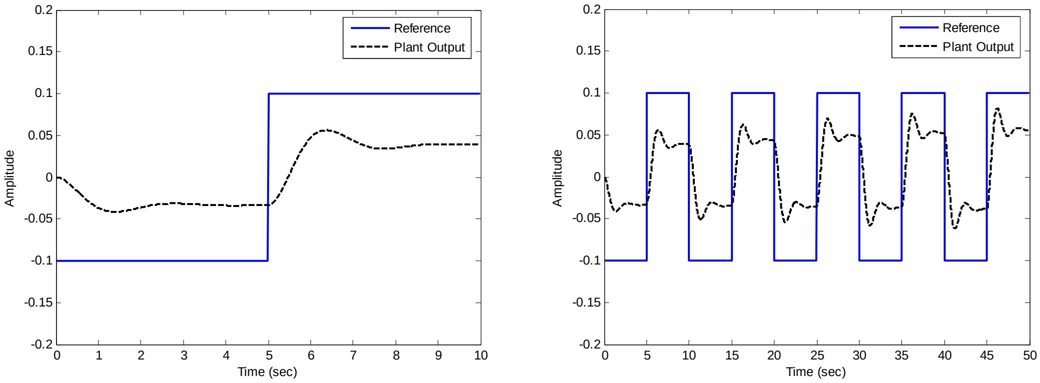
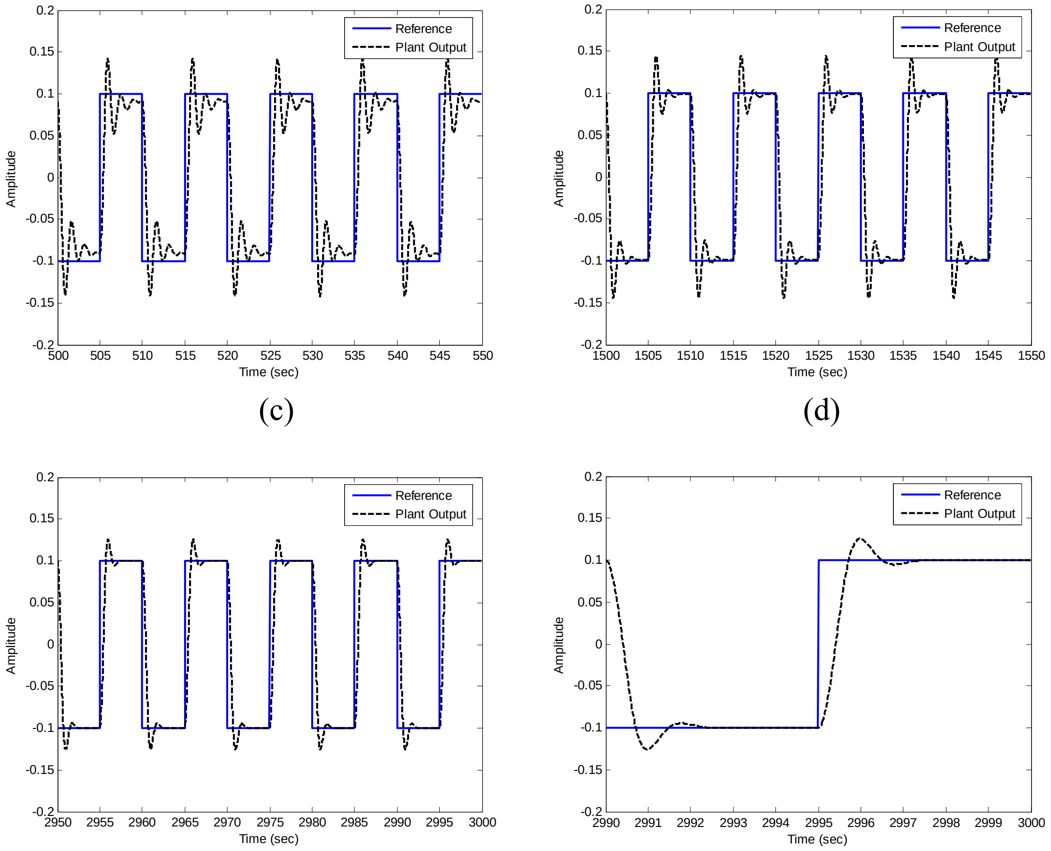


**Figure 5.40** Input Noise (MS = $3 \times 10^{-7}$)

Figure 5.41 (a) shows the initial cycle in the simulation. Figure 5.41 (b)-(e) shows the tracking performance of the controller in five cycle long plots. Figure 5.41 (f) shows the final cycle in the simulation.



(a) (b)

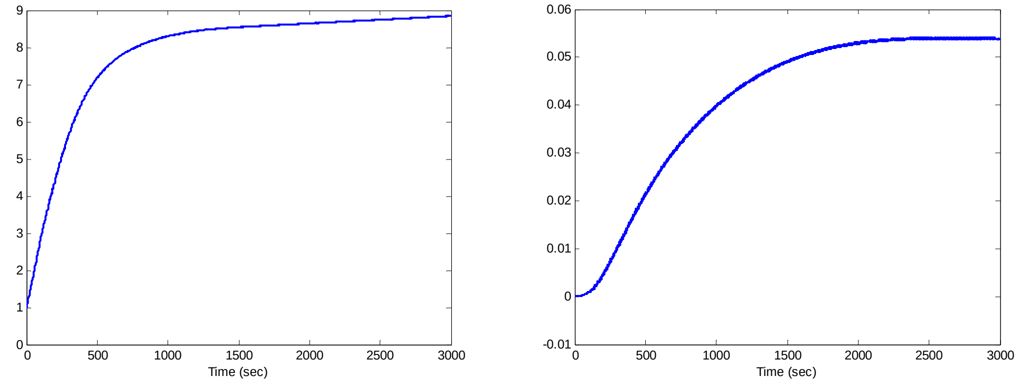


(c) (d)

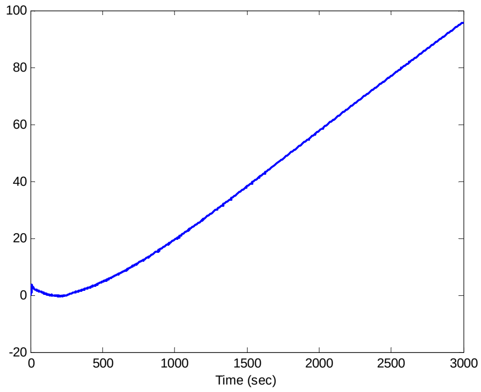
(e) (f)

**Figure 5.41 (a)** Initial Cycle, **(b)-(e)** Tracking Plots and **(f)** Final Cycle (Input SNR = 61.44)

The final values of the PID terms were $K_P = 8.848$, $K_I = 0.0537$, $K_D = 95.771$. Figure 5.42 (a)-(c) shows the plots of the PID terms during the simulation.



(a) (b)

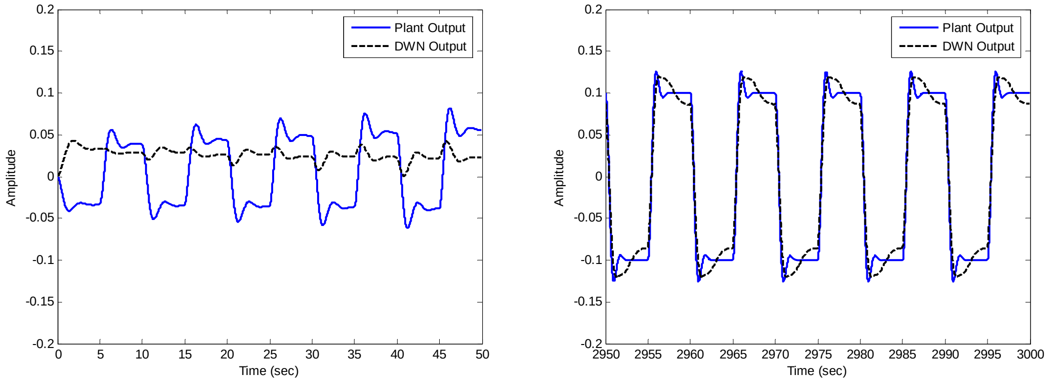


(c)

**Figure 5.42 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

(Input SNR = 61.44)

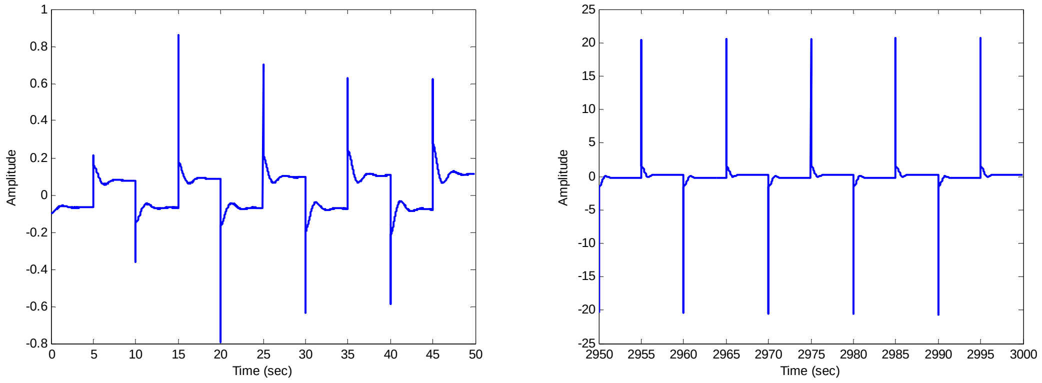
Figure 5.43 (a) shows the DWN output during the first five cycles and 5.43 (b) shows the DWN output during the last five cycles.



(a) (b)

**Figure 5.43 (a)-(b)** DWN Output during Simulation (Input Noise = 61.44)

Figure 5.44 (a) shows the controller output during the first five cycles and 5.44 (b) shows the controller output during the last five cycles.



(a) (b)

**Figure 5.44 (a)-(b)** Controller Output during Simulation (Input Noise = 61.44)

The controller displayed similar behavior to the noise-free case with minor differences in the final values of the PID terms. Thus, an input signal having a SNR of 61.44 had no effect on the performance of the PID-DWN controller.

The second noise signal had a mean squared value of 3 x$10^{-5}$, shown in Figure 5.45. This resulted in the input signal having a SNR of 41.44.

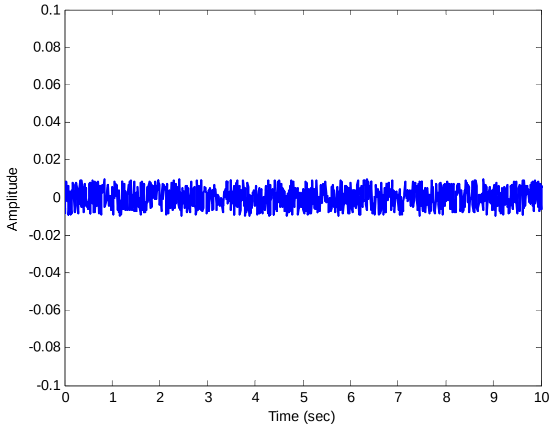
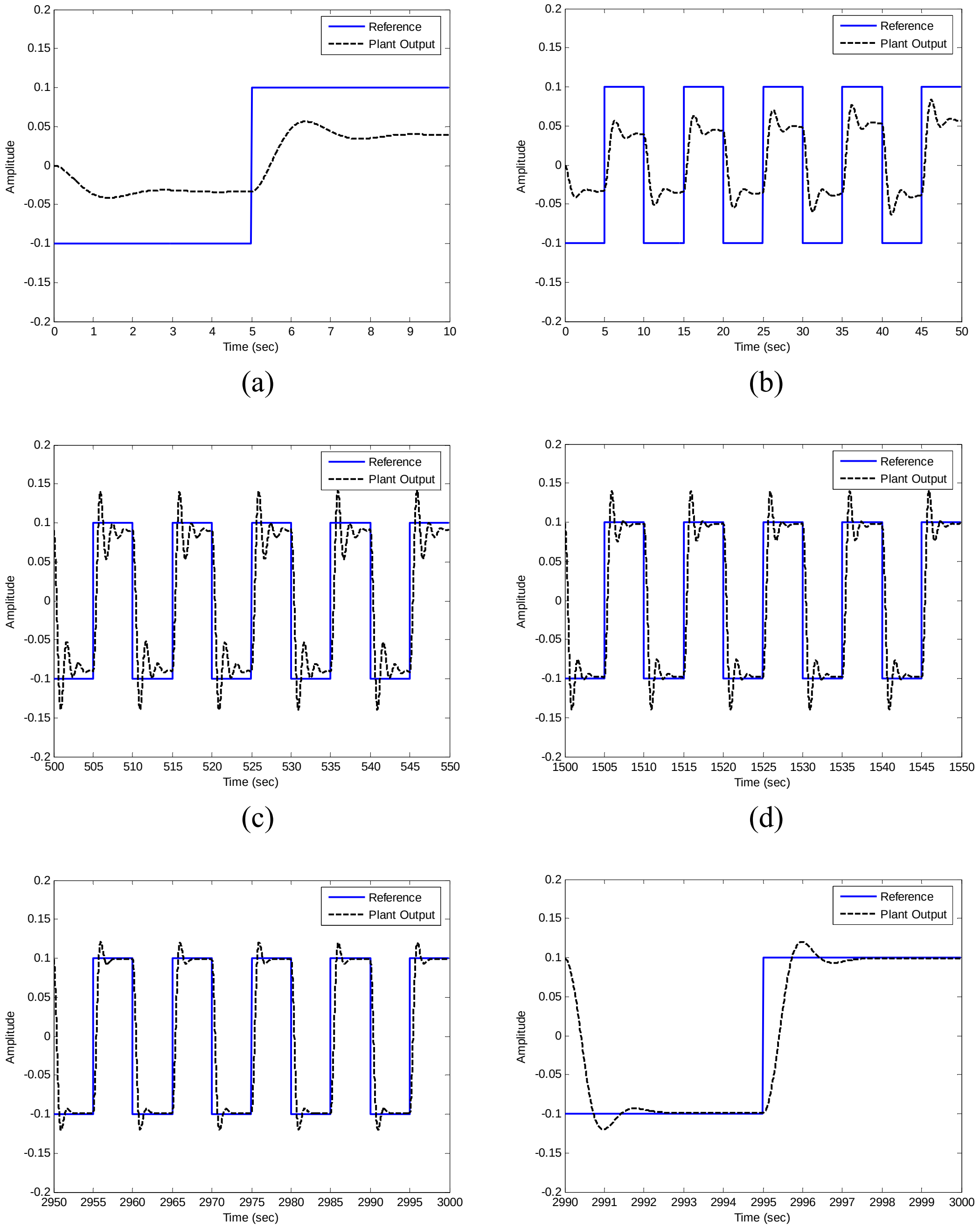


**Figure 5.45** Input Noise (MS = 3 x$10^{-5}$)

Figure 5.46 (a) shows the initial cycle in the simulation. Figure 5.46 (b)-(e) shows the tracking performance of the controller in five cycle long plots. Figure 5.46 (f) shows the final cycle in the simulation.



(a) (b)

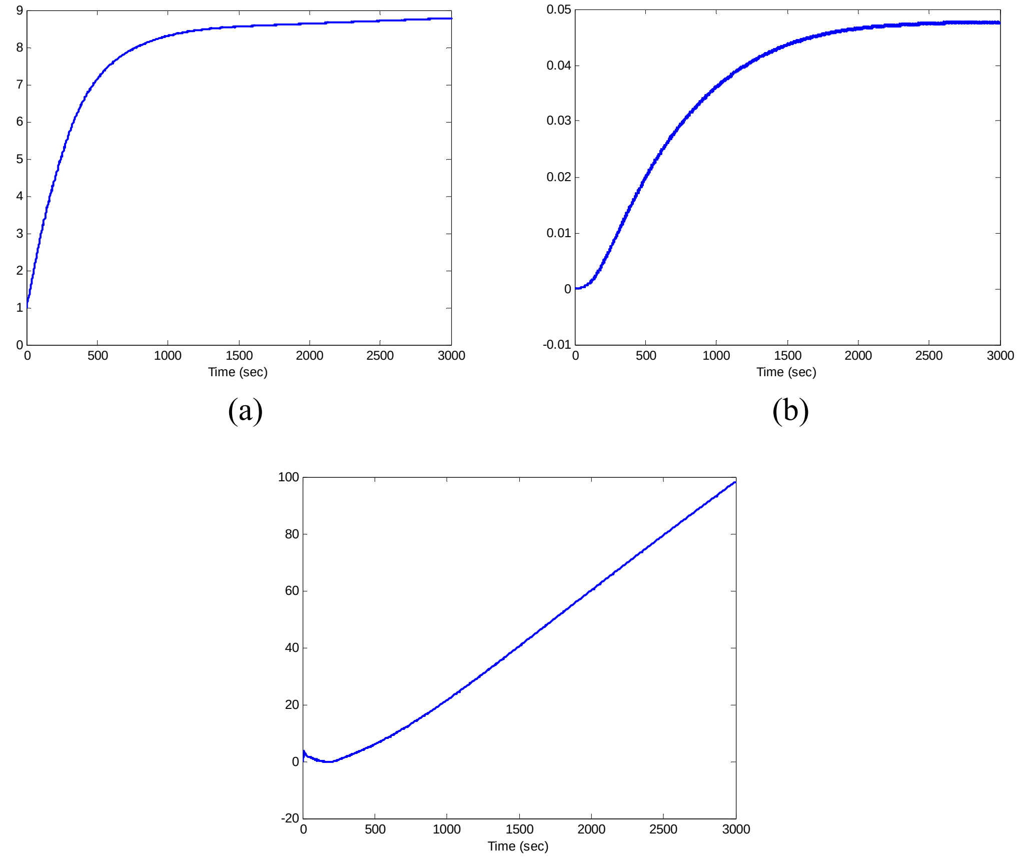
(c) (d)

(e) (f)

**Figure 5.46 (a)** Initial Cycle, **(b)-(e)** Tracking Plots and **(f)** Final Cycle

(Input SNR = 41.44)

The final values of the PID terms were $K_P = 8.787$, $K_I = 0.0475$, $K_D = 98.205$. Figure 5.47 (a)-(c) shows the plots of the PID terms during the simulation.



(a)

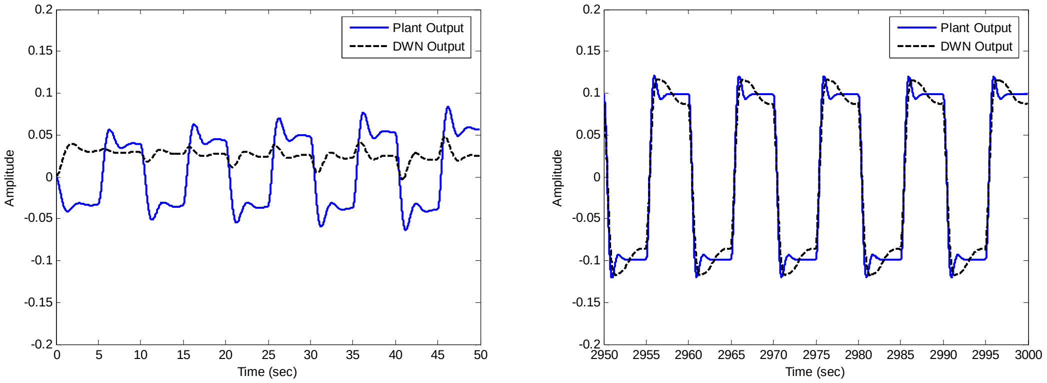
(b)

(c)

**Figure 5.47 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

(Input SNR = 41.44)

Figure 5.48 (a) shows the DWN output during the first five cycles and 5.48 (b) shows the DWN output during the last five cycles.

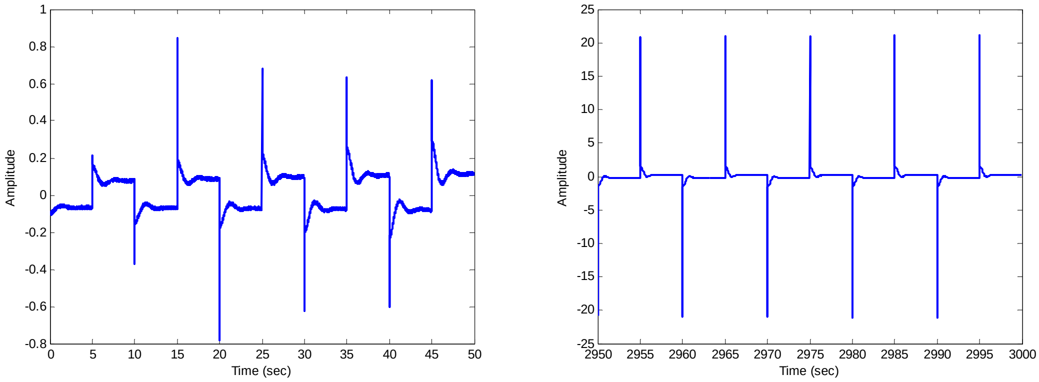


(a)

(b)

**Figure 5.48 (a)-(b)** DWN Output during Simulation (Input SNR = 41.44)

Figure 5.49 (a) shows the controller output during the first five cycles and 5.49 (b) shows the controller output during the last five cycles.



(a) (b)

**Figure 5.49 (a)-(b)** Controller Output during Simulation (Input SNR = 41.44)

The controller displayed similar behavior to the noise-free case. However, the final value of the integral term was smaller. As a result, there was a small steady state error by the end of the simulation. Thus, an input signal having a SNR of 41.44 had a minor effect on the performance of the PID-DWN controller.

For the case of output noise, the first noise signal had a mean squared value of 1.2 x$10^{-8}$, shown in Figure 5.50. This resulted in the output signal having a SNR of 58.65.

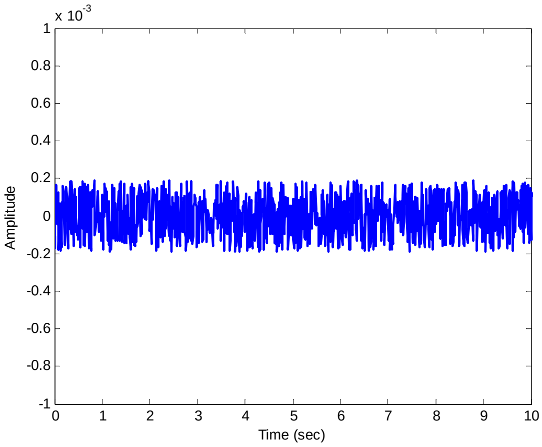
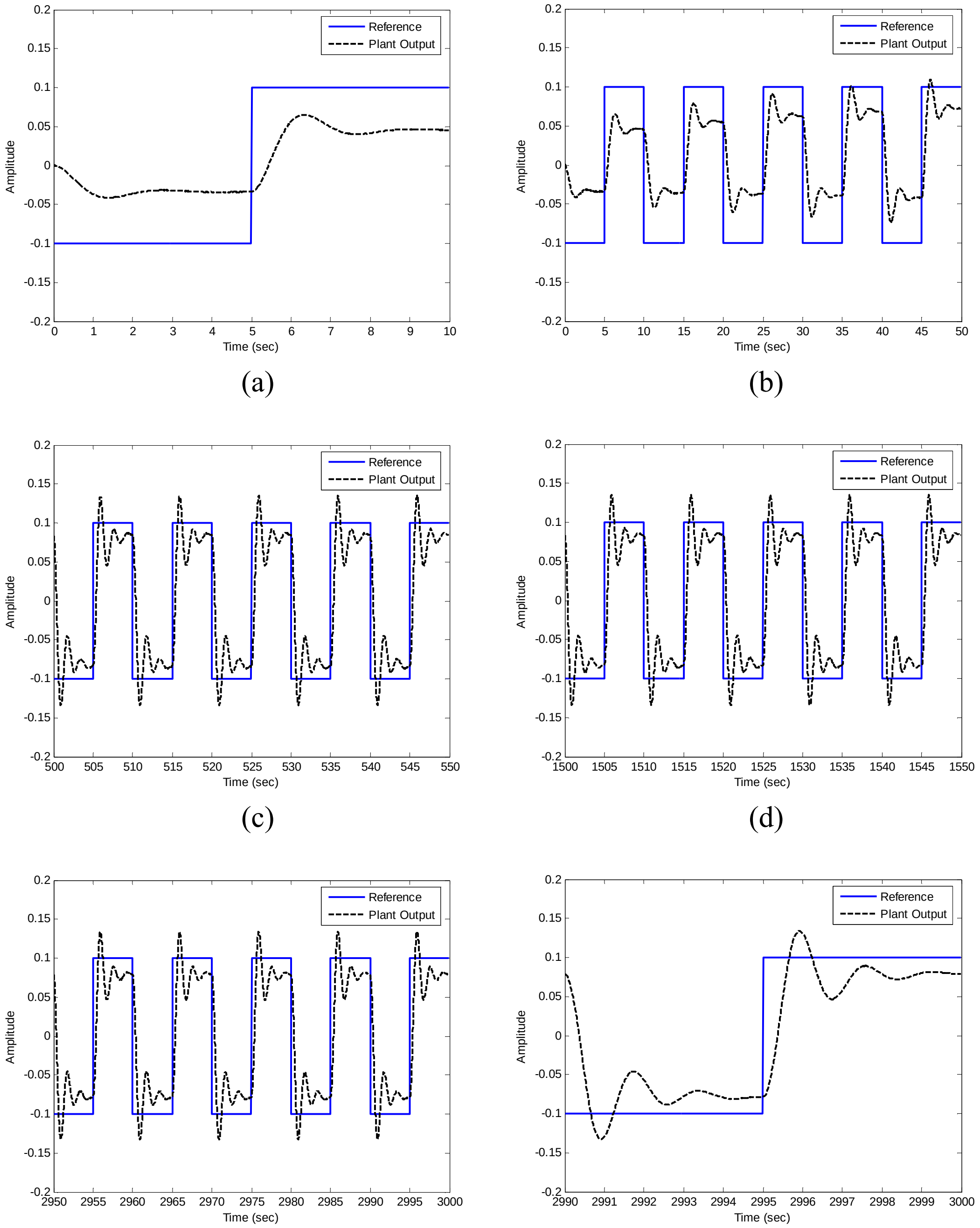


**Figure 5.50** Output Noise (MS = 1.2 x$10^{-8}$)

Figure 5.51 (a) shows the initial cycle in the simulation. Figure 5.51 (b)-(e) shows the tracking performance of the controller in five cycle long plots. Figure 5.51 (f) shows the final cycle in the simulation.



(a) (b)

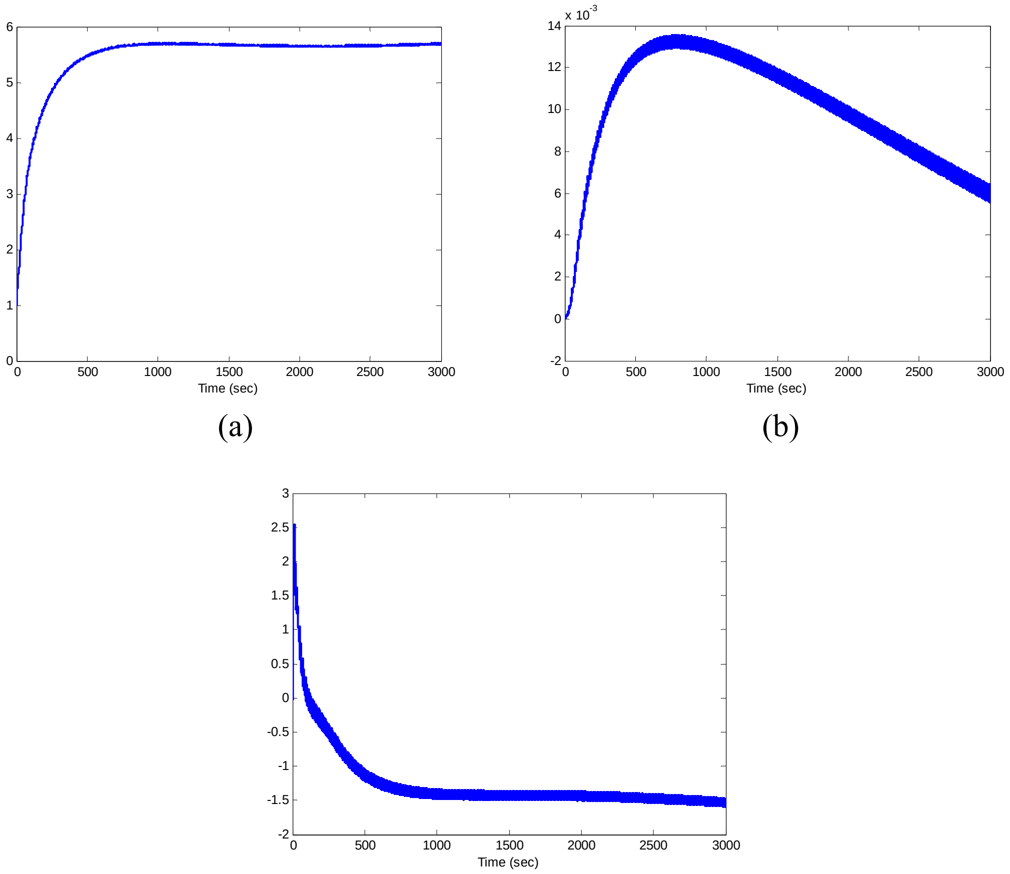
(c) (d)

(e) (f)

**Figure 5.51 (a)** Initial Cycle, **(b)-(e)** Tracking Plots and **(f)** Final Cycle

(Output SNR = 58.65)

The final values of the PID terms were $K_P = 5.697$, $K_I = 0.0056$, $K_D = -1.581$. Figure 5.52 (a)-(c) shows the plots of the PID terms during the simulation.



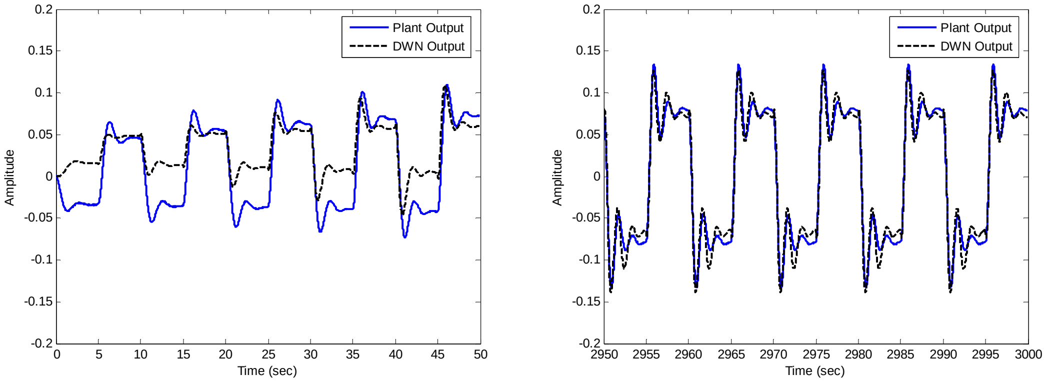
(a) (b)

(c)

**Figure 5.52 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

(Output SNR = 58.65)

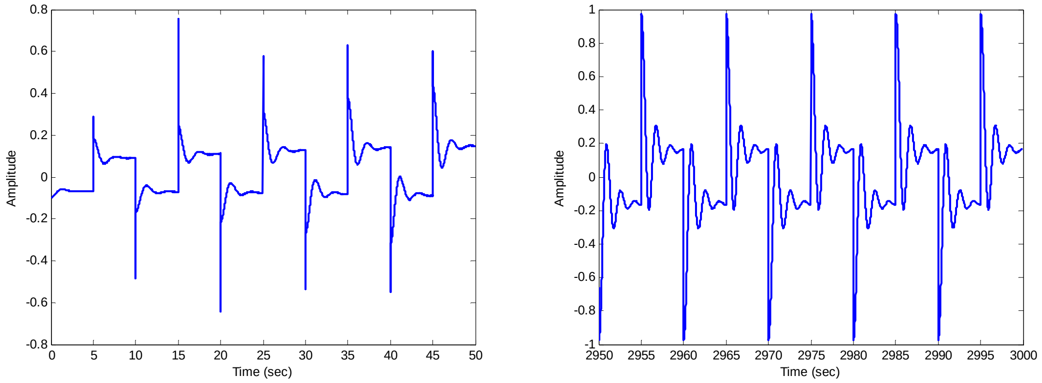
Figure 5.53 (a) shows the DWN output during the first five cycles and 5.53 (b) shows the DWN output during the last five cycles.



(a) (b)

**Figure 5.53 (a)-(b)** DWN Output during Simulation (Output SNR = 58.65)

Figure 5.54 (a) shows the controller output during the first five cycles and 5.54 (b) shows the controller output during the last five cycles.

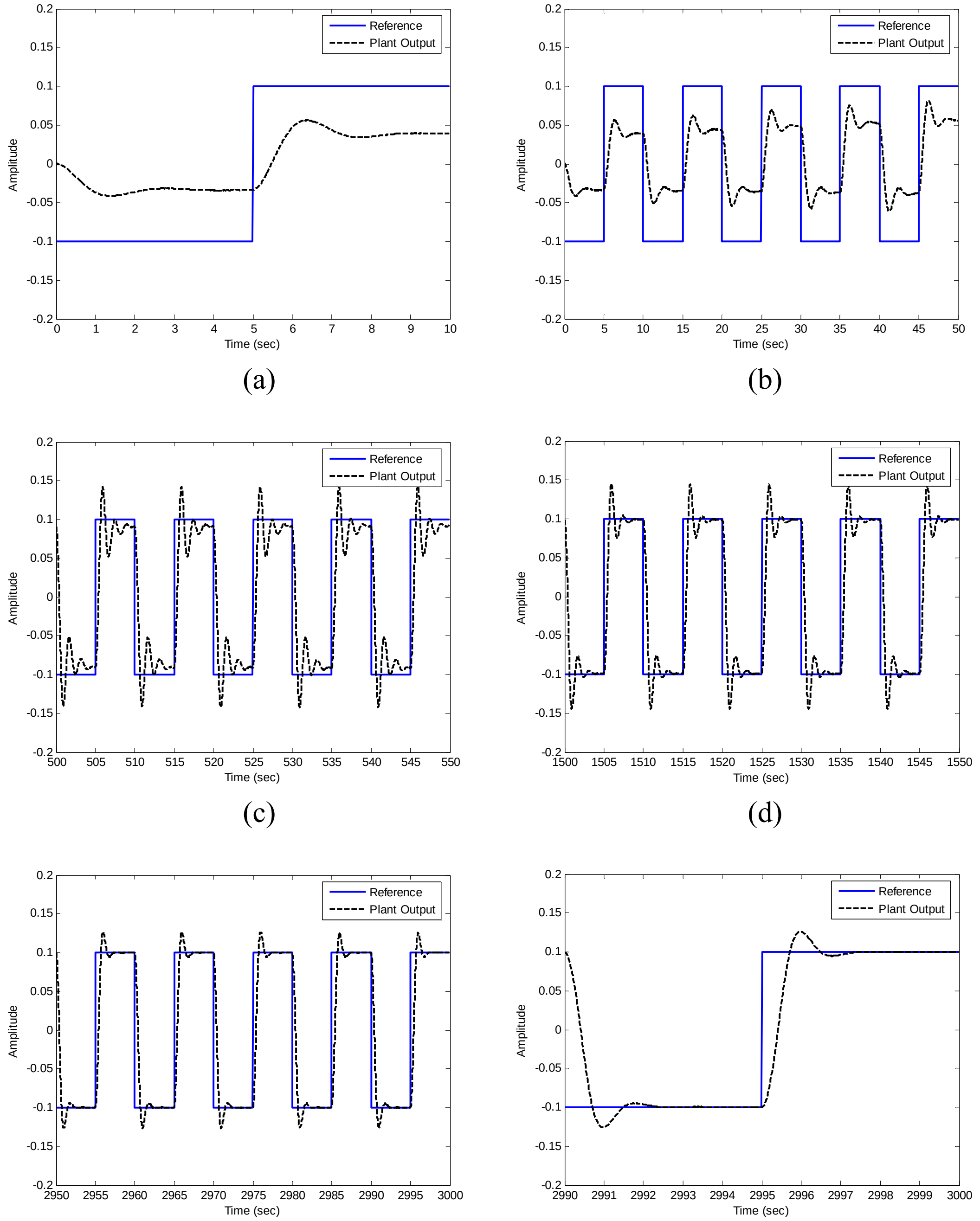


(a) (b)

**Figure 5.54 (a)-(b)** Controller Output during Simulation (Output SNR = 58.65)

The controller did not display similar behavior to the noise-free case. The final values of the PID terms were smaller. As a result, the oscillations were not minimized and there was a big steady state error by the end of the simulation. Thus, an output signal having a SNR of 58.65 had a major effect on the performance of the PID-DWN controller.

The simulation was repeated with the DWN trained on noise-free data as in subsection 4.5.2. This simulated the process of filtering the training data prior to modeling the plant. Figure 5.55 (a) shows the initial cycle in the simulation. Figure 5.55 (b)-(e) shows the tracking performance of the controller in five cycle long plots. Figure 5.55 (f) shows the final cycle in the simulation.



(a) (b)

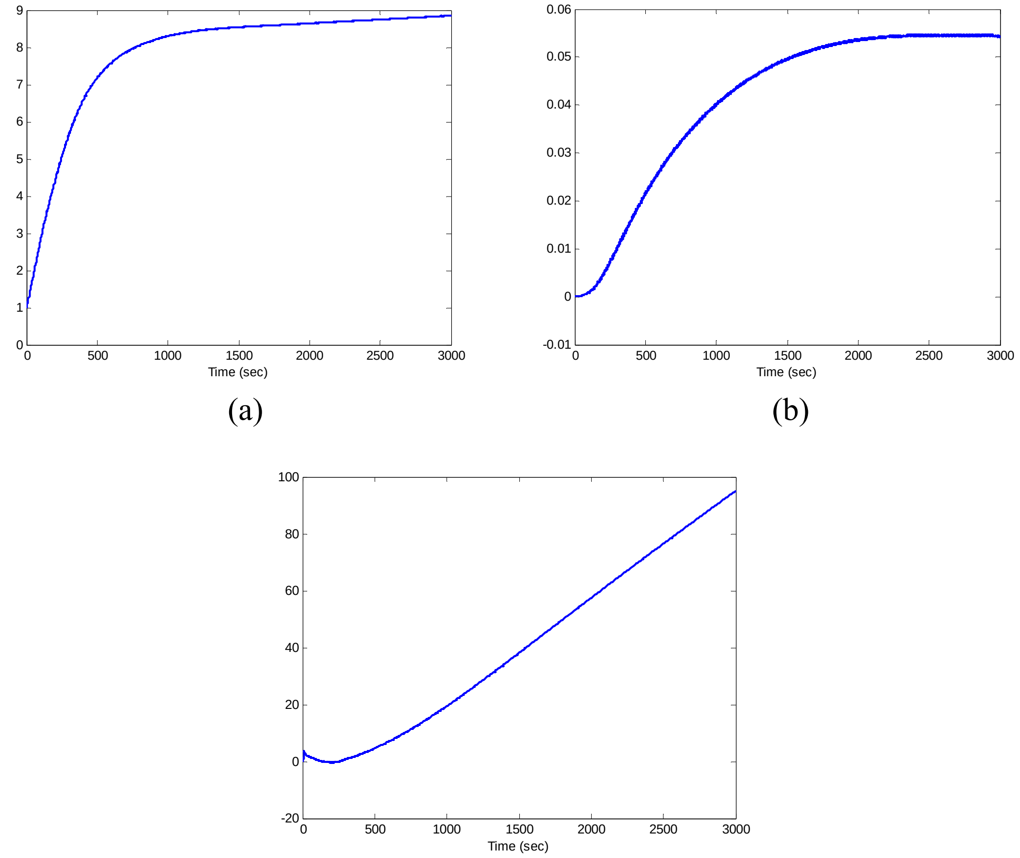
(c) (d)

(e) (f)

**Figure 5.55 (a)** Initial Cycle, **(b)-(e)** Tracking Plots and **(f)** Final Cycle

(Output SNR = 58.65)

The final values of the PID terms were $K_P = 8.854$, $K_I = 0.0543$, $K_D = 94.889$. Figure 5.56 (a)-(c) shows the plots of the PID terms during the simulation.



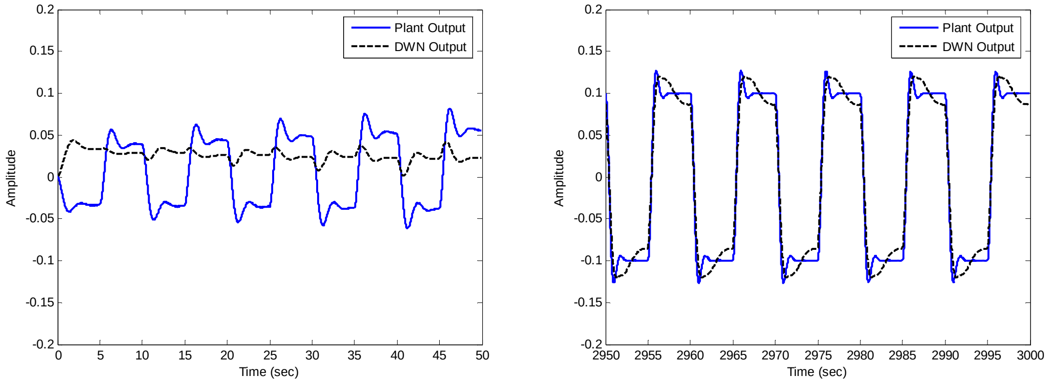
(a) (b)

(c)

**Figure 5.56 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

(Output SNR = 58.65)

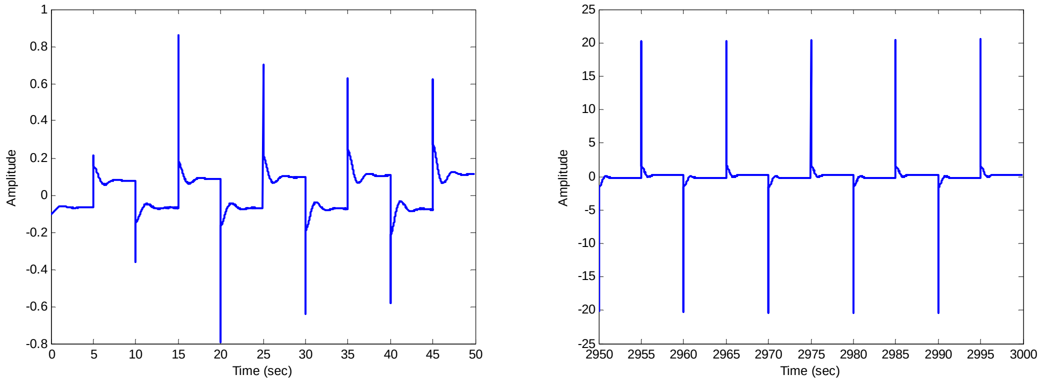
Figure 5.57 (a) shows the DWN output during the first five cycles and 5.57 (b) shows the DWN output during the last five cycles.



(a) (b)

**Figure 5.57 (a)-(b)** DWN Output during Simulation (Output SNR = 58.65)

Figure 5.58 (a) shows the controller output during the first five cycles and 5.58 (b) shows the controller output during the last five cycles.



(a) (b)

**Figure 5.58 (a)-(b)** Controller Output during Simulation (Output SNR = 58.65)

The controller displayed similar behavior to the noise-free case with a minor difference in the final value of the derivative term. Thus, an output signal having a SNR of 58.65 had no effect on the performance of the PID-DWN controller when the DWN was trained on noise-free data.

The second noise signal had a mean squared value of 1.2 $x10^{-6}$, shown in Figure 5.59. This resulted in the output signal having a SNR of 38.65.

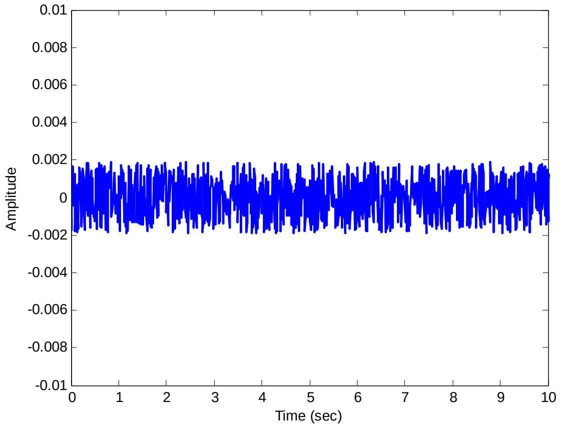
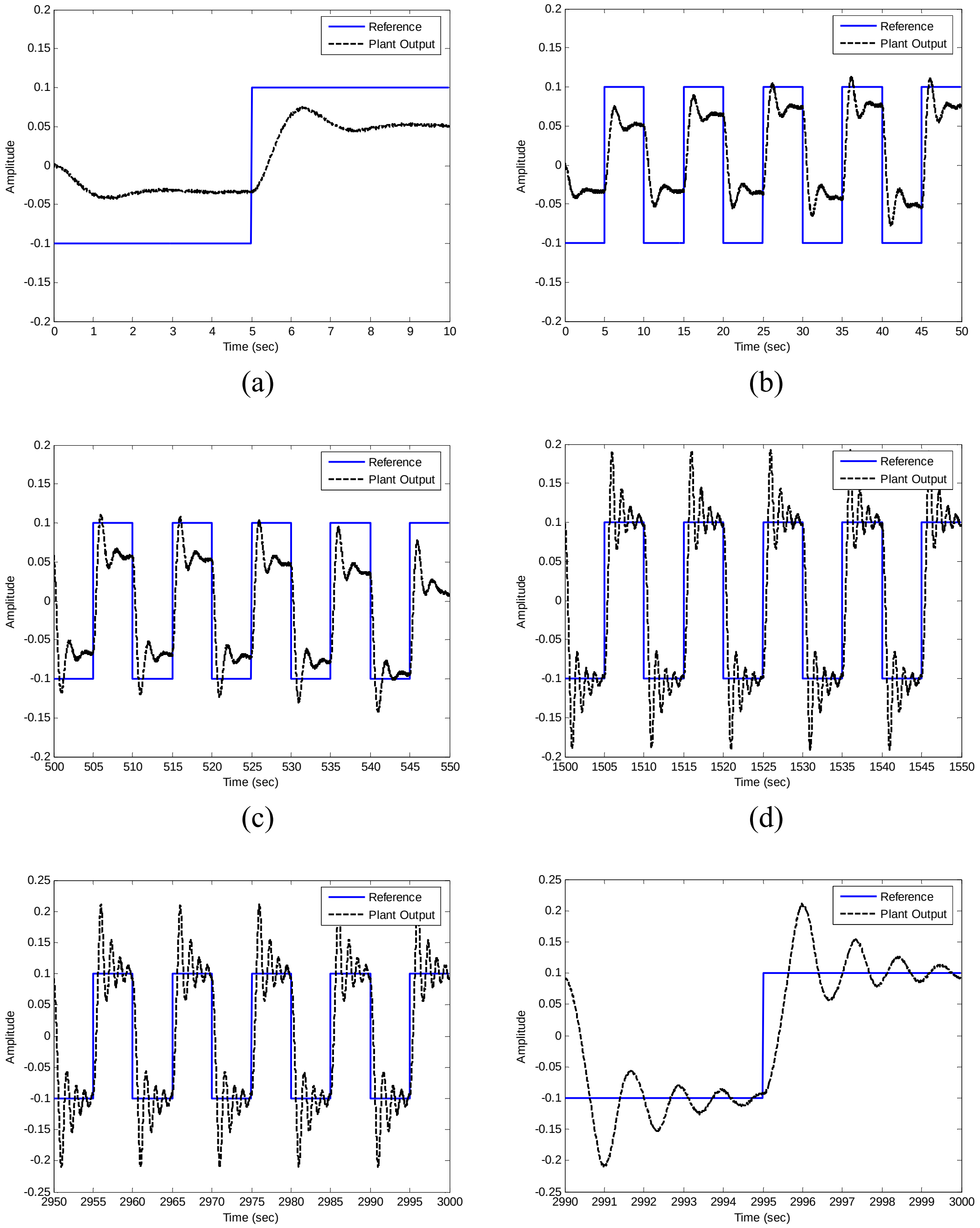


**Figure 5.59** Output Noise (MS = 1.2 $x10^{-6}$)

Figure 5.60 (a) shows the initial cycle in the simulation. Figure 5.60 (b)-(e) shows the tracking performance of the controller in five cycle long plots. Figure 5.60 (f) shows the final cycle in the simulation.



(a) (b)

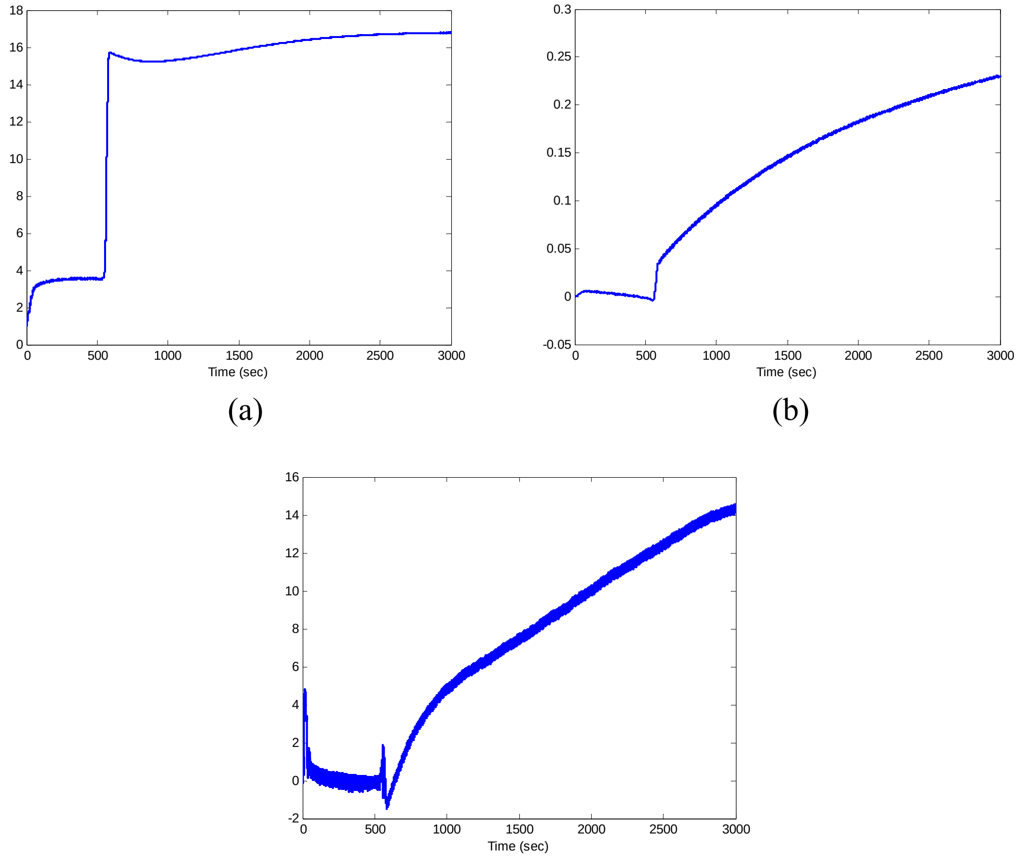
(c) (d)

(e) (f)

**Figure 5.60 (a)** Initial Cycle, **(b)-(e)** Tracking Plots and **(f)** Final Cycle

(Output SNR = 38.65)

The final values of the PID terms were $K_P = 16.792$, $K_I = 0.23$, $K_D = 14.507$. Figure 5.61 (a)-(c) shows the plots of the PID terms during the simulation.



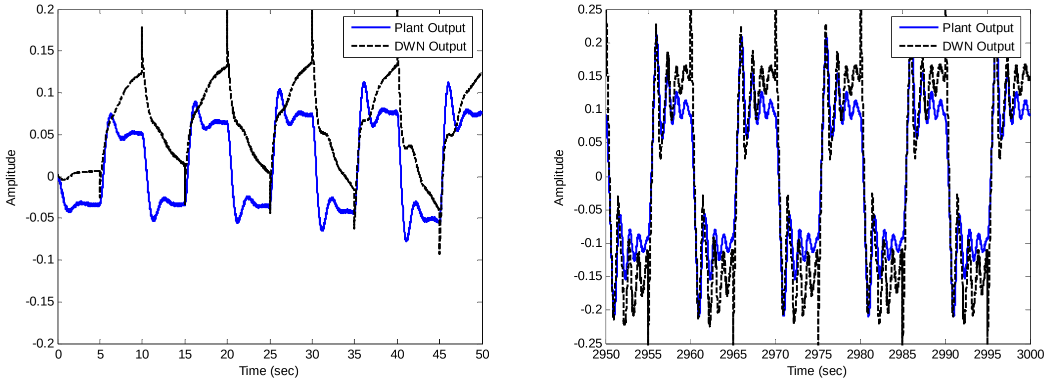
(a) (b)

(c)

**Figure 5.61 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

(Output SNR = 38.65)

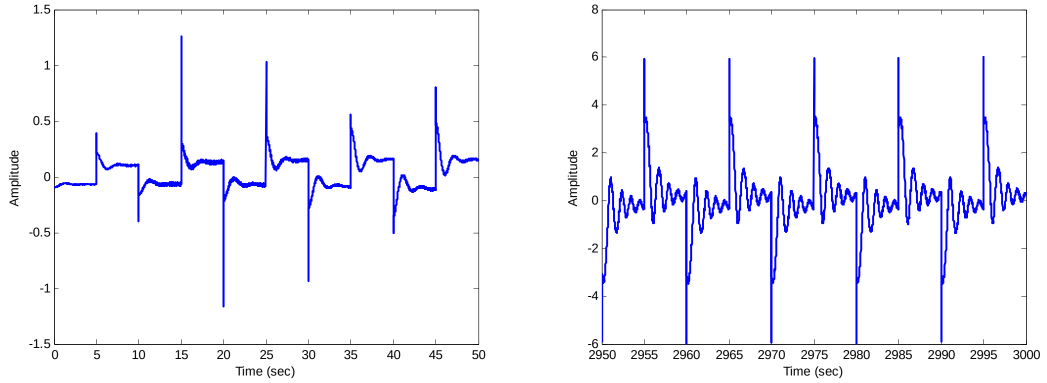
Figure 5.62 (a) shows the DWN output during the first five cycles and 5.62 (b) shows the DWN output during the last five cycles.



(a) (b)

**Figure 5.62 (a)-(b)** DWN Output during Simulation (Output SNR = 38.65)

Figure 5.63 (a) shows the controller output during the first five cycles and 5.63 (b) shows the controller output during the last five cycles.

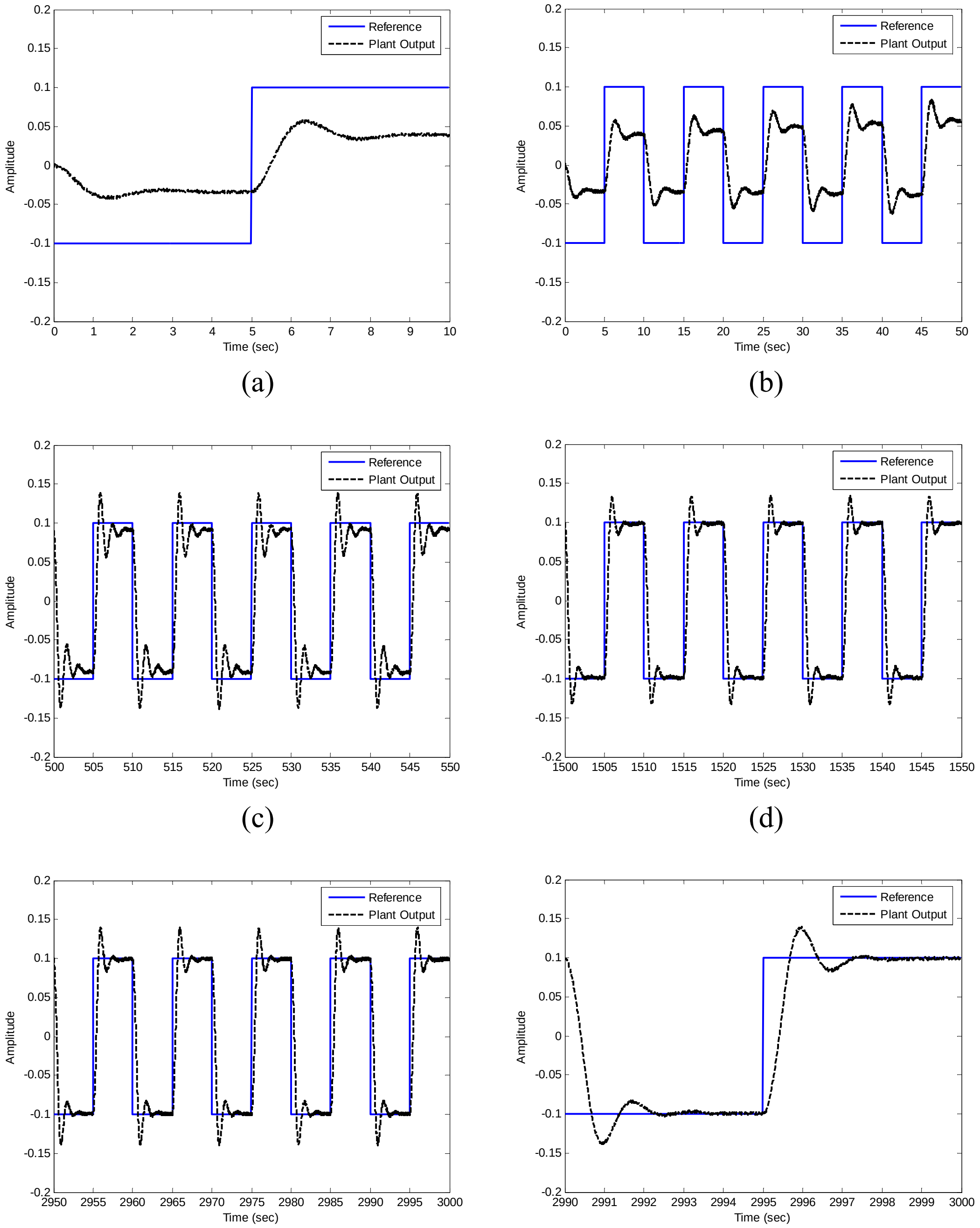


(a) (b)

**Figure 5.63 (a)-(b)** Controller Output during Simulation (Output SNR = 38.65)

The controller did not display similar behavior to the noise-free case. The final values of the proportional and integral terms were larger, while the derivative term was smaller. As a result, the plant output had a large overshoot and many oscillations by the end of the simulation. Thus, an output signal having a SNR of 38.65 had a major effect on the performance of the PID-DWN controller.

The simulation was repeated with the DWN trained on noise-free data as in subsection 4.5.2. This simulated the process of filtering the training data prior to modeling the plant. Figure 5.64 (a) shows the initial cycle in the simulation. Figure 5.64 (b)-(e) shows the tracking performance of the controller in five cycle long plots. Figure 5.64 (f) shows the final cycle in the simulation.



(a) (b)

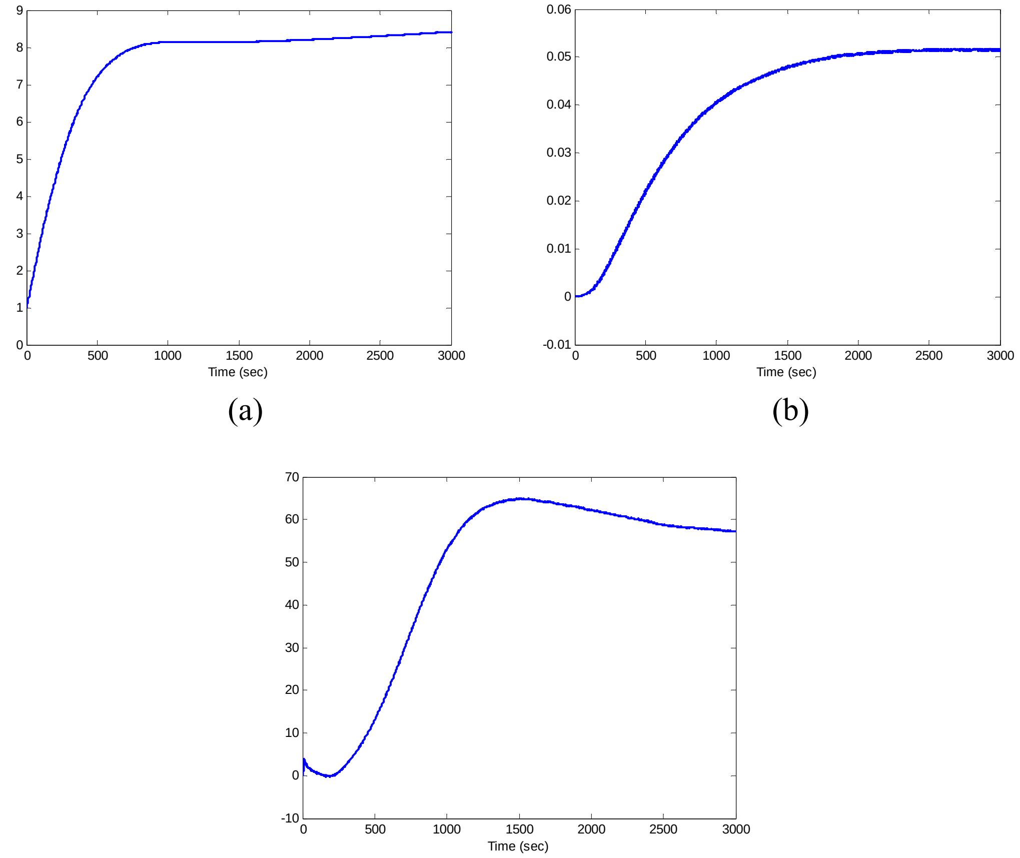
(c) (d)

(e) (f)

**Figure 5.64 (a)** Initial Cycle, **(b)-(e)** Tracking Plots and **(f)** Final Cycle

(Output SNR = 38.65)

The final values of the PID terms were $K_P = 8.42$, $K_I = 0.0513$, $K_D = 57.364$. Figure 5.65 (a)-(c) shows the plots of the PID terms during the simulation.



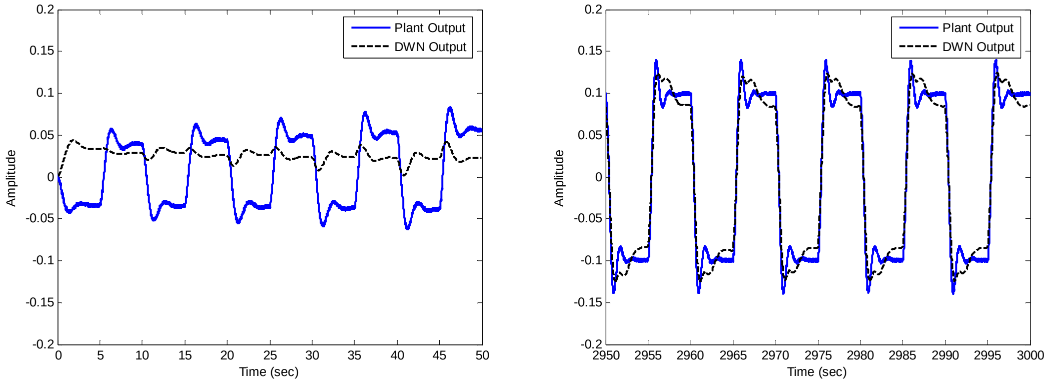
(a) (b)

(c)

**Figure 5.65 (a)** Proportional, **(b)** Integral and **(c)** Derivative Terms

(Output SNR = 38.65)

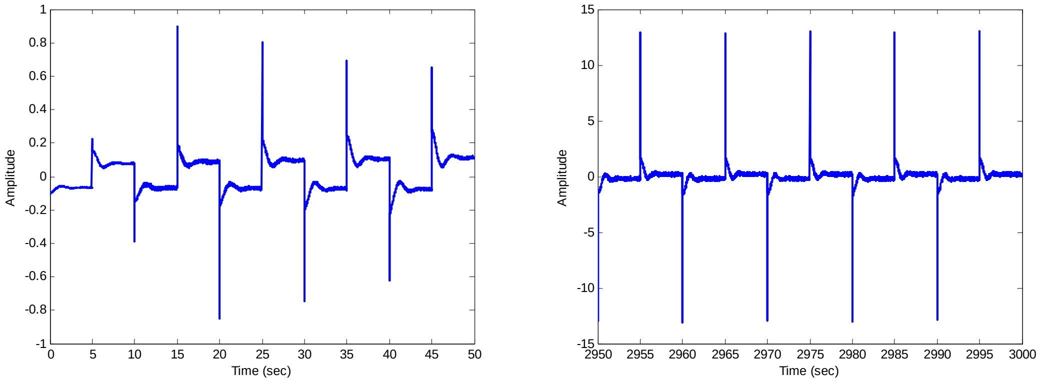
Figure 5.66 (a) shows the DWN output during the first five cycles and 5.66 (b) shows the DWN output during the last five cycles.



(a) (b)

**Figure 5.66 (a)-(b)** DWN Output during Simulation (Output SNR = 38.65)

Figure 5.67 (a) shows the controller output during the first five cycles and 5.67 (b) shows the controller output during the last five cycles.



(a) (b)

**Figure 5.67 (a)-(b)** Controller Output during Simulation (Output SNR = 38.65)

The controller displayed similar behavior to the noise-free case. However, the final value of the derivative term was smaller. As a result, by the end of the simulation, the oscillations were not minimized as well as the noise-free case. Thus, an output signal having a SNR of 38.65 had a minor effect on the performance of the PID-DWN controller when the DWN was trained on noise-free data.

## 5.3 Conclusion

When implemented on a first order nonlinear system, the PID-DWN controller successfully tracked the sinusoidal and step reference signals by tuning the PID terms. In the cases of input and output noise contamination, performance of the controller was shown to be unaffected by input noise resulting in SNR values of 61.63 dB and 41.63 dB, and by output noise resulting in SNR values of 59.35 dB and 39.35 dB.

When implemented on a second order nonlinear system, the PID-DWN controller successfully tracked the sinusoidal and step reference signals by tuning the PID terms. In the case of input noise contamination, performance of the controller was shown to be unaffected by input noise resulting in a SNR of 61.44 dB, while being slightly affected by input noise resulting in a SNR of 41.44 dB. However, in the case of output noise contamination, performance of the controller was shown to be greatly affected by output noise resulting in SNR values of 58.65 dB and 38.65 dB.

When the controller used a DWN trained on noise-free data, which simulated the process of filtering the training data prior to modeling the plant, performance of the controller was shown to be unaffected by output noise resulting in SNR of 58.65 dB, and slightly affected by output noise resulting in SNR of 38.65 dB. Therefore, performance of the controller was shown to be directly related to off-line training results of the DWN.

# CHAPTER 6

# CONCLUSION

This work investigated wavelet networks for system identification and control. The first part started with an introduction to wavelets and the wavelet transform. This was followed by an introduction to wavelet networks and the new wavelet network architecture, the dynamic wavelet network (DWN). Their architecture, training process and initialization methods were explained. The identification of nonlinear systems was demonstrated first in the scope of static modeling. The heuristic method and the selection method were compared in one dimensional and two dimensional function approximation examples. The selection method was shown to be superior to the heuristic method in both examples. The superiority of wavelet networks over neural networks in nonlinear function approximation was also demonstrated. The focus then moved to the use of wavelet networks in dynamic modeling of nonlinear systems. The DWN was compared to the conventional WN scheme in the dynamic modeling of first order and second order nonlinear systems with input saturation. The DWN exhibited faster convergence speed than the conventional WN when used in the dynamic modeling of first order nonlinear systems, while when used in the dynamic modeling of second order nonlinear systems, both schemes had about the same convergence speed. Furthermore, the DWN enabled a method in which to approximate the level of saturation, thus providing additional information about the system. This was an advantage to using the DWN over the conventional WN scheme. The superiority of wavelet networks over neural networks in dynamic modeling of nonlinear systems was also demonstrated. The effect of noise on the DWN was investigated using three successively increasing noise levels. In the case of input noise contamination, dynamic modeling of first order and second order nonlinear systems was virtually unaffected at all noise levels. However, in the case of output noise contamination, dynamic modeling of first order nonlinear systems was unaffected at all noise levels, while dynamic modeling of second order nonlinear systems was greatly affected.

The second part addressed the design of an adaptive PID controller, based on a self-tuning dynamic wavelet network. The controller was implemented on first order and second order nonlinear systems by tracking two reference signals: a sinusoid and a step. In both cases, the PID-DWN controller successfully tracked the reference signals by tuning the PID terms. The effect of noise on the controller was investigated for two successively increasing noise levels. In the case of input noise contamination, control of first order nonlinear systems was unaffected at both noise levels, while control of second order nonlinear systems was unaffected at the lower noise level and was minimally affected at the higher noise level. In the case of output noise contamination, control of first order nonlinear systems was unaffected at both noise levels, while control of second order nonlinear systems was greatly affected. This was due to the results of off-line training of the DWN in the presence of output noise. When the controller was implemented with a DWN trained on noise-free data, simulating the process of filtering the training data prior to modeling the plant, the performance of the controller improved significantly. As a result, the controller was unaffected at the lower noise level and was minimally affected at the higher noise level.

The contributions of this work have been: i) the introduction of the dynamic wavelet network scheme and its training mechanism and ii) the implementation of the DWN in an adaptive PID controller and defining the adaptive mechanism for the PID parameters. The outcome of this work has shown the effectiveness of the DWN in the modeling and control of first order and second order nonlinear dynamic systems.

# REFERENCE LIST

[1]. S. S. Iyengar, E.C. Cho, V.V. Phoha, *Foundations of Wavelet Networks and Applications*, Chapman & Hall/CRC, June 2002.

[2]. Y.C. Pati and P.S. Krishnaprasad, "Discrete Affine Wavelet Transforms for Analysis and Synthesis of Feedforward Neural Networks", *Advances in Neural Information Processing Systems*, Vol. 3, 1991, pp. 743-49

[3]. Y.C. Pati and P.S. Krishnaprasad, "Analysis and Synthesis of Feedforward Neural Networks Using Discrete Affine Wavelet Transformations", *IEEE Trans. on Neural Networks*, Vol. 4, 1992, pp. 73-85

[4]. Q. Zhang, A. Benveniste, "Wavelet Networks", *IEEE Trans. on Neural Networks*, Vol. 3, Nov. 1992, pp.889-98.

[5]. H.H. Szu, B. Telfer, and S. Kadambe, "Neural network adaptive wavelets for signal representation and classification", *Optical Engineering*, No. 9, Vol. 31, Sept. 1992, pp. 1907-16.

[6]. Q. Zhang, "Regressor Selection and Wavelet Network Construction", INRIA, *Signal Processing, Automatic and Computer-Integrated Manufacturing*, Project AS, No. 1967, April 1993, pp. 1-21.

[7]. Y. Zhuang, J.S. Baras, "Identification of Infinite Dimensional Systems Via Adaptive Wavelet Neural Networks", Institute for Systems Research and Department of Electrical Engineering, *Technical Research Report* T.R. 93-64, The University of Maryland, pp. 1-21.

[8]. Q. Zhang, "Using Wavelet Network in Nonparametric Estimation", *Signal Processing, Automatic and Computer-Integrated Manufacturing*, Project AS, No. 833, June 1994, pp. 1-44.

[9]. J.Fernando Marrar, E.C.B. Carvalho Filho, G.C. Vasconcelos. "Function Approximation by Polynomial Wavelets Generated from Powers of Sigmoids", SPIE Aerosense96, *10th Annual International Aeroscience Symposium*, Vol. 2762, Wavelet Application III, 1996, pp. 365-74.

[10]. S. Li, S. Chen, "Function Approximation Using Robust Wavelet Neural Networks", Department of Information Management, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, ROC, and School of Computer Science, Florida International University, Miami, FL., pp. 1-6.

[11]. M. Sgarbiy, V. Colla, L.M. Reyneri, "A Comparison between Weighted Radial Basis Functions and Wavelet Networks", Department of Electronics, Turin Polytechnic, Turin, Italy, 1998, pp.1-6.

[12]. Y. Oussar and G. Dreyfus, "Initialization by Selection for Wavelet Network Training", *Neurocomputing*, Vol. 34, 2000, pp. 131-43.

[13]. L.F. Yeung and X.W. Li, "Multi-Input System Identification and its Applications Using Wavelet Constructive Method", *Proceedings of the IEEE Conference on Decision and Control*, Vol. 3, Dec. 11-13, 1996, pp. 3230-35.

[14]. Y. Oussar, I. Rivals, L. Personnaz, and G. Dreyfus, "Training Wavelet Networks for Nonlinear Dynamic Input-Output Modeling", *Neurocomputing*, Vol. 20, 1998, pp. 173-88.

[15]. J. Wang, F. Wang, J. Zhang, J. Zhang, "Intelligent Controller Using Neural Network", Department of Automatic Control, Northeastern University, Shenyang, Liaoning, 110006, China, 1995, pp.755-61.

[16]. S. Omatu, "Dynamical Systems Regulation by Neuro-Controllers", Faculty of Engineering, University of Tokushima, Tokushima, 770, Japan, 1995, pp.590-99.

[17]. G. Lekutai, "Adaptive Self-tuning Neuro Wavelet Network Controllers", Electrical Engineering Department, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, March 31, 1997, pp. 1-112.

[18]. Y.-M. Cheng, B.-S. Chen, F.-Y. Shiau, "Adaptive Wavelet Network Control Design for Nonlinear Systems", Department of Electrical Engineering, National Tsing-Hua University, Hsin-Chu, Taiwan, 1998, pp.783-99.

[19]. R.-J. Wai, J.-M. Chang, "Intelligent control of Induction Servo Motor Drive via Wavelet Neural Network", Department of Electrical Engineering, Yuan Ze University, Chung Li, 320, Taiwan, 2002, pp.67-76.

[20]. R.K. Young, *Wavelet Theory and its Applications*, Kluwer Academic Publishers, Boston, MA, 1993.

[21]. P. Goupillard, A. Grossmann, J. Morlet, "Cycle-Octave and Related Transforms in Seismic Signal Analysis", *Geoxploration*, Vol. 23, 1984, pp.85-102.

[22]. A. Grossmann, J. Morlet, "Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape", *SIAM J. Math. Anal.*, Vol. 15, No. 4, 1984, pp.723-36.

[23]. Y. Sheng, D. Roberge, H. H. Szu, "Optical Wavelet Transform", *Optical Engineering*, Vol. 31, No. 9, 1992, pp.1840-45.

[24]. R. K. Martinet, A. Grossmann, J. Morlet, "Analysis of Sound Patterns Through Wavelet Transforms", *International J. of Pattern Recognition and Artificial Intelligence*, Vol. 1, No. 2, 1987, pp.273-302.

[25]. J. M. Combes, A. Grossmann, Ph. Tchamitchian, *Wavelets: Time-Frequency Methods and Phase Space*, 2$^{nd}$ Edition, Springer-Verlag, New York, NY, 1989..

[26]. L.G. Weiss, "Wavelets and Wideband Correlation Processing", *IEEE Signal Processing Mag.*, Vol. 11, No. 1, 1994, pp.13-32.

[27]. R. Polikar, “The Wavelet Tutorial”, Department of Electrical and Computer Engineering, Rowan University, Glassboro, New Jersey, 2001, www.engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html.

# APPENDIX A

# SIMULINK MODELS

## First Order PID-DWN Controller

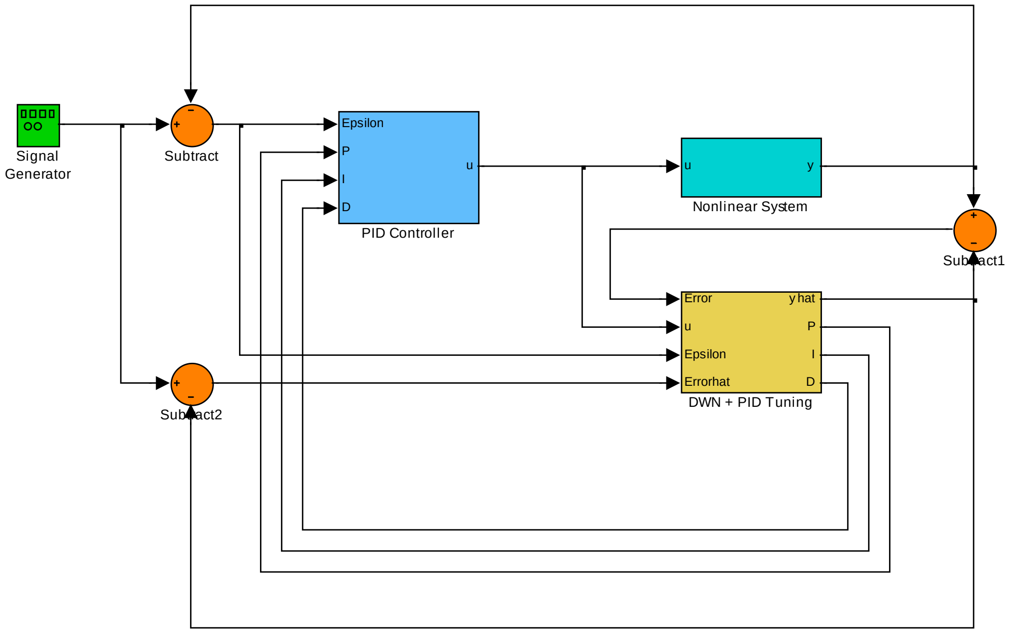


**Figure A1** First Order PID-DWN Controller

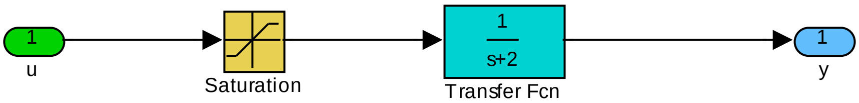


**Figure A2** First Order Nonlinear System

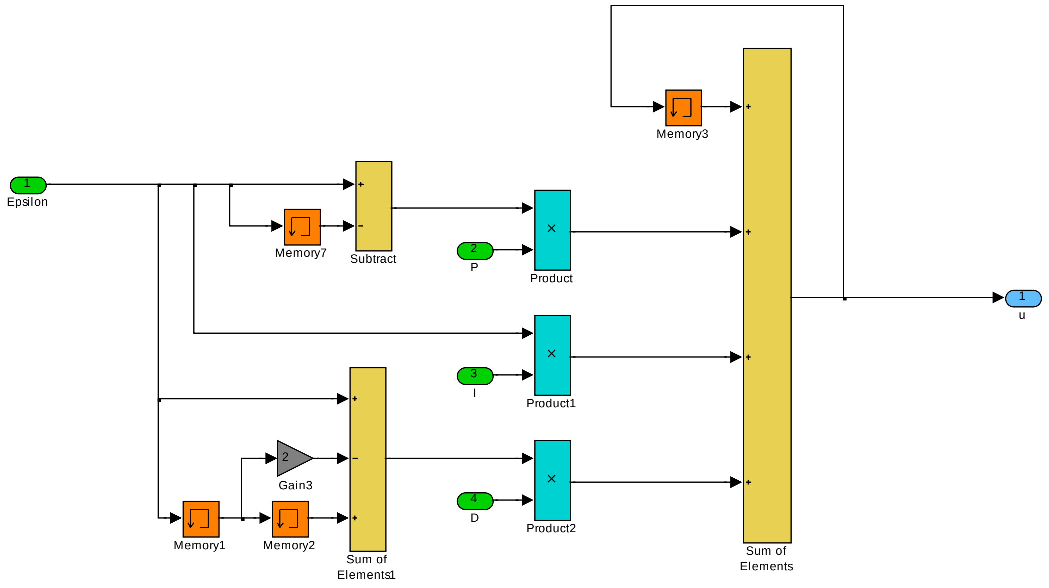


**Figure A3** PID Controller Block

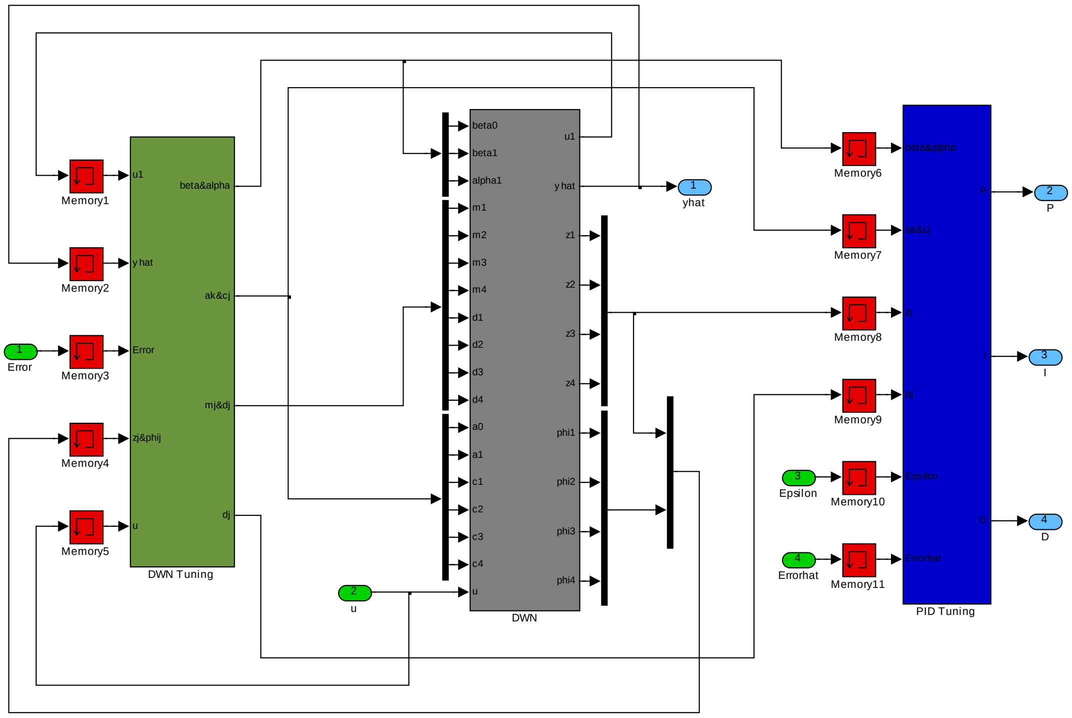


**Figure A4** PID and First Order DWN Tuning Block

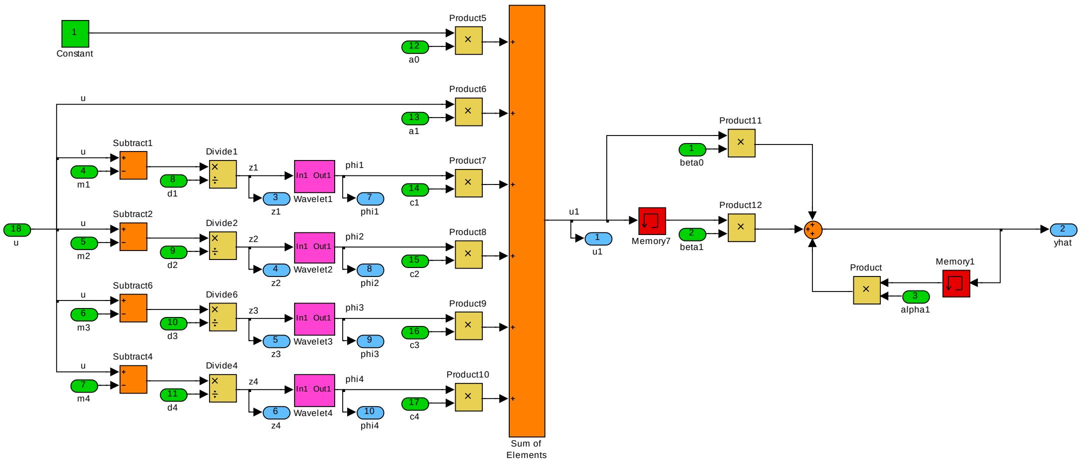


**Figure A5** First Order DWN

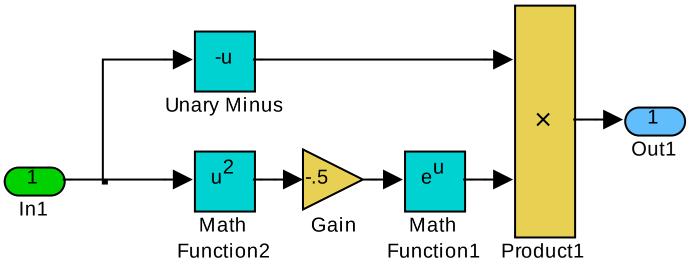


**Figure A6** Wavelet Neuron

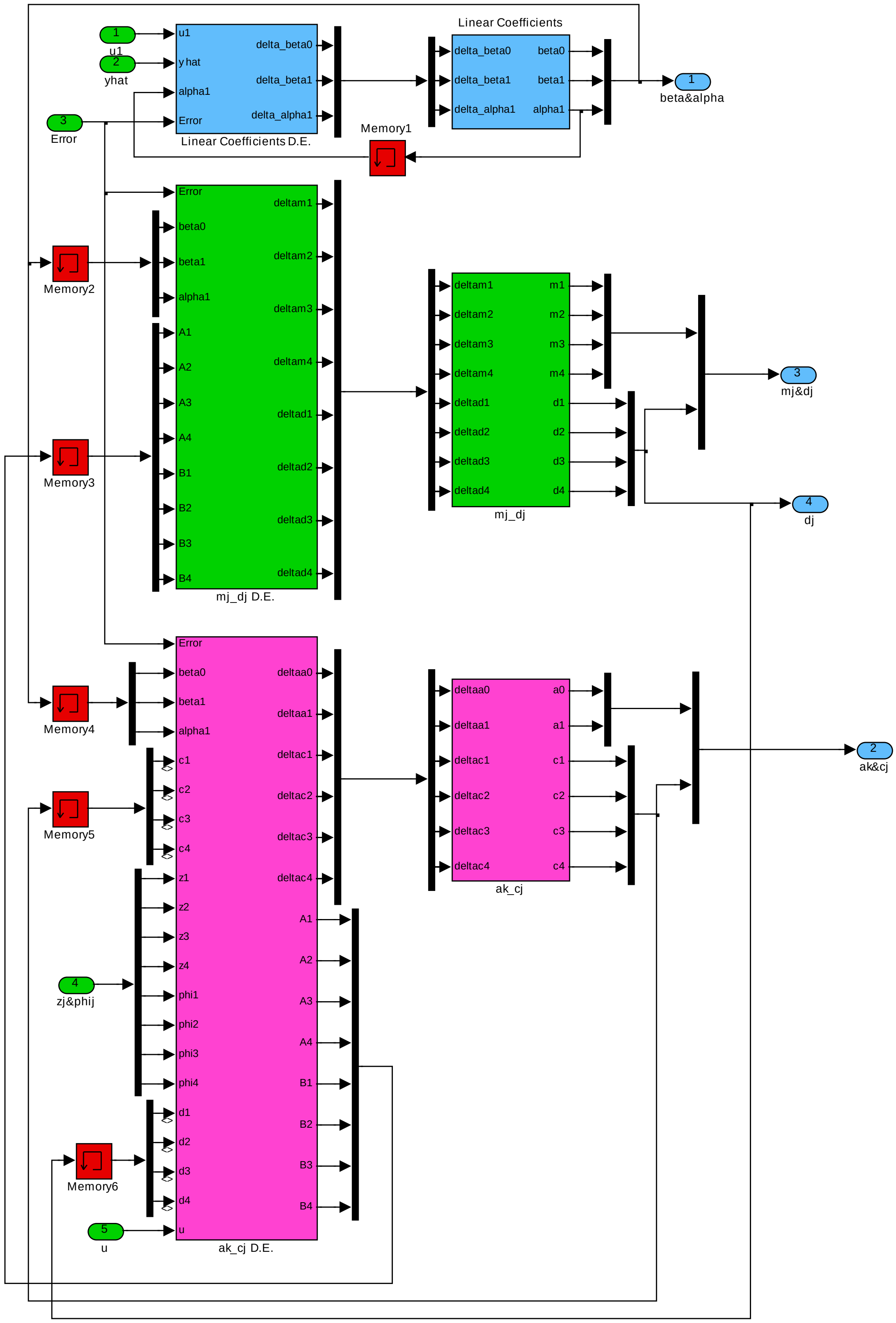


**Figure A7** First Order DWN Tuning Block

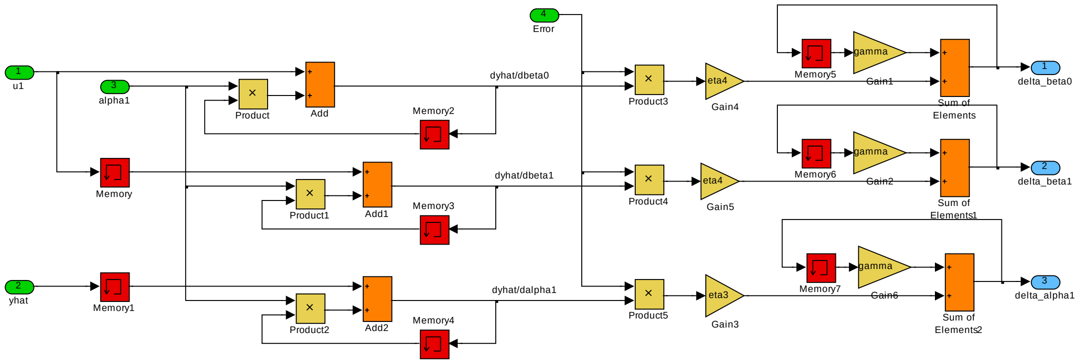


**Figure A8** Linear Coefficients Differential Equation Block

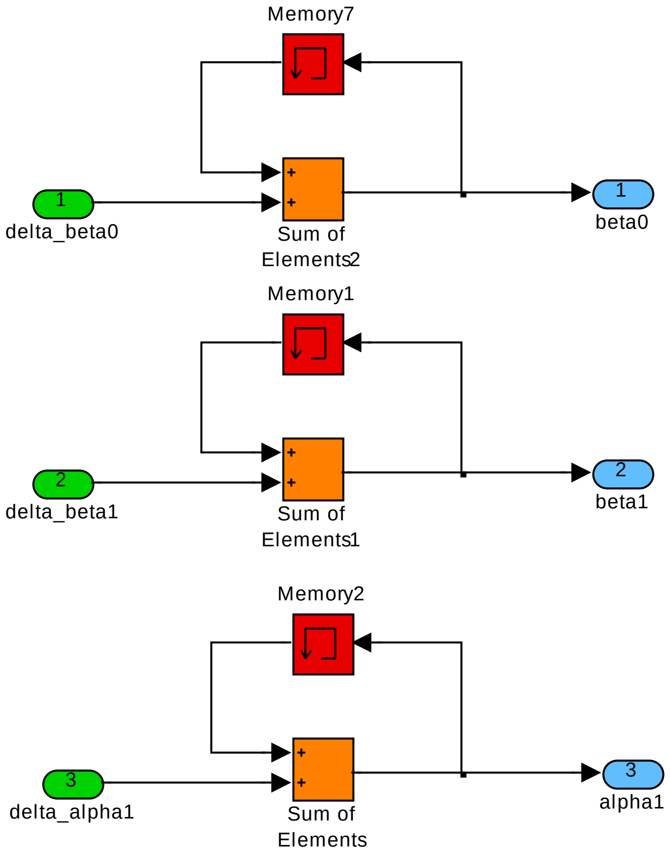


**Figure A9** Linear Coefficients Block

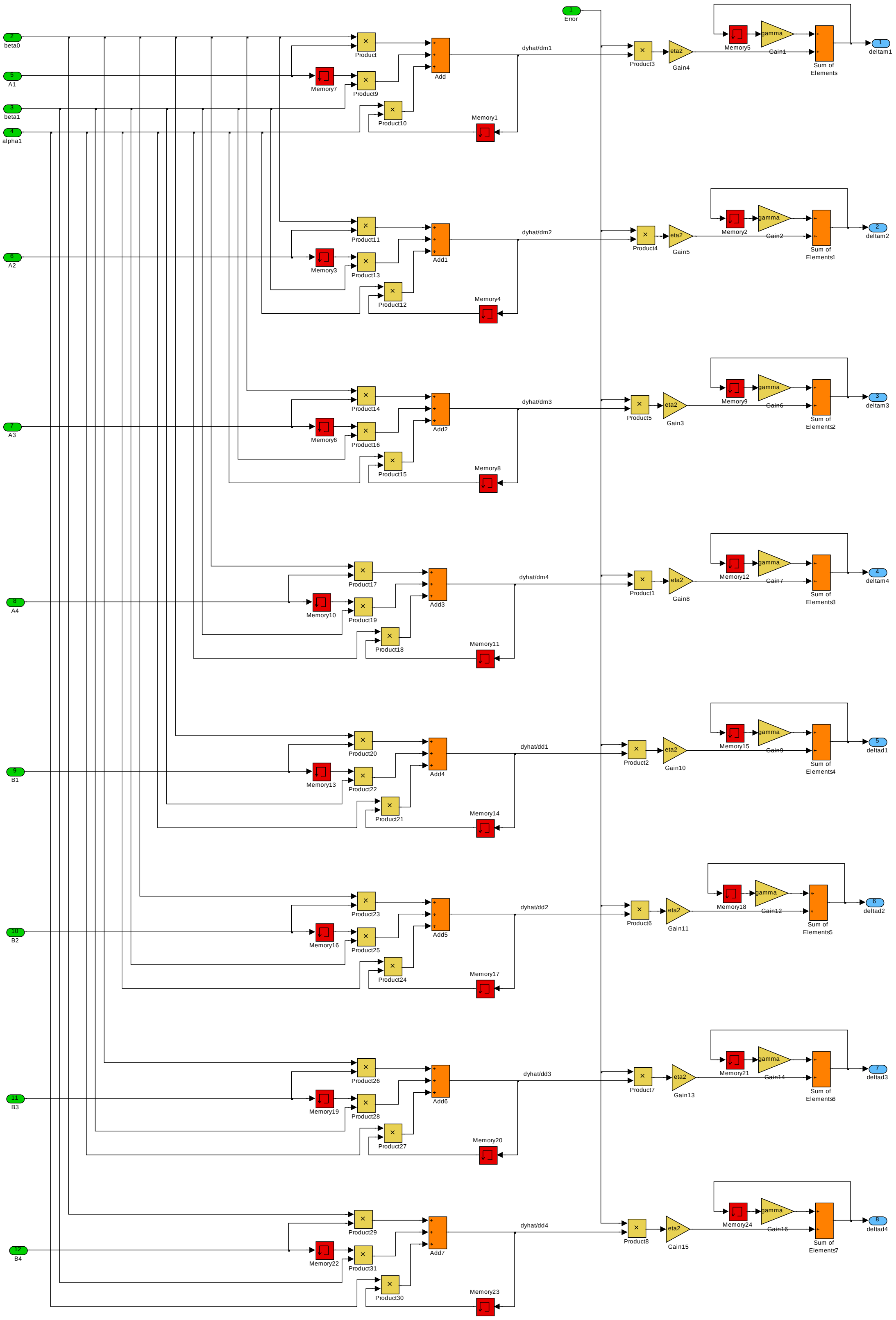


**Figure A10** Translations and Dilations Differential Equation Block

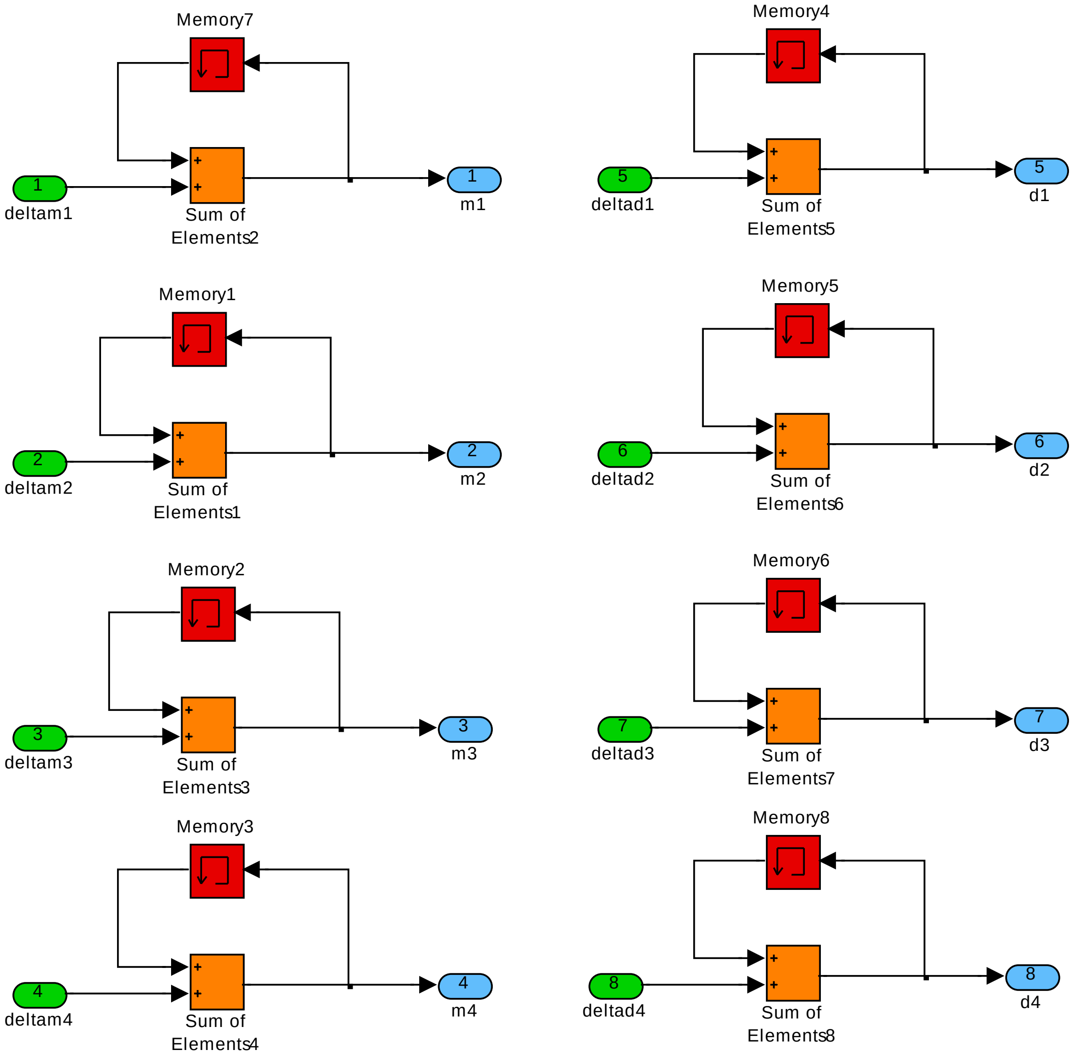


**Figure A11** Translations and Dilations Block

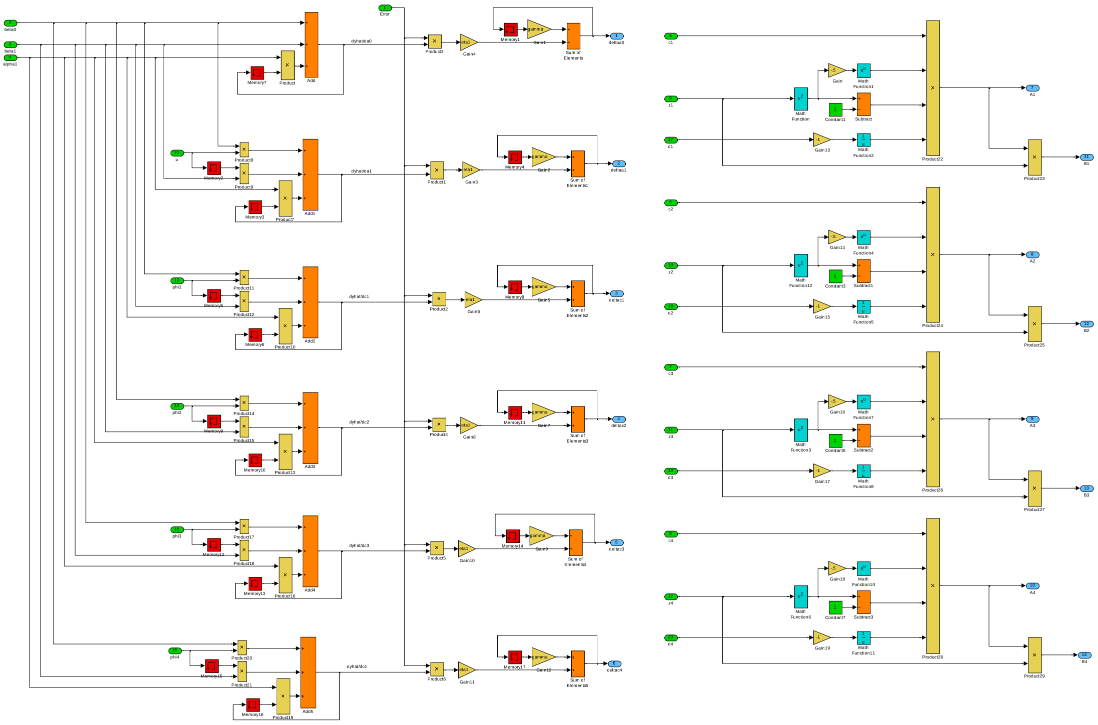


Figure A12 Direct Connections and Wavelet Neuron Weights D.E. Block

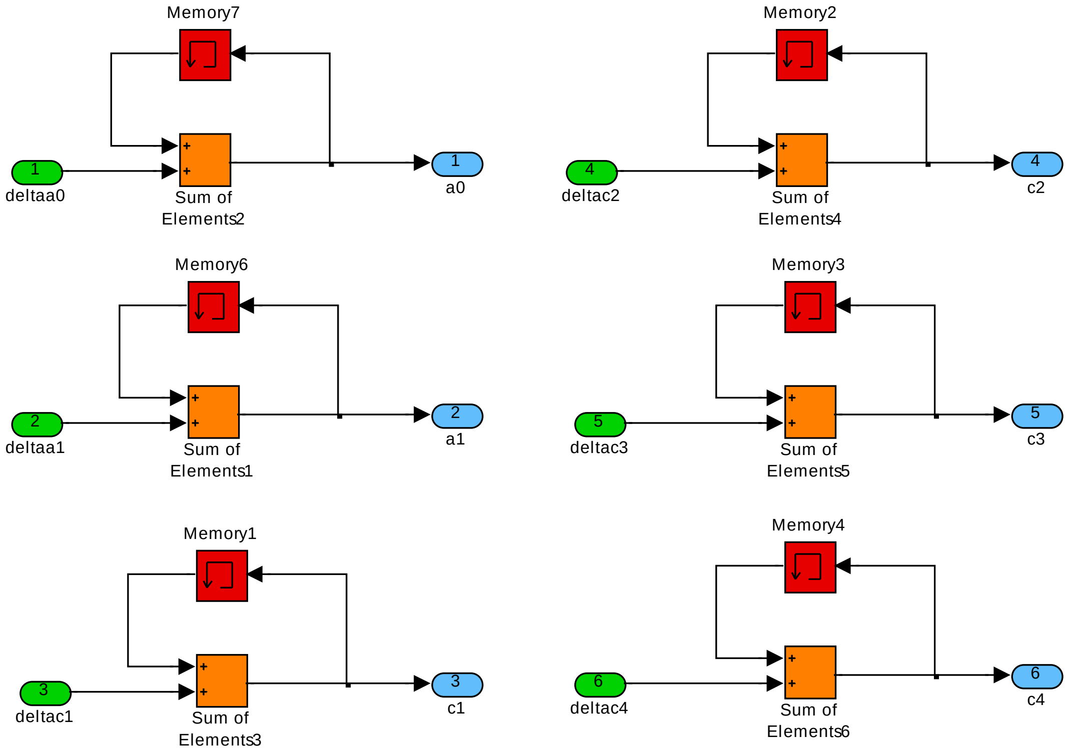


Figure A13 Direct Connections and Wavelet Neuron Weights Block

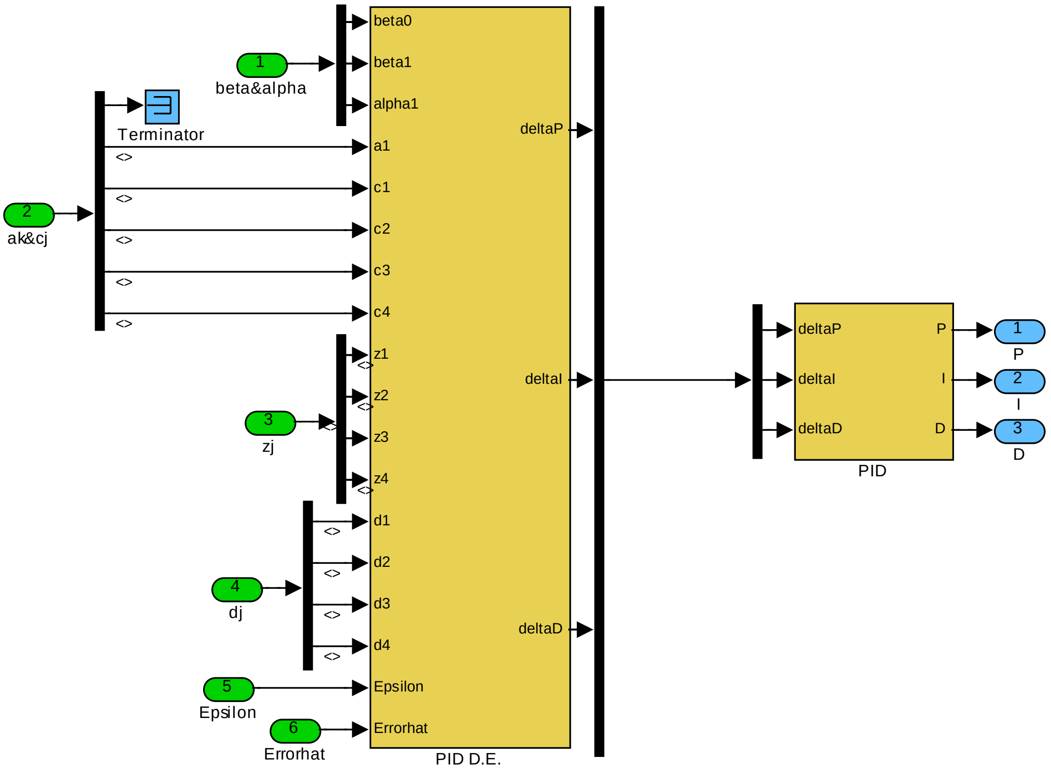


**Figure A14** PID Tuning Block

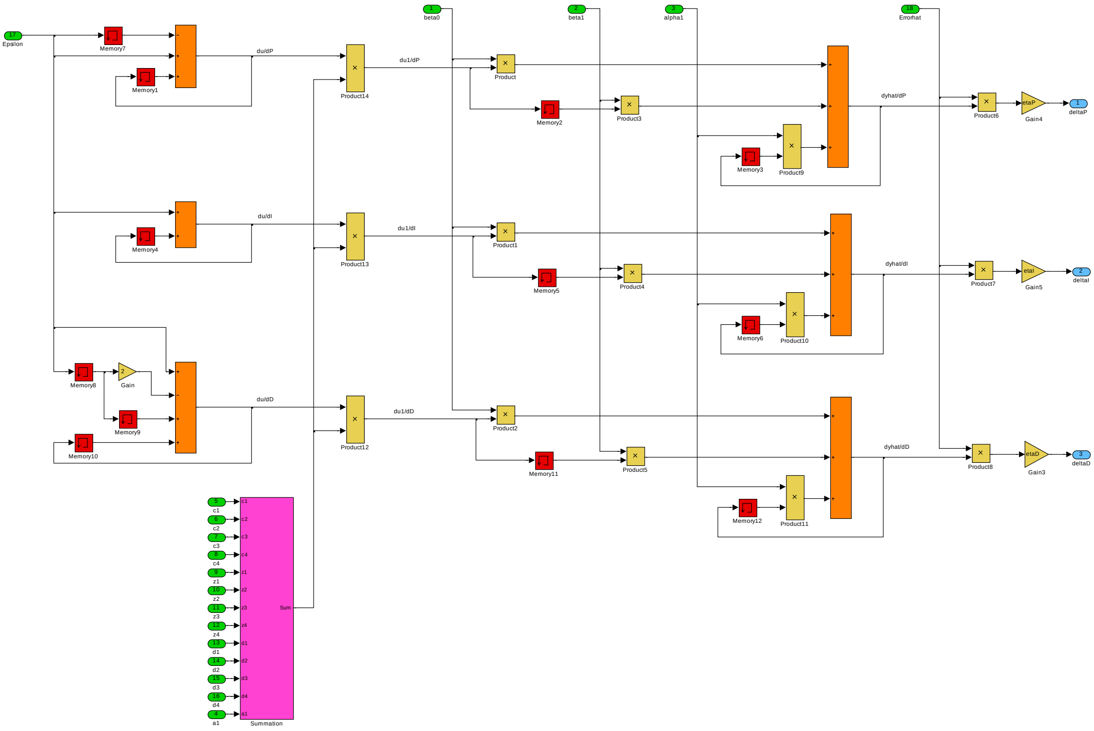


**Figure A15** PID Differential Equation Block

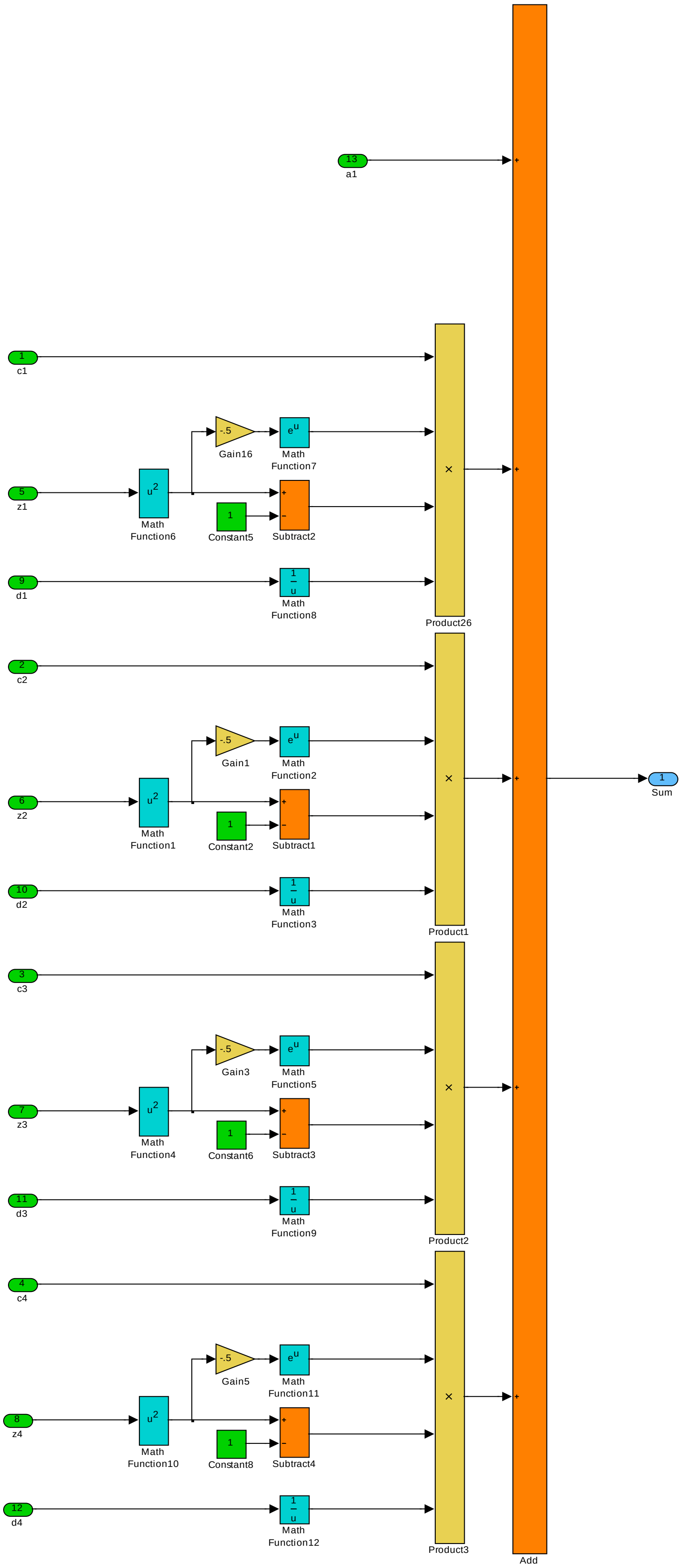


**Figure A16** Summation Block

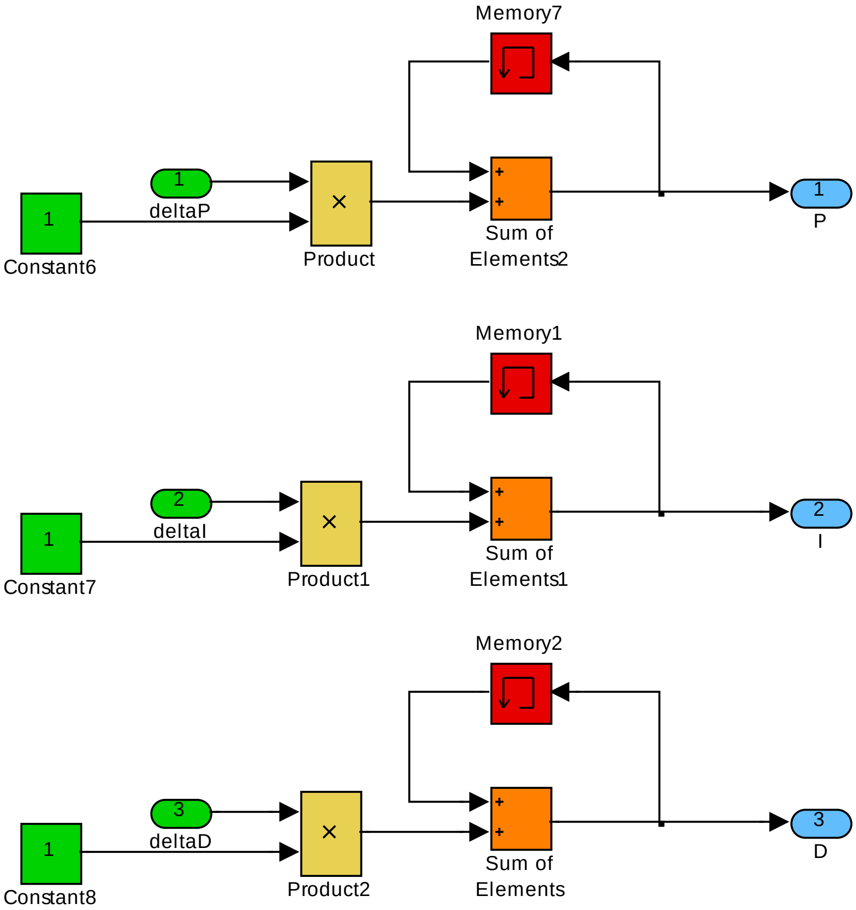


**Figure A17** PID Block

# VITA

Wissam Hassouneh was born on May 2, 1976, in Tripoli, Lebanon. He was raised in the United Arab Emirates and educated in private schools. He graduated from Al Manhal Canadian High School in Abu Dhabi, in 1994. He then immigrated to Canada with his family and studied at Carleton University in Ottawa, Ontario, from which he graduated in 2002. His degree was a Bachelor of Engineering in Electrical Engineering.

Mr. Hassouneh moved back to the United Arab Emirates in 2002 and worked as an Electromechanical Supervision Engineer for six months at Khatib & Alami in Abu Dhabi. After that, he worked as an Evaluation Engineer for two and a half years at Lin Scan in Ajman. In 2003, Mr. Hassouneh began a master’s program in Mechatronics Engineering at the American University of Sharjah. He was awarded the Master of Science degree in Mechatronics Engineering in 2006.