Adaptive Self-Tuning Neuro Wavelet Network Controllers

by

Gaviphat Lekutai

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

The Electrical Engineering Department

APPROVED

Dr. Hugh F. VanLandingham, Chairman
Dr. Richard L. Moose
Dr. Jeffrey H. Reed
Dr. Ioannis M. Besieris
Dr. Kenneth Hannsgen

March 31, 1997
Blacksburg, Virginia

Adaptive Self-Tuning Neuro Wavelet Network Controllers

Gaviphat Lekutai

The Bradley Department of Electrical Engineering

Virginia Polytechnic Institute and State University

Blacksburg, Virginia 24061-0111

Dr. Hugh F. VanLandingham, Chairman

(*ABSTRACT*)

Single layer feedforward neural networks with hidden nodes of adaptive wavelet functions (wavenets) have been successfully demonstrated to have potential in many applications. Yet applications in the process control area have not been investigated. In this paper an application to a self-tuning design method for an unknown nonlinear system is presented. Different types of frame wavelet functions are integrated for their simplicity, availability, and capability of constructing adaptive controllers. Infinite impulse response (IIR) recurrent structures are combined in cascade to the network to provide a double local structure resulting in improved speed of learning. In particular, neuro-based controllers assume a certain model structure to approximate the system dynamics of the "unknown" plant and generate the control signal. The capability of neuro-controllers to self-tuning of an unknown nonlinear plants is then illustrated through design examples. Simulation results demonstrate that the self-tuning design methods are directly applicable for a large class of nonlinear control systems.

Acknowledgments

I would like to express my sincere gratitude to my advisor, Dr. Hugh F. VanLandingham for his support, advice, and guidance throughout my Ph.D. research at Virginia Tech. I thank to Dr. VanLandingham for introducing me to the area of Artificial Neural Networks and for fostering an atmosphere of openness, creativity, and support. I am also very grateful to Dr. VanLandingham for many reviews and proofread of the materials.

I would like to thank Dr. Richard L. Moose and Dr. Kenneth Hannsgen for their valuable time to serve on my Ph.D. advisory committee. Their attentive responses and suggestions toward my dissertation are sincerely appreciated.

Grateful acknowledgment is extended to Dr. Jeffrey H. Reed and Dr. Ioannis M. Besieris whose reviews and constructive criticisms have improved the quality of this work. My sincere thanks are due to them for taking time from their busy schedules to serve on my advisory committee.

Finally, this research can never be completed without the continuous support, love, and encouragement of my parents. This dissertation is entirely dedicated to them. My very special thanks to my beloved fiancee, Puntipa Yanothai, for her love, patience, support, and unfailing enthusiasm in keeping continuity of this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A self-tuning method is an important system design consideration for constructing adaptive controllers of an *unknown slowly-varying system*. The basic idea in adaptive control is to estimate the uncertain plant parameters and correspondingly adjust control parameters on-line, based on the measured system signals, using the estimated parameters in the control input computation. However, traditional self-tuning adaptive control techniques can only deal with linear or special nonlinear systems. Typically, these techniques assume that the control model is operating in a linear region. The parameters of a *linearized* plant model are estimated recursively and used to update the controller. Often, it is not possible to represent, adequately, system characteristics such as nonlinearity, time delay, saturation, time-varying parameters, and overall complexity. It is important to develop an effective technique in which the structure of the nonlinear plant model can be identified by an adaptive process.

1.1 Literature Review

With emerging development in *neural networks*, *wavelets*, and *fuzzy logic* technologies, adaptive control designs can expand to even greater horizons. Some developments in *neuro-controller* concept have already proved to be useful for a wide class of practical situations; showing that they can cope with significant unknown nonlinearities [NP90] [LN95]. The authors in [KSS91] and [SS91] demonstrated a novel neuro controller using a three-layer dynamical recurrent neural network; however, they assumed that a system state space model of the plant was available. The authors in [Che90], [PSY88] and [LS89] employed multilayer feedforward neural networks in designing an unknown nonlinear self-tuning adaptive control with demonstrated potential; however, the *backpropagation* (BP) training technique is computationally complex and usually requires off-line computation to minimize the error.

Generally, in many applications multi/single layer feedforward neural networks have demonstrated an amazing ability to *learn* the desired map from discrete data. A number of rigorous mathematical proofs have been provided to explain this uncanny ability of feedforward neural networks to approximate maps [HSW89]. Recently, based on adaptive signal processing, combining wavelet transform theory with feedforward neural network architecture [PK93], [ZB92], a new mapping network called *neural network adaptive wavelets* has been proposed as an alternative to linear feedforward neural networks for approximating arbitrary nonlinear functions f in R^n. The term "*wavenet*" will be used for short, but is not to be confused with other authors' definitions [Wil94]. The concept of *wavenet* introduces a *super-wavelet* which is a linear combination of *daughter wavelets* that themselves are treated as wavelets. The daughter wavelet is simply a dilated and shifted version of the original wavelet or *mother wavelet*. The super-wavelet allows the shape of the wavelet to adapt to a particular problem, a concept which goes beyond adapting the parameters of a fixed shape wavelet. In terms of engineering applications, *wavenets* have also shown promising results in both signal representation and classification [STK92]. Studies are ongoing in some applications to demonstrate further potential, for example, low bit rate speech coding (about 2.4 k bits per sec.) [KS94a], [KS95], text independent speaker identification [KS94b], [KS94c], radar imaging [JK94], nonlinear channel equalization [CY94], chaotic signal detection [BGLSR96], and acoustic backscatter [TSD94], [TSD95]. The problem of self-tuning control with *wavenets* has not yet been addressed, thus providing a good research opportunity to investigate their merits.

The b*ackpropagation* (*BP*) algorithm, including its variants such as double backpropagation [DC91], dynamic backpropagation [KL92], novel backpropagation [Hag90], robust backpropagation [CJ94], etc., is the most popular network training method, but it is well known for its slow convergence and vulnerability to local minima. Unlike the multilayer perceptron which is a *global network*, *wavenet* is a *local network* in which the output function is well localized in both time and frequency domains. In a global network all the network weights are active for any given point in the input space and they may be updated with each training sequence. This global nature of the weight effects, however, tends to blur the details of local structure, slows the rate of learning and results in the

existence of local minima. However, in a local network only a small subset of weights are active at each point in the output space. The training of the network in one part of the input space does not corrupt that which has already been learned in more distant regions. Thus the learning speed of the local network is generally much faster than the global network. Furthermore, local minima can be eliminated in the local network. The *radial basis function* (*RBF*) network is an example of a *spatially local* network. In addition, *local recurrent* networks or *temporal local* networks such as center recurrent, linear recurrent, and IIR recurrent structures can be used in cascade with RBF networks to provide a double local network architecture resulting in even quicker learning and faster convergence [YL93].

Recently a class of *frame wavelets* has been introduced for its simplicity and availability that suits an adaptive process particularly in the neural network architecture [IRR95], [MFV96]. The ideas herein are derived from different viewpoints of the complex residue theorem, sigmoids, and hermiticity, namely rational functions with second-order poles (RASP) wavelets, superposed logistic functions (SLOG) wavelets, and polynomials windowed with Gaussians (POLYWOG) wavelets, respectively.

1.2 Objectives of Research

The goal of this research is to create design methods that address the self-tuning control problems for severely nonlinear systems by utilizing neural networks to achieve a nonlinear controller design. The methodology shows promise for control problems that are so complex that analytical design techniques do not exist. The research shows how *wavenets* can combine with self-tuning control algorithms to control tracking of a known/unknown discrete-time complex plant model. *Wavenet* learning architectures of neuro-controllers are introduced and various types of controllers are applied to constructing adaptive controllers. Two controllers based on *wavenet* structures will be described. One control scheme is based on an assumed plant model, approximating the unknown uncertain dynamics of the plant system, in which the parameters of the neural network are adjusted off-line (batch mode learning). Then the parameters of the neuro-identified predictive controls are adjusted on-line within a feedback loop based on the approximated algebraic computations at the

sampling instant. The other control scheme is an adaptive proportional-integral-derivative (PID) controller based on the self-tuning *wavenet* parameter adaptations. A well-known traditional backpropagation training will be included in the design for a comparison. Design examples will be given in different situations based on various noise environments assimilating practical purposes. Original contributions of the research will be given mainly to the *wavenet* design alogortihms and its computationally effective architecture.

1.3 Research Organization

This research work is divided into eight chapters. The introduction, including a literature review, research objectives, and research organization are presented in Chapter 1. Overviews of the least mean square (LMS) algorithms used as the basis of the *wavenet* algorithms are reviewed in Chapter 2. Including in the chapter are various forms of LMS adaptive filters such as normalized LMS, leaky LMS, averaged LMS, median LMS and sign-based LMS. Chapter 3 describes the fundamentals of neural networks and their current applications. Linear perceptron, multilayer perceptrons, a table of some common activation functions, backpropagation algorithms, and radial basis functions are among the contents in this chapter. Chapter 4 provides the fundamentals of wavelets theory and their transform. Morlet wavelets, Shannon wavelets, and a new class of frame wavelets such as Rational functions with Second-order Poles (RASP) wavelets, Superposed LOGistic functions (SLOG) wavelets, and POLYnomials WindOwed with Gaussians (POLYWOG) wavelets are defined and illustrated as a mother wavelet basis used in the next chapter. Chapters 5 and 6 reveal the core content of the research by unifying all information from the previous chapters. *Wavenet* structures and algorithms are provided in Chapter 5 with computer simulations and results of the investigation. Chapter 6 demonstrates control structures, definitions, and examples of engineering applications to set-point control of nonlinear systems. Finally, suggestions for future work and conclusions are given in Chapter 7 and Chapter 8, respectively.

Chapter 2

Overviews of LMS Algorithms

In *adaptive signal processing* a process structure is adjusted or adapted to track changes in the optimal solution to provide improved behavior or action according to some desired criterion. A typical approach for adaptive filtering algorithms is based on the classical *method of* (*fixed*) *least squared error reduction* which involves the use of time averages and a fixed, finite length filter. Figure 2.1 represents this system configuration and its essential features. The system consists of an adaptive *finite impulse response* (FIR) filter $\underline{f}_n$ acting on an input sequence $x(n)$ to produce an output signal $y(n)$. The filter is designed so that the output should approximate a training signal or desired response $d(n)$. The estimation error $e(n)$, which is used for controlling the filter coefficients, is the difference between the desired output signal and the actual output signal of the adaptive system.

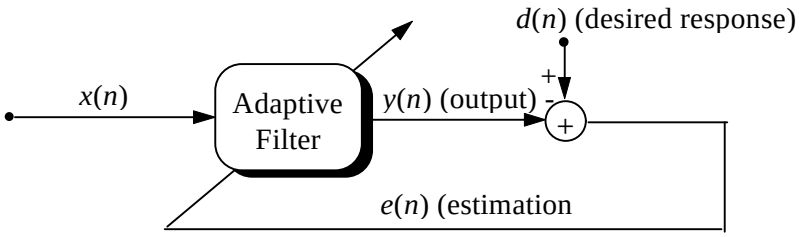


Figure 2.1 Basic Adaptive Filtering System

The basic adaptive filter structure of Figure 2.1 serves as the core of many adaptive filtering applications as shown in Figure 2.2. The prediction application in Figure 2.2(a) is used to estimate the present value of a random input signal from its past values. Obviously, the desired signal is the current input signal and a delayed version of the current input signal is supplied to the adaptive filter. Prediction is used in signal encoding and noise reduction [WS85].

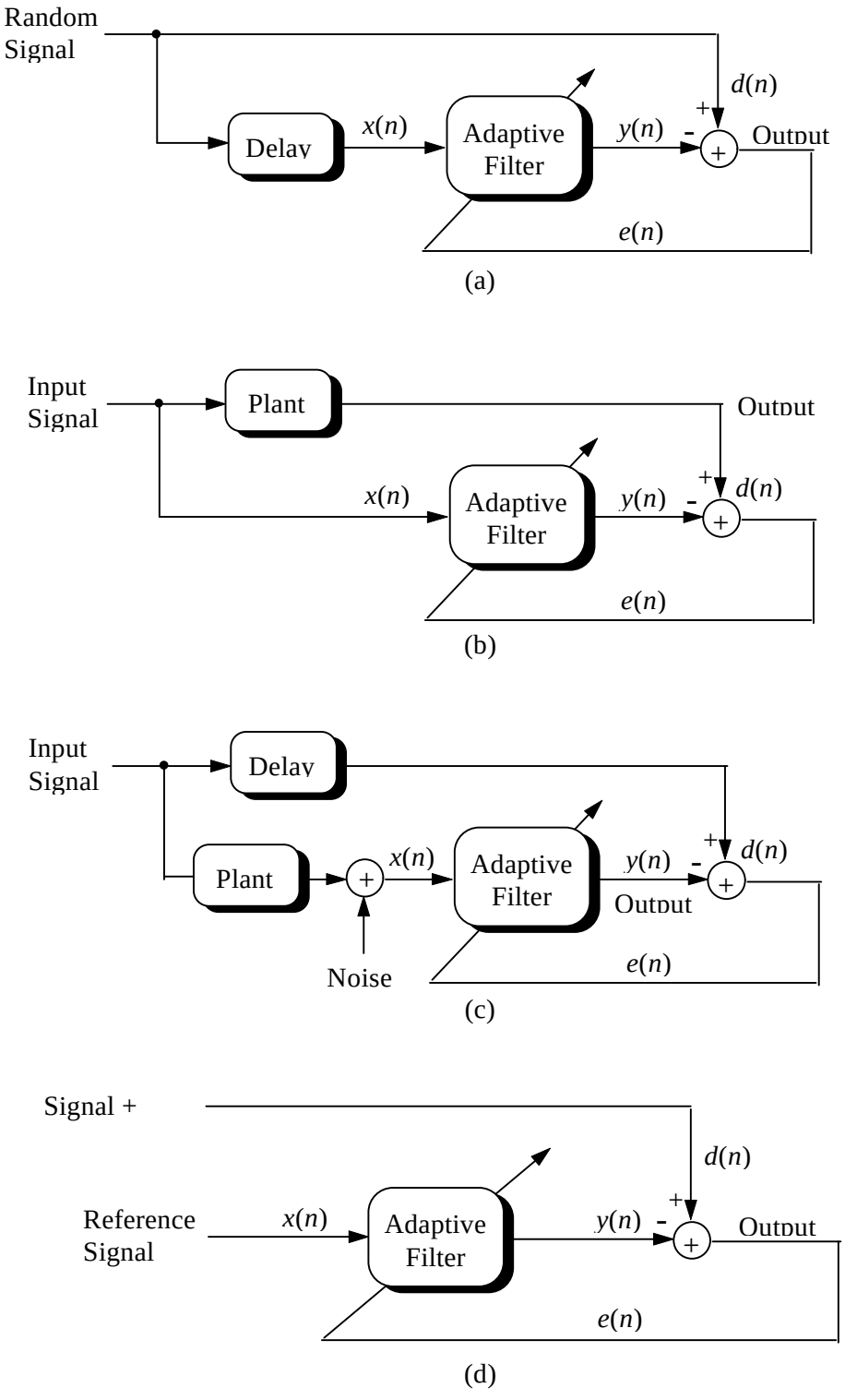


Figure 2.2 Examples of Adaptive Filtering Applications: (a) Prediction; (b) Identification; (c) Equalization (Inverse Modeling); (d) Interference Canceling

The system identification application in Figure 2.2(b) is used to provide a linear model that represents the best fit to an *unknown plant*. The plant and the adaptive filter are driven by the same broadband input signal. The plant output supplies the desired response for the adaptive filter that approximates the plant's transfer characteristic. After adaptation the plant is *identified* in the sense that its transfer function is approximated by the adaptive processor. Identification or modeling can be used to model a slowly time varying plant whose input and output signals are available. Example applications include multipath channel modeling and automatic gain control (AGC) used in radio and television receivers [MA90].

The inverse modeling application in Figure 2.2(c) is used to provide an *inverse model* that estimate to an unknown noisy plant. The adaptive filter attempts to recover a delayed and noisy version of the input signal. Ideally, the inverse model has a transfer function equal to the *reciprocal* of the plant's transfer function. Adaptive equalization is an example application which is used to deconvolve the effects of a transducer, and eliminate the intersymbol interference from future communication data symbols [HM84].

Finally, Figure 2.2(d) illustrates the adaptive processor applied to interference cancellation. The goal of the adaptive filter is to cancel unknown interference contained in the input source by a corrupted noise, and produce an output that closely resembles the true signal. A reference signal is employed as the input to the adaptive filter. Adaptive interference canceling is used in echo cancellation, radar polarimetry, and adaptive beam forming [Hay91].

2.1 Least Mean Squares Adaptive Filter

Figure 2.1 represents the structure of a *least mean squares* (LMS) adaptive filter. Let the set of filter impulse response coefficients be $f_n(j)$, where j = 0,1,2,...,L-1 at sample index n. In terms of the vector notation, the filter coefficients $\underline{f}_n$ can be written as

$$\underline{f}_n = [f_n(0), f_n(1), \ldots, f_n(L-1)]^T \tag{2.1}$$

Similarly, the input sequence $x(n)$ can be written in terms of the data vector $\underline{x}_n$ as

$$\underline{x}_n = [x(n), x(n-1), \ldots, x(n-L+1)]^T \tag{2.2}$$

The output of the filter $y(n)$ is expressed as a linear convolution sum

$$y(n) = \sum_{i=0}^{L-1} f_n(i)x(n-i) \tag{2.3a}$$

or, in vector form $$y(n) = \underline{f}_n^T \underline{x}_n \tag{2.3b}$$

The estimation error is defined by

$$e(n) = d(n) - y(n) \tag{2.4}$$

For random input sequences, the objective functional to be minimized is

$$J = \tfrac{1}{2} E\{e^2(n)\} \tag{2.5}$$

where E is the expectation operator. For deterministic sequences J is

$$J = \tfrac{1}{2} \sum_n e^2(n) \tag{2.6}$$

where the range of n is generally determined by the length of the input data sequence L.

Minimization is performed by differentiating J with respect to each coefficient $f(j)$ and equating the results to zero. For example, the differentiation of the random input sequences in equation (2.5) yields

$$\frac{\partial J}{\partial f(j)} = E\left\{e(n)\frac{\partial e(n)}{\partial f(j)}\right\} \tag{2.7}$$

Using equations (2.4) and (2.3a), we have

$$\frac{\partial e(n)}{\partial f(j)} = -x(n-j) \tag{2.8}$$

Thus $$\frac{\partial J}{\partial f(j)} = \sum_i r_{xx}(j-i) f(i) - r_{xd}(j) \tag{2.9}$$

where $r_{xx}(j-i)$ is the autocorrelation of the input signal $x(n)$ defined as

$$r_{xx}(j-i) = E\{x(n-i)x(n-j)\} \tag{2.10}$$

$r_{xd}(j)$ is the crosscorrelation between the input $x(n)$ and the desired output $d(n)$, defined as

$$r_{xd}(j) = E\{x(n-j)d(n)\} \tag{2.11}$$

It is assumed that $x(n)$ and $d(n)$ are wide-sense stationary. Setting equation (2.9) equal to zero gives

$$\sum_i r_{xx}(j-i) f(i) = r_{xd}(j), \tag{2.12a}$$

or, in vector notation,

$$R\underline{f} = \underline{g} \tag{2.12b}$$

where R is the matrix of autocorrelation coefficients $r = r_{xx}$:

$$R = \begin{bmatrix} r(0) & r(1) & .... & r(L-1) \\ r(1) & r(0) & .... & : \\ : & : & .... & r(1) \\ r(L-1) & r(L-2) & .... & r(0) \end{bmatrix} \tag{2.13}$$

and $\underline{g}$ is given by

$$\underline{g} = \left[r_{xd}(0), r_{xd}(1), ..., r_{xd}(L-1)\right]^T \tag{2.14}$$

Note that the autocorrelation is an even sequence $r(i) = r(-i)$, thus the elements on each diagonal of R are identical. Such an (L x L) matrix is called *Toeplitz*. Also, equation (2.12) forms a set of linear equations known as the *normal equations*. The solution of the normal equations yields the optimal filter known as the *minimum mean squared error* solution or simply as the *least squares filter*. The optimal solution thus is given by

$$\underline{f}_{opt} = R^{-1}\underline{g} \tag{2.15}$$

However, (2.15) cannot be easily used, especially in real time applications, because the matrix R and the vector $\underline{g}$ must first be estimated and then R must be inverted. A much simpler iterative formula for the solution can be found using the approximate algorithm:

$$\underline{f}_{n+1} = \underline{f}_n - \mu_n \nabla J_n \tag{2.16}$$

where $\underline{f}_n$ denotes the kth update of the filter coefficient vector, μ_n is a *step-size* parameter, and ∇J_n defines the **search direction** of the algorithm. Obviously, ∇J_n is the *gradient of the mean squared error* of the performance index J with respect to the filter coefficients $\underline{f}_n$.

$$\nabla J_n = \frac{\partial J}{\partial \underline{f}_n} = \left[\frac{\partial J}{\partial f_n(0)}, \frac{\partial J}{\partial f_n(1)}, ..., \frac{\partial J}{\partial f_n(L-1)}\right]^T \tag{2.17}$$

Equation (2.17) is the same as equation (2.9) written in vector form (for stationary process):

$$\nabla J_n = R\underline{f}_n - \underline{g} \tag{2.18}$$

Thus using equations (2.16) and (2.18) the filter coefficients are adaptable to general changes in the input signal environment, the iterative least squares filter solution now is

$$\underline{f}_{n+1} = \underline{f}_n - \mu_n (R_n \underline{f}_n - \underline{g}_n) \tag{2.19}$$

The main problem with this approach is the computational complexity for R_n and $\underline{g}_n$. To overcome this problem, the gradient ∇J_n is replaced by the gradient $\hat{\nabla J}$ *estimate* based on the *instantaneous* value of the squared error. That is

$$\hat{\nabla J} = \frac{\partial \hat{J}}{\partial \underline{f}_n} = \frac{\frac{1}{2}\partial e^2(n)}{\partial \underline{f}_n} \tag{2.20}$$

The resulting iterative procedure is known as the **Least Mean Squares** (**LMS**) algorithm [Cla93]. The filter coefficient iteration (update formula) for the LMS algorithm is derived by using the instantaneous estimate in the steepest descent update equation, so that equation (2.16) becomes

$$\underline{f}_{n+1} = \underline{f}_n - \mu \hat{\nabla J} \tag{2.21}$$

Here let the step-size adaptation μ be constant. Now the estimate performance is given by

$$\hat{J} = \frac{1}{2} e^2(n) = \frac{1}{2}\left(d(n) - \underline{f}_n^T \underline{x}_n \right)^2 \tag{2.22}$$

As a consequence,

$$\hat{\nabla J} = e(n) \frac{\partial e(n)}{\partial \underline{f}_n} \tag{2.23}$$

and

$$\frac{\partial e(n)}{\partial \underline{f}_n} = -\underline{x}_n \tag{2.24}$$

Hence equation (2.21) becomes

$$\underline{f}_{n+1} = \underline{f}_n + \mu\, e(n) \underline{x}_n \tag{2.25}$$

The LMS adaptive filter equations is summarized in Table 2.1.

Table 2.1 The LMS Adaptive Filter

Let $\underline{f}_0 = \underline{0}$

$$\underline{f}_{n+1} = \underline{f}_n + \mu e(n)\underline{x}_n \qquad (2.25)$$

$$e(n) = d(n) - y(n) \qquad (2.4)$$

$$y(n) = \underline{f}_n^T \underline{x}_n \qquad (2.3b)$$

where

$$\underline{f}_n = [f_n(0), f_n(1), \ldots, f_n(L-1)]^T \qquad (2.1)$$

$$\underline{x}_n = [x(n), x(n-1), \ldots, x(n-L+1)]^T \qquad (2.2)$$

2.2 Normalized LMS

The *normalized LMS* (*NLMS*) algorithm is used in many applications where the input signals are subject to widely fluctuating power levels causing a *gradient noise amplification* effect, which in turn influences the stability, convergence and steady-state properties of the LMS algorithm. The NLMS update can be derived from the solution to a constrained optimization problem [WS85]. The term "normalized LMS algorithm" was introduced by Bitmead and Anderson in 1980 and is the modification of the LMS form [BA80].

$$\underline{f}_{n+1} = \underline{f}_n + \mu \frac{e(n)\underline{x}_n}{\underline{x}_n^T \underline{x}_n} \qquad (2.26)$$

In order to avoid the probability of zero division in (2.26), the update equations can be modified

$$\underline{f}_{n+1} = \underline{f}_n + \mu \frac{e(n)\underline{x}_n}{c + \underline{x}_n^T \underline{x}_n} \qquad (2.27)$$

where c is a small positive constant.

The disadvantage of equation (2.27) is the increasing response time of the algorithm when power changes occur. Ref. [Ber86] shows that for zero-mean Gaussian inputs, the basic normalized algorithm converges in the mean to the least squares solution (2.15). However, the normalized LMS also requires a large computational storage.

2.3 Leaky LMS

The *Leaky* or *leakage LMS* (*LLMS*) algorithm is more robustly than the LMS. Basically, leakage prevents the occurrence of digital filter instability, sensitivity, and overflow in a limited point arithmetic by providing a compromise between minimizing the mean squared error and containing the energy in the impulse response of the adaptive filter. The LLMS algorithm employs the LMS update modified by the presence of a constant leakage factor γ

$$\underline{f}_{n+1} = \gamma \underline{f}_n + \mu e(n) \underline{x}_n \tag{2.28}$$

where $0 < \gamma < 1$, but typically γ is close to 1.

This adaptive algorithm searches for the minimization of the modified cost function

$$J_{leaky} = \tfrac{1}{2}\left[e^2(n) + a \, \underline{f}_n^T \underline{f}_n \right] \tag{2.29}$$

where $\alpha = (1-\gamma)/\mu$.

With the LMS algorithm in (2.25), if the gradient estimates or μ were suddenly to become zero, the filter coefficients would remain constant indefinitely, but with the LLMS algorithm in (2.28) the filter vector would gradually decay towards zero. This is important when the filter cannot converge or becomes marginally stable. For example, in a sidelobe canceler, the LLMS algorithm can be used to establish an equivalent input noise power level which compares to the power of the incoming signal and the ratio determines the extent of acceptance or cancellation [WS85]. In adaptive differential pulse code modulation (ADPCM), the transmission is often corrupted by bit errors. Typically bit error rates (BER) may range from lows of 10^{-6} to up to 10^{-2}. In DPCM coding, such errors create problems for both the quantizer and predictive segments of the system. The LLMS allow the impact of these channels errors to gradually decay [Gib80].

Rearranging the update (2.28), the LLMS algorithm can be written

$$\underline{f}_{n+1} = (\gamma I - \mu \underline{x}_n \underline{x}_n^T) \underline{f}_n + \mu d(n) \underline{x}_n \tag{2.30}$$

Taking the expectation operator of both sides

$$E\left[\underline{f}_{n+1}\right] = (gI - mR)\,E\left[\underline{f}_{n}\right] + m\,\underline{g} \tag{2.31}$$

Assumed wide sense stationary, the converged solution for the leaky LMS algorithm is

$$\lim_{n\to\infty} E\left\{\underline{f}_{n}\right\} = \left[R + \frac{(1-\gamma)}{\mu} I\right]^{-1} \underline{g} \tag{2.32}$$

which is clearly biased from the optimal solution $R^{-1}\underline{g}$.

2.4 Averaged LMS

The *averaged LMS* (*ALMS*) algorithm is often used in the case of a noisy instantaneous gradient estimate which causes the LMS algorithm to diverge. The ALMS has been shown in [Gar84] to reduce the noise in the gradient estimate by linearly smoothing the LMS gradient estimates, i.e., averaging over several consecutive instantaneous values. The smoothing can be constructed using a moving window of the N most recent gradient estimates. The ALMS algorithm has the form

$$\underline{f}_{n+1} = \underline{f}_{n} + \frac{\mu}{N} \sum_{j=n-N+1}^{n} e(j)\underline{x}_{j} \tag{2.33}$$

An alternative approach is to filter the noisy gradient vectors by a general lowpass filter and use the output of the filter as an estimate of the gradient vector [Pro95]. Then the update equation for this modified algorithm is

$$\underline{f}_{n+1} = \underline{f}_{n} + \mu \underline{h}_{n} \tag{2.34}$$

where $\underline{h}_{n} = [h_{n}(0), h_{n}(1), ..., h_{n}(L-1)]^{T}$, and $h_{n}(i)$ is the output of a lowpass filtering operation applied to the ith component of the gradient estimate, i.e.,

$$h_{n}(i) = LPF\{\, e(n)x(n-i), e(n-1)x(n-i-1), ...\} \tag{2.35}$$

The lowpass filter in equation (2.33) has a uniform impulse response, {1/N, 1/N, ... , 1/N}. However, there is no restriction to what type of filters that can be used; for example, consider the first-order recursive equation

$$\underline{h}_{n} = \underline{h}_{n-1} + m\left(e(n)\underline{x}_{n} - \underline{h}_{n-1}\right) \tag{2.36}$$

where the choice of $0 < m < 1$ determines the bandwidth of the lowpass filter. When m is

close to unity, the pole of the filter approaches zero, thus the filter bandwidth is small and the effective averaging is performed over many gradient vectors. On the other hand, when *m* is small, the pole approaches unity, hence, the lowpass filter has a large bandwidth and it provides little averaging of the gradient estimates. Combining Equations (2.36) with (2.34), an overall modified update equation is

$$\underline{f}_{n+1} = \underline{f}_n + (1-m)\left(\underline{f}_n - \underline{f}_{n-1}\right) + m\mu e(n)\underline{x}_n \tag{2.37}$$

This algorithm is also known as the *momentum LMS*. Note that the authors in [RS90] have concluded that using momentum algorithm reduces the convergence rate.

2.5 Median LMS

In this section, the linear smoothing of the LMS algorithm of the previous section is extended to a *nonlinear* smoothing using the sample *median* of the *N* most recent instantaneous gradient estimates. It has been shown that the nonlinear smoothing using median filter outperforms the moving average filter in the case of additive long-tailed noise [PV90]. This is very useful for eliminating interference in either $d(n)$ or $x(n)$. The median of *N* samples is obtained by algebraically ranking the samples from smallest to largest and then selecting the middle value. Thus, given a sequence of *N* signal samples $z_1, z_2, ..., z_N$, and ranking the samples as $z_{(1)}, z_{(2)}, ..., z_{(N)}$, with $z_{(1)} \le z_{(2)} ... \le z_{(N)}$, the sample median is defined as

$$z_{med} = \begin{cases} z_{(\upsilon)} & \text{if } N \text{ is odd} \\ \frac{1}{2}\left(z_{(\upsilon-1)} + z_{(\upsilon)}\right) & \text{if } N \text{ is even} \end{cases} \tag{2.38}$$

where $\upsilon = [N/2]+1$ and [.] is the integer part of decimal number, for example, [2.5] = 2. Consequently, a *median LMS* (MLMS) algorithm is defined by an update of the form

$$f_{n+1}(i) = f_n(i) + \mu\{e(n)x(n-i)\}_{med} \tag{2.39}$$

where *i* is again the length of the filter coefficients = 0,1,...,*L*-1, and $\{e(n)x(n\text{-}i)\}_{med}$ denotes the median operation of equation (2.38) to the *N* most recent gradient estimates, $\{e(n)x(n\text{-}i), e(n\text{-}1)x(n\text{-}i\text{-}1), \ldots, e(n\text{-}N\text{+}1)x(n\text{-}i\text{-}N\text{+}1)\}$.

The MLMS has slower convergence when the inputs are non-impulsive. Analytic performance predictions have only proved possible for very restrictive inputs [Cla93].

2.6 Sign Based LMS

In this section, the motivation of *sign operation based LMS algorithms* is to provide simplified algorithms with a reduced computational requirement; particularly, in high speed communications where the LMS computational load is too high. The algorithm can be derived directly from the error function based on the *least absolute values* criterion:

$$J = \sum_{n} |e(n)| \tag{2.40}$$

The algorithm reduces computation and simplifies hardware requirements by the fact that the gradient estimates of the error function (2.40) become the *sign function* given by

$$\text{sgn}(e) = \frac{e}{|e|} = \begin{cases} 1 & e > 0 \\ 0 & e = 0 \\ -1 & e < 0 \end{cases} \tag{2.41}$$

Thus, the adaptive update represents only the sign changes of the error signal quantity.

$$\underline{f}_{n+1} = \underline{f}_{n} + \mu\, \text{sgn}(e(n))\underline{x}_{n} \tag{2.42}$$

This algorithm is sometimes called the *pilot LMS*, or *signed error* (*SE*), or simply *sign algorithm* (*SA*). Moreover, some possible variations of the sign based algorithm have been incorporated into many commercial adaptive equalizers that are used in high-speed modems [DPH93]. The variations are obtained by using only sign information in the error signal or in the quantize data or both. The variations of the algorithm are

The clipped LMS, or *signed regressor* (*SR*)

$$\underline{f}_{n+1} = \underline{f}_{n} + \mu\, e(n)\, \text{sgn}(\underline{x}_{n}) \tag{2.43}$$

and *the zero-forcing LMS*, or *sign-sign* (*SS*)

$$\underline{f}_{n+1} = \underline{f}_{n} + \mu\, \text{sgn}(e(n))\text{sgn}(\underline{x}_{n}) \tag{2.44}$$

The advantage of the use of sign representations is that multiplications in the update are reduced to single bit operations. However, this crude quantization of the gradient estimates reduces the convergence rate and increases steady-state error. Also, the SR algorithm of (2.43) can produce unstable results in some cases because of the changes in the direction of the update by replacing $\underline{x}_n$ with $\text{sgn}(\underline{x}_n)$ causing the filter coefficients to move away from the optimal solution [SMA88].

Chapter 3

Fundamentals of Neural Networks

A *neural network* is an interconnected network of simple processing elements, e.g. scaling and filtering. The processing elements interact along paths of variable connection strengths which when suitably adapted can collectively produce complex overall desired behavior.

The design of *artificial neural network* (*ANN*) has been inspired by the biological research on how the human's brain works. The brain is a network consisting of approximately 2.5 billion simple processors, called *neurons*, connected to one another through branchlike structured called *axons* and *dendrites*. *Synapses* connect the axons and dendrites of one neuron to those of another. The objective of ANN is to mimic the neurons in the brain by linking together many simple processors, called *artificial neurons* or *nodes*. Variable strength connections, called *weights*, implement the biological synapses [MR90]. Obviously neural networks are well suited to solving the same types of problems as the human's brain. Particularly, neural networks excel at recognition, identification, and classification types of problems [Wel94]. The following problems are a few examples to which neural networks have been applied.

- *Radar and sonar signal classification*. Neural networks can distinguish among different types of radar returns (weather, birds, aircraft) with greater accuracy than conventional systems [HD91]. A neural network has been trained to distinguish between sonar returns from a rock and returns from a steel cylinder [GS88].
- *Speech-to-text conversion*. The author in [SR87] designed a famous NETTALK program that has learned to read text and convert it to speech.
- *VLSI technology*. The relatively small and simple topology of neural networks greatly simplifies the design and layout of VLSI circuits [MAL91].
- *Biomedical applications.* Neural networks are finding many uses in medical classification and diagnosis applications e.g. image data to classify reactions of blood cells to antibodies [DC92]. For a more thorough overview of biomedical

applications of neural networks see [EB95] which provides a selected bibliography of over 100 references in several medical disciplines.

- *Process control*. Texas Eastman Company has been able to reduce the use of an expensive additive by one-third at its Longview chemical plant [Ham93].
- *Character recognition*. Security Pacific in Los Angeles uses NN to detect patterns of fraudulent credit card use by noticing changes in buying patterns [New95].
- *Virus protection*. IBM uses neural-based detection to eradicate new viruses automatically through an "artificial immune" system for computers [Joh95].
- *Investing and trading*. Morgan Stanley & Fidelity Investments companies use a neural network to search for patterns in stocks, foreign exchanges, etc. [Cor95]

3.1 Neuron Modeling

In an effort to model certain capabilities of the brain, Warren McCulloch and Walter Pitts established a simplified model of a biological neuron in 1943 called *the McCulloch-Pitts* model consisting of multiple inputs and one output with a central processing unit (CPU) [Lip87]. Figure 3.1 shows the *model* for a neuron which is described by:

$$y = f\left(\sum_{i=1}^{N} w_i x_i - v_t\right) \tag{3.1}$$

where x_i = input signals, $i = 1, 2, 3, ..., N$

w_i = synaptic weights

v_t = *threshold* or bias

$f(\cdot)$ = *activation function* or *squashing function* or processing element

y = output signal of the neuron.

The use of threshold v_t is to provide a bias to the activation function $f(\cdot)$. McCulloch and Pitts did not provide any method through which the node or neuron could self-adjust or adapt its synaptic weights in a learning manner. In 1949, Hebb suggested a simple mathematical formula that can adaptively change the neuron weights in proportion to the activity between the pre- and post- synaptics of the neuron:

$$\Delta w_i(n) = \mu\, y(n)\, x_i(n) \tag{3.2}$$

where μ is the positive, constant learning rate at all time n.

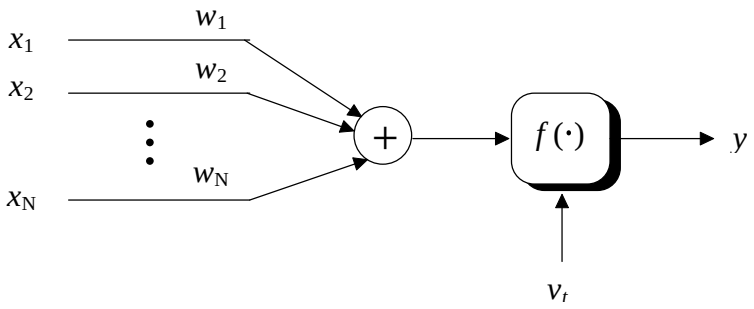


Figure 3.1 The McCulloch-Pitts Model of a Neuron

3.2 The Perceptron

In 1958, Rosenblatt demonstrated some practical applications using the *perceptron* [HH93]. The perceptron is a single level connection of McCulloch-Pitts neurons sometimes called *single-layer feedforward networks*. The network is capable of linearly separating the input vectors into pattern of classes by a hyperplane. A *linear associative memory* is an example of a single-layer neural network. In such an application, the network associates an output pattern (vector) with an input pattern (vector), and information is stored in the network by virtue of modifications made to the synaptic weights of the network. Figure 3.2 illustrates the perceptron which is described by:

$$y_i = f\left(\sum_{j=1}^{N} w_{ij}x_j - v_t\right) \tag{3.3}$$

where $i = 1, 2, ..., M$ (output nodes), $j = 1, 2, ..., N$ (inputs).

Rosenblatt derived a learning rule based on weights adjusted in proportion to the error between the output neurons and the desired outputs (target). The weight adaptations are given by:

$$\Delta w_{ij}(n) = \mu[d_i(n) - y_i(n)]x_j(n) \tag{3.4}$$

where $i = 1, 2, ..., M$ (outputs), $j = 1, 2, ..., N$ (inputs), and d_i is the desired output at node i of time n.

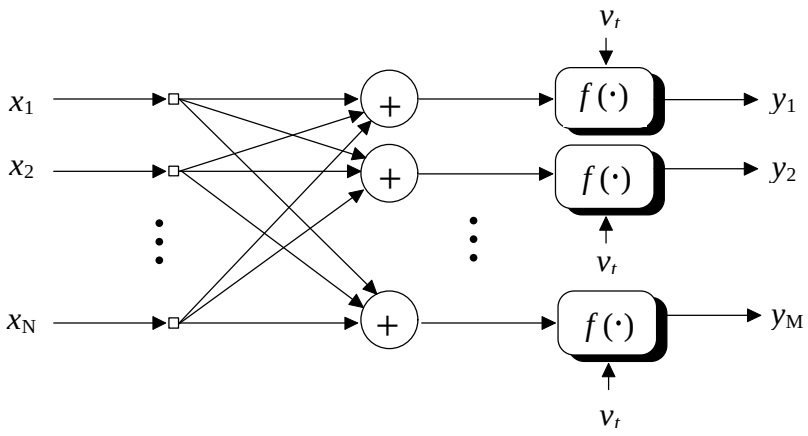


Figure 3.2 Single-Layer Perceptron (Single-Layer Feedforward Network)

3.3 Multilayer Perceptrons

In order to extract higher-order statistics such as the realization of the simple XOR or XNOR logic function (without a pre-processor unit which is often used in a single-layer perceptron), *nonlinear multilayer perceptrons* or *multilayer feedforward networks* were suggested by Minsky and Papert in 1969 who mathematically demonstrated that there were fundamental limits on what single layer perceptron could compute [Hay94]. The multilayer perceptrons (MLP) introduce one or more *hidden layers*, whose computation nodes are correspondingly called *hidden neurons*. The function of the hidden neurons is to intervene between the external input and the network output. Figure 3.3 provides a fully connected feed-forward three layer multilayer perceptrons with one hidden layer and output. The source nodes in the input layer of the network consist of N elements of the pattern which constitute the input signals applied to K neurons in the second layer or the first hidden layer ($l = 1$). The output signals of M neurons are the final layer ($l = L$) of the network constitute the overall response of the network to the pattern supplied by the source nodes. Also, the proper choice of the number of hidden nodes can be calculated initially for better generalization (see [CU93]).

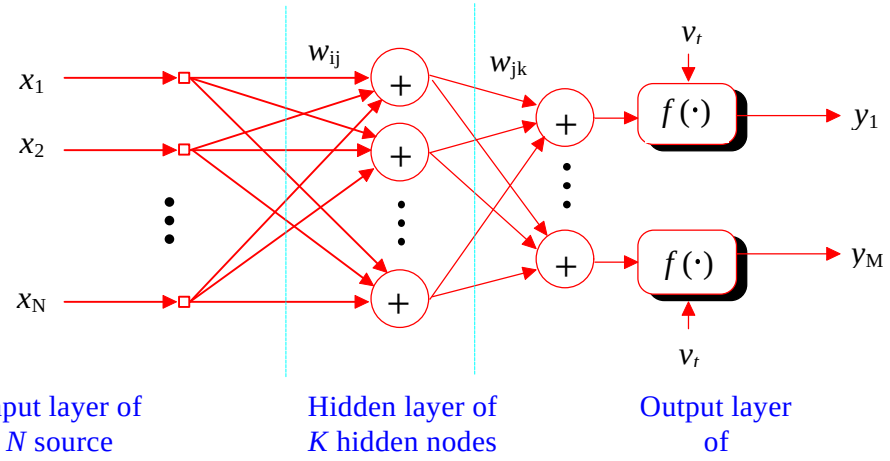


Figure 3.3 Three Layer Multilayer Perceptrons With One Hidden Layer and Output

All the three neural network architectures described so far employ the activation function $f(\cdot)$ which defines as the output of a neuron in terms of the activity level at its input (ranges from -1 to 1 or 0 to 1). Table 3.1 summarizes the basic types of activation functions.

The most practical activation functions are the sigmoid and the hyperbolic tangent functions. This is because they are differentiable.

Table 3.1 Common Activation Functions

Name	Definition
Linear	$f(x) = kx$
Step (commonly: $\beta = 1$, $\delta = 0$, $x_k = 0$)	$f(x) = \begin{cases} \beta & \text{if } x \geq x_k \\ \delta & \text{if } x < x_k \end{cases}$
Ramp	$f(x) = \begin{cases} \rho & \text{if } x \geq \rho \\ x & \text{if } \lvert x \rvert < \rho \\ -\rho & \text{if } x \leq -\rho \end{cases}$
Sigmoid	$f(x) = \dfrac{1}{1+e^{-\alpha x}}, \quad \alpha > 0$

Table 3.1 Common Activation Functions (Continued)

Name	Definition
Hyperbolic Tangent	$f(x) = \tanh(\gamma x) = \dfrac{1-e^{-2\gamma x}}{1+e^{-2\gamma x}}, \quad \gamma > 0$
Rational	$f(x) = \begin{cases} \dfrac{x^2}{1+x^2} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$
Gaussian	$f(x) = \dfrac{1}{\sqrt{2\pi}\,\sigma} \exp\left[-\dfrac{(x-\mu)^2}{2\sigma^2}\right]$

3.3.1 Backpropagation Algorithm

The most popular and succesful learning method for training the multilayer perceptrons is *the backpropagation algorithm*. The development of the backpropagation learning was reported by Rumelhart Hinton and Williams in 1986 [Hay94]. The algorithm employs an iterative gradient-descent method of minimization which minimizes the mean squared error (*L2* norm) between the desired output and network output (supervised learning). The backpropagation training procedure is presented below.

Let
$$E = \frac{1}{2}\sum_{j=1}^{N}\sum_{i=1}^{M} e_i^2(n) \tag{3.5}$$

where M = dimension of output space

N = number of training input patterns

n = number of iterations

$$e_i(n) = d_i(n) - y_i^{(L)}(n) \tag{3.6}$$

where L = final hidden layer or the output layer

$$v_i^l(n) = \sum_{j=1}^{N} w_{ij}^{(l)}(n)\, y_j^{(l-1)}(n) \tag{3.7}$$

where $y_j^{(l-1)}(n)$ is the function signal of neuron j in the previous layer l - 1 at iteration n

$w_{ij}^{(l)}(n)$ is the weight of neuron i in layer l that is fed from neuron j in layer l - 1

Then the output signal of neuron i in layer l is

$$y_i^{(l)}(n) = f(v_i^l(n)) \tag{3.8}$$

where $f(\cdot)$ is the activation function (given in Table 3.1).

If neuron i is in the first hidden layer (l = 1), then set $y_i^{(0)}(n) = x_i(n)$.

Backward computation (local gradients)

$$\delta_i(n) = -\frac{\partial E_i}{\partial v_i} \tag{3.9}$$

is called the local error or local gradients. Equation (3.9) can be simplified to

$$d_i^L(n) = e_i^L(n) f'(v_i^L(n)) \qquad \text{(for neuron } i \text{ in output layer } L\text{)} \tag{3.10a}$$

$$d_i^l(n) = f'(v_i^l(n)) \sum_k d_k^{(l+1)}(n) w_{ki}^{(l+1)}(n) \qquad \text{(for neuron } i \text{ in hidden layer } l\text{)} \tag{3.10b}$$

where $f'(\cdot)$ is the derivative of the activation function with respect to $v(n)$. If the activation function is chosen to be the hyperbolic tangent function given in Table 3.1, then $f'(\cdot)$ is

$$f'(v_i) = \frac{d f(v_i)}{d v_i} = g(1 - f^2(v_i)) \tag{3.11}$$

Hence, adjust the weights of the network in layer l according to the generalized delta rule:

$$w_{ij}^{(l)}(n+1) = w_{ij}^{(l)}(n) + \mu \delta_i^{(l)}(n) y_j^{(l-1)}(n) \tag{3.12}$$

where μ is the positive constant learning rate, usually equals 0.01.

If after updating the weights, the error E is not minimized, new epochs/iterations are required.

3.4 Radial Basis Function Network

The backpropagation algorithm for the design of a multilayer perceptron described earlier may be viewed as a form of *stochastic approximation*. *Radial basis functions* (RBFs) take a different approach by viewing the design of a neural network as a *curve-fitting problem* by finding a best fit to the training data in a multidimensional space. The use of RBF in the design of neural networks was first introduced by Broomhead and Lowe in 1988 [Hay94].

The RBF network basically involves three entirely different layers; an input layer, a hidden layer of high enough dimension, and an output layer. The transformation from the hidden unit to the output space is *linear*. Each output node is the weighted sums of the outputs of the hidden layer. However, the transformation from the input layer to the hidden layer is *nonlinear*. Each neuron or node in the hidden layer forming a linear combination of the basis (or kernel) functions which produces a localized response with respect to the input signals. This is to say that RBF produce a significant nonzero response only when the input falls within a small localized region of the input space. Figure 3.4 illustrates the RBF neural network structure. The most common basis of the RBF is a Gaussian kernel function of the form:

$$\varphi_l(\underline{x}) = \exp\left[-\frac{(\underline{x}-\underline{c}_l)^T(\underline{x}-\underline{c}_l)}{2\sigma_l^2}\right], \qquad l = 1,2,\text{K}\ ,L \tag{3.13}$$

where φ_l is the output of the lth node in hidden layer, $\underline{x}$ is the input pattern, $\underline{c}_l$ is the weight vector for the lth node in hidden layer, i.e., the center of the Gaussian for node l; σ_l^2 is the

normalization parameter (the measure of spread) for the *l*th node, and *L* is the number of nodes in the hidden layer. The outputs are in the range from zero to one so that the closer the input is to the center of the Gaussian, the larger the response of the node. The name RBF comes from the fact that these Gaussian kernels are radially symmetric; that is, each node produces an identical output for inputs that lie a fixed radial distance from the center of the kernel $\underline{c}_l$.

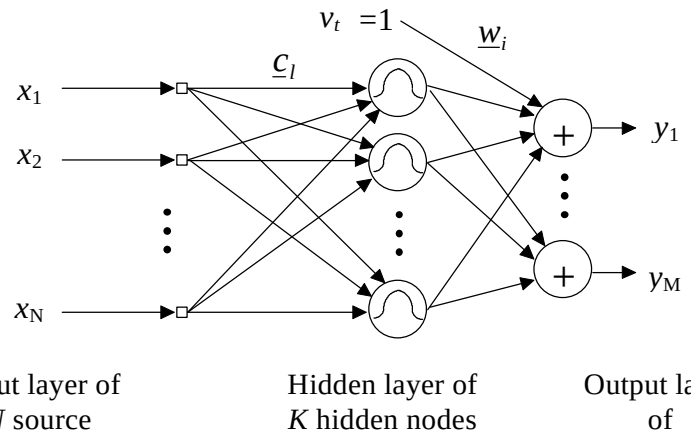


Figure 3.4 Radial Basis Function Network with a Gaussian Kernel

The network outputs are given by

$$y_i = \underline{w}_i^T \varphi(\underline{x}), \qquad i = 1,2,\text{K}\ ,M \tag{3.14}$$

where y_i is the output of the *i*th node, $\underline{w}_i$ is the weight vector for this node, *M* is the number of nodes in the output layer, and $\varphi(\underline{x})$ is the vector of outputs from the hidden layer (augmented with an additional component or a bias which assumes a value of one).

There are two common ways to calculate the measure of spread σ_l^2 of the data associated with each node

1. Find the measure of spread from the set of all training patterns grouped with each cluster center $\underline{c}_l$, i.e., set them equal to the average distance between the cluster centers and the training patterns:

$$\sigma_l^2 = \frac{1}{N_l} \sum_{k \in \underline{c}_l} (\underline{x}_k - \underline{c}_l)^T (\underline{x}_k - \underline{c}_l), \qquad l = 1,2,\text{K}\ ,L \tag{3.15}$$

where N_l = the number of patterns that belongs to the *l*th cluster
k = the index number of a pattern that belongs to the *l*th cluster

2. Find the measure of spread from among the centers (*p*-nearest neighbor heuristic):

$$\sigma_l^2 = \frac{1}{L} \sum_{k=1}^{L} (\underline{c}_k - \underline{c}_l)^T (\underline{c}_k - \underline{c}_l), \qquad l = 1,2,\text{K}\ ,L \tag{3.16}$$

A popular choice of clustering algorithms used in training the RBF network (to give a class *k*) is the generalized Lloyd algorithm or the *K*-means clustering algorithm. The algorithm developed by J. MacQueen in 1967 provides a simple mechanism for minimizing the sum of squared errors with k clusters, with each cluster consisting of a set of N_l samples $\underline{x}_1, \underline{x}_2, \text{K}\ , \underline{x}_{N_l}$ that are similar to each other. The *K*-means algorithm proceeds as follows:

1. Choose a set of clusters $\{\underline{c}_1, \underline{c}_2, \text{K}\ , \underline{c}_k\}$ arbitrarily.

2. Assign the N_l samples to the k clusters using the minimum Euclidean distance rule:

$$\underline{x} \in \underline{c}_l \text{ if } \|\underline{x} - \underline{c}_l\| = \min_l \|\underline{x} - \underline{c}_l\| \tag{3.17}$$

3. Compute new cluster centers

$$\underline{c}_k = \frac{1}{N_l} \sum_{k \in \underline{c}_l} \underline{x}_k \tag{3.18}$$

4. If any cluster center changes, return to step 2; otherwise, stop.

Chapter 4

Fundamentals of Wavelets Theory

The term "*wavelet*" as it implies means a little wave. This little wave must have at least a minimum oscillation and a fast decay to zero, in both the 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 [You93]. Figure 4.1a is an example of a wavelet called "Morlet wavelet" named after Jean Morlet, the inventor, in 1984 [GGM84].

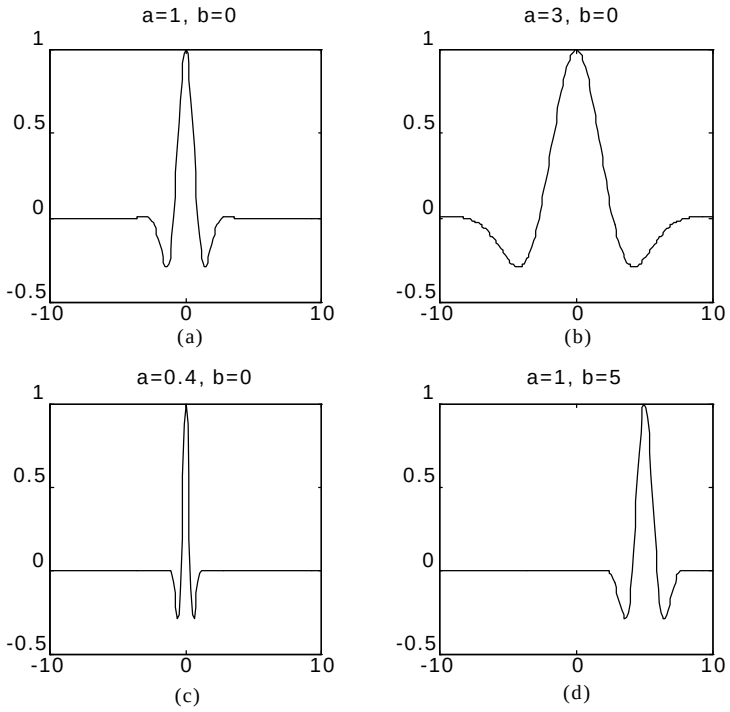


Figure 4.1 Dilated and Translated Morlet Mother Wavelets

Sets of "wavelets" are employed to approximate a signal and the goal is to find a set of daughter wavelets constructed by a dilated (scaled or compressed) and translated (shifted) original wavelets or mother wavelets 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. Figure 4.1b-d display various daughter wavelets where *a*

is a dilation and *b* is a translation corresponding to the Morlet mother wavelet. Note the constant shape of these daughter wavelets; the same number of oscillations is in each wavelet.

4.1 Wavelet Transform

The *wavelet transform* is an operation that transforms a function by integrating it with modified versions of some kernel function [CGT89]. 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. Recall, from the beginning of the chapter that for a function to be a wavelet it must be oscillatory and have fast decay toward zero. If these conditions are combined with the condition that the wavelet must also integrate to zero (its "d.c." or zero frequency component is zero), then these three conditions are said to satisfy **the Grossmann-Morlet admissibility condition** . More rigorously, a function $h \in L^2(R)$ (the set of all square integrable or a finite energy function) is admissible if [GM84]

$$c_h = \int_{-\infty}^{\infty} \frac{|H(\omega)|^2}{|\omega|} d\omega < \infty \tag{4.1}$$

where $H(\omega)$ 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 [Wei94]. 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 [SRS92]:

$$W_f(a,b) = \int_{-\infty}^{\infty} f(t)\, h^*_{a,b}(t)dt \tag{4.2}$$

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:

$$h_{a,b}(t) = \frac{1}{\sqrt{a}} h\left(\frac{t-b}{a}\right) \tag{4.3}$$

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. Moreover, the admissibility condition expressed in Eq. (4.1) yields **the resolution of identity** property of wavelet [You93]:

$$c_h \langle f_1, f_2 \rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \langle f_1, h_{a,b} \rangle \langle h_{a,b}, f_2 \rangle \frac{da\ db}{a^2} \tag{4.4}$$

where $f_1, f_2 \in L^2(R)$, $h(t)$ be admissible, with a not equal to zero, and $\langle , \rangle$ denotes the inner product:

$$\langle f_1, f_2 \rangle = \frac{1}{2\pi} \int_{-\infty}^{\infty} F_1(\omega) F_2^*(\omega)\, d\omega = \int_{-\infty}^{\infty} f_1(t) f_2^*(t)\, dt \tag{4.5}$$

Practically, the resolution of identity is a formula that allows two wavelet transforms to be combined into one wavelet transform (an operator that maps a product of inner products to a single inner product). When $f_1 = f_2$, the resolution of identity (4.4) becomes **the energy conservation**

$$c_h \int_{-\infty}^{\infty} |f(t)|^2\, dt = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |W_f(a,b)|^2 \frac{da\ db}{a^2} \tag{4.6}$$

which says the energy in both the time domain and the wavelet transform domain are equal (equivalent to the Parseval energy relation of the Fourier transform). So within a scale factor, the wavelet transform is an *isometry* from $L^2(R) \rightarrow L^2(R^2)$. The energy distribution in the scale-translation plane or the wavelet transform domain, i.e., the right hand side of Eq. (4.6) is termed the ***wavelet spectrogram***, or ***scalogram*** [RV91]. It is similar to the *spectrogram* defined as the squared modulus of the short time Fourier transform. Both the spectrogram and the scalogram produce a more or less easily interpretable visual two-dimensional representation of signals where each pattern in the time-frequency or time-scale plane contributes to the global energy of the signal. However, there are some disadvantages; the spectogram, as well as the scalogram, cannot be inverted, and the undesirable existence of *cross terms* appear as interferences between patterns in the time-frequency or time-scale plane

[KB92]. More involved representations have been developed to investigate these effects, for example, the Wigner distribution, Choi-Williams distribution, Cone-kernel distribution, Page distribution, and so forth [HB92], [Coh95], [LAP93], [Ril89].

The **inversion formula** of the wavelet transform is given by [Dau90]:

$$f(t) = \frac{1}{c_h} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_f(a,b) h_{a,b}(t) \frac{da\ db}{a^2} \tag{4.7a}$$

To constitute a frame, the discrete wavelet transform generated by sampling the wavelet parameters (time/scale) on a grid or lattice, which is not equally spaced, must satisfy the admissibility condition and the lattice points must be sufficiently close to satisfy basic information-theoretic needs. The family of basis functions $\{h_{a,b}(t)\}$ with $a, b \in Z$ is said to be a frame if it satisfies the property that there exists two frame bounds $A > 0$ and $B < \infty$ where A, B are independent of $f(t)$ such that for all the functions $h \in L^2(R)$, the sum of square moduli of $W_f(a,b)$ must lie between the two positive bounds:

$$A\|f\|^2 \le \sum_{a,b} \left| \langle f, h_{a,b} \rangle \right|^2 \le B\|f\|^2 \tag{4.7b}$$

For more properties of the wavelet transform such as the *regularity* or smoothness, finite support, marginals, etc., see the references [Wei94] and [HB92] for more rigorous derivations and proofs see [GM84], [Dau90], [JS94], [Kai94], [BF94], and [Coh95], and for more historical context see [Mey93], and [Res91].

Recently, new classes of frame wavelets which are in general more attractive than orthonormal wavelets, e.g. Daubechies's wavelet have been made available for a wide variety of mother wavelet choices (of course, under the constraints of admissibility condition) [IRR95]. These wavelet choices provide extended abilities to generalize adaptive signal processes particularly in neural network applications. However, these new classes of frame wavelets do not present better performance or advantages over others, but rather to characterize new families of wavelets that are simple and available to effect the needs called for in diverse applications. Three classes of general frame wavelets mainly derived from the viewpoints of complex residues, *sigmoids*, and *Hermitian operator*, together with a few familiar ones like Morlet and Shannon wavelets, will be included in the following sections.

4.2 Types of Wavelet Filters

4.2.1 Morlet Wavelet

The Morlet's basic wavelet function is a multiplication of the Fourier basis with a Gaussian window [MM87]:

$$h(t) = \exp(jw_0 t)\exp(-0.5t^2) \tag{4.8}$$

Its real part is a Cos-Gaussian and the imaginary part is a Sin-Gaussian function. The Cos-Gaussian wavelet is a real even function. The Fourier transform of the Cos-Gaussian wavelet is the Gaussian functions shifted to ω_0 and $-\omega_0$, respectively:

$$H(w) = \sqrt{\frac{p}{2}}\,(\exp[-0.5(w-w_0)^2] + \exp[-0.5(w+w_0)^2]) \tag{4.9}$$

which is even and real positive valued. Figure 4.2 provides the plots of the Morlet wavelet and its Fourier spectrum, with $\omega_0 = 4$.

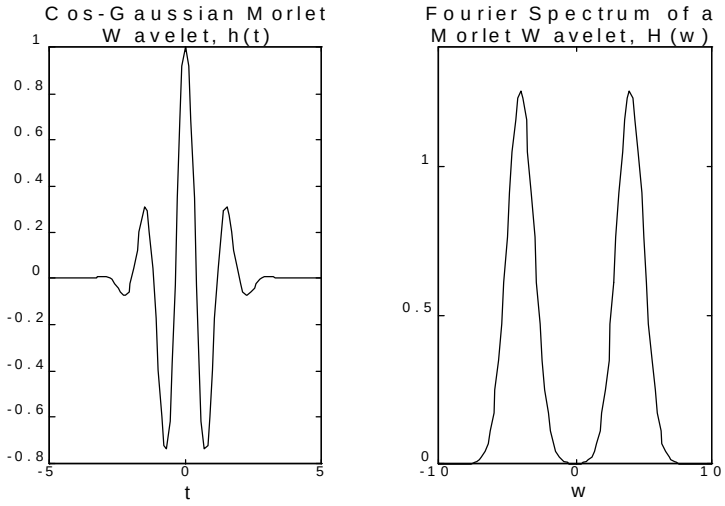


Figure 4.2 Cos-Gaussian Morlet Wavelet $h(t)$ and its Fourier Transform $H(\omega)$

The Morlet wavelets do not satisfy the wavelet admissible condition, because

$$H(0) = \sqrt{2\pi}\,\exp(-0.5w_0^{\,2}) \neq 0 \tag{4.10}$$

which leads to $c_h = +\infty$. However, if ω_0 is sufficiently large, say $\omega_0 = 4$, $H(0)$ comes very close to zero (0.00084) and can be practically considered as zero in numerical computations.

4.2.2 RASP Wavelets

RAtional functions with **S**econd-order **P**oles (RASP) arise from the residue theorem of complex variables [IRR95]. Figure 4.3 presents three examples of the RASP mother wavelets with the normalized gains k_i, i = 1, 2, 3 equal to 3.0788, 2.7435, and 0.6111, respectively. These wavelets are real, odd functions with zero mean. The distinction among these mother wavelets is their rational form of functions being strictly proper and having simple/double poles of second order.

According to the admissibility condition (4.1), it must be shown that the families of RASP mother wavelets have zero mean value. This can be done easily using the residue theorem for evaluation of integrals. As an example RASP2 wavelet of Figure 4.3 will be analyzed and others follow in a similar manner.

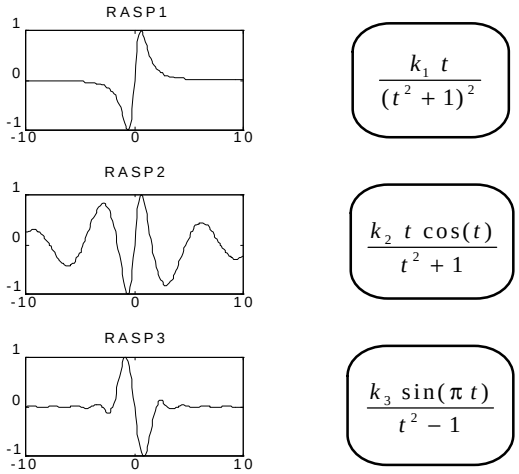


Figure 4.3 Examples of **RA**tional functions with **S**econd-order **P**oles (RASP) Wavelets

In order to show that the RASP2 mother wavelet $h(t) = \dfrac{t\cos(t)}{t^2+1}$ is zero mean, consider the integral in the complex plane $\oint_C \dfrac{z\,e^{jz}}{z^2+1}\,dz$ where C is the contour of a simple closed path of Figure 4.4. The integrand has simple poles at $z = \pm j$, but only $z = j$ lies inside C. According to **the Residue Theorem**: Let $f(z)$ be analytic/holomorphic (its derivative exists at all points z) inside and on a simple closed curve C except at the singularities z_i enclosed by C. Then the residue theorem states that [You93]

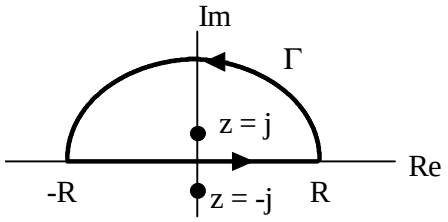


Figure 4.4 A Simple Closed Path Contour *C*

$$\oint_C f(z)\,dz = 2\pi j \sum_{i=1}^{n} \text{Res}[\,f\,;z_i\,] \tag{4.11}$$

Let *f* have a pole of order *k* at z_i. Then

$$\text{Res}[\,f\,;z_i\,] = \lim_{z\to z_i} \frac{1}{(k-1)!}\frac{d^{k-1}}{dz^{k-1}}\left\{(z-z_i)^k f(z)\right\} \tag{4.12}$$

Thus the residue of $h(z)$ at $z = j$ is

$$\text{Res}\,[\,h;j\,] = \lim_{z\to j}(z-j)\frac{z\,e^{jz}}{(z+j)(z-j)} = \frac{1}{2e} \tag{4.13}$$

and so from (4.11),

$$\oint_C \frac{z\,e^{jz}}{z^2+1}\,dz = 2\pi j\,\text{Res}[h;j\,] = \frac{\pi}{e}j \tag{4.14}$$

or

$$\int_{-R}^{R}\frac{x\,e^{jx}}{x^2+1}\,dx + \int_{\Gamma}\frac{z\,e^{jz}}{z^2+1}dz = \frac{\pi}{e}j \tag{4.15}$$

i.e.,

$$\int_{-R}^{R}\frac{x\cos(x)}{x^2+1}\,dx + j\int_{-R}^{R}\frac{x\sin(x)}{x^2+1}\,dx + \int_{\Gamma}\frac{z\,e^{jz}}{z^2+1}dz = \frac{\pi}{e}j \tag{4.16}$$

and so the real part is

$$\int_{-R}^{R}\frac{x\cos(x)}{x^2+1}\,dx + \int_{\Gamma}\frac{z\,e^{jz}}{z^2+1}dz = 0 \tag{4.17}$$

Taking the limit as $R\to\infty$ and show that the integral around Γ approaches zero,

$$\int_{\Gamma}\frac{z\,e^{jz}}{z^2+1}dz = \lim_{R\to\infty}\int_0^{\pi}\frac{Re^{j\theta}\,e^{j\,\text{Re}^{j\theta}}Re^{j\theta}\,j\,d\theta}{R^2e^{j2\theta}+1} \le \int_0^{\pi}\lim_{R\to\infty}\frac{R^2\left|e^{-R\sin\theta}\right|d\theta}{R^2-1} \tag{4.18}$$

Note that we have made use of the inequality $|A+B|\ge|A|-|B|$ in the denominator and $\left|\int AB\,d\theta\right| \le \int|A||B|\,d\theta$ in the numerator. So as $R\to\infty$, the exponential term in Eq. (4.18)

approaches zero. Therefore Eq. (4.17) simplifies to $\lim_{R\to\infty}\int_{-R}^{R}\frac{x\cos(x)}{x^2+1}dx = 0$, and the mother wavelet $h(t) = \frac{t\cos(t)}{t^2+1}$ has zero mean.

4.2.3 SLOG Wavelets

The **S**uperposed **LOG**istic functions (SLOG) results from finite term sums of weighted and delayed *logistic* functions. A logistic function is a type of monotonically increasing, smooth, asymptotic sigmoid (*S*-shaped) function which usually represents the threshold function at the neuron output of the neural networks model [Hay94]. A logistic function characterizes an asymmetric unipolar representation having the form:

$$f_{logistic}(t) = \frac{1}{1+e^{-t}} \tag{4.19}$$

Figure 4.5 displays the logistic function.

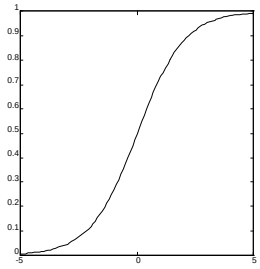


Figure 4.5 A Logistic Sigmoid Function

This family of mother wavelets, thus, specifically arises from the neural networks representation and classification problems. The ability that a neural network input-output function can realize a wavelet demonstrates that the neural network has the same *universal approximation* properties as wavelets, i.e., the ability to represent arbitrary L^2 mappings [ZB92].

Recently, the authors in [PK93] have developed constructively the proof for the representation theory of universal approximation in $L^2(R)$ by showing that the implementation of feedforward neural networks effect the construction of an *affine* frame for $L^2(R)$. The authors' theorem states that "**Feedforward neural networks with sigmoidal activation functions and a single hidden layer can represent any function** $f \in L^2(R)$." [PK93] The derivation of the proof came under the time-frequency analysis *frame* property in multiresolution signal decomposition [AH92]. To constitute a frame, the *discrete* wavelet transform generated by sampling the wavelet parameters (time/scale) on a grid or lattice, which is not equally spaced (unlike the short-time Fourier transform, STFT), must satisfy the admissibility condition and the lattice points must be sufficiently close to satisfy basic information-theoretic needs. Thus the question of reconstruction of the signal from its transform values naturally depends on the *coarseness* of the sampling grid, i.e., how close the spacing is. A fine grid permits easy reconstruction, but with evident redundancy, i.e., *oversampling*. A too-coarse grid results in loss of information. The goal is to obtain a set of wavelet functions in discretized parameters on the sampling grid called *affine* wavelets such that the set is minimal, i.e., nonredundant. Such complete sets are called *frames*. Note that the *tightest* set (exact frame) is the *orthonormal wavelet* set [Dau90].

Figure 4.6a represents the first SLOG mother wavelet exhibiting the following superposition of logistic sigmoids:

$$h_{slog1}(t) = \frac{1}{1+e^{-t+1}} - \frac{1}{1+e^{-t+3}} - \frac{1}{1+e^{-t-3}} + \frac{1}{1+e^{-t-1}} \tag{4.20}$$

This mother wavelet function represents a minimum-oscillation wavelet (down-up-down).

Figure 4.6b demonstrates another SLOG wavelet having an additional half-oscillation (down-up-down-up). Again, a finite sum of weighted and delayed logistic functions is employed:

$$h_{slog2}(t) = \frac{3}{1+e^{-t-1}} - \frac{3}{1+e^{-t+1}} - \frac{1}{1+e^{-t-3}} + \frac{1}{1+e^{-t+3}} \tag{4.21}$$

Similarly, the whole class of new family SLOG frame wavelets with greater numbers of half-oscillations can be generated the same way.

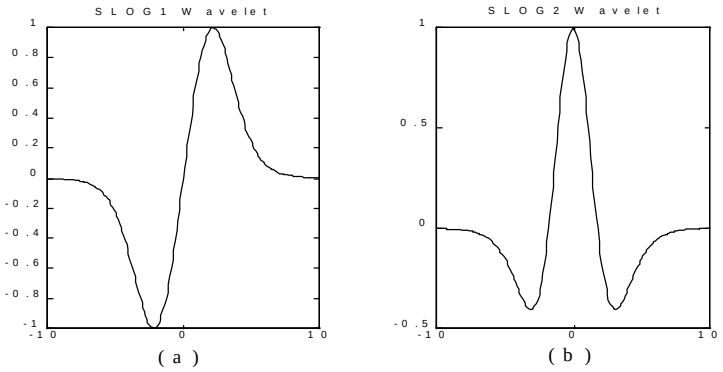


Figure 4.6 Examples of The **S**uperposed **LOG**istic functions (SLOG) Mother Wavelets

Note: All plots were normalized to have unity amplitudes.

4.2.4 POLYWOG Wavelets

Numerous useful mother wavelets arise from **POLY**nomials **W**ind**O**wed with **G**aussians (POLYWOG) types of functions. For example, all derivatives of a Gaussian function are POLYWOGs and so are admissible mother wavelets. Four examples are given below:

POLYWOG1: $$h_{POLYWOG\,1} = k_1\, t\, \exp\left(-\frac{t^2}{2}\right), \quad k_1 = \sqrt{e} \tag{4.22}$$

POLYWOG2: $$h_{POLYWOG\,2} = k_2\, \left(t^3 - 3t\right) \exp\left(-\frac{t^2}{2}\right), \quad k_2 = 0.7246 \tag{4.23}$$

POLYWOG3: $$h_{POLYWOG\,3} = k_3\, \left(t^4 - 6t^2 + 3\right) \exp\left(-\frac{t^2}{2}\right), \quad k_3 = \frac{1}{3} \tag{4.24}$$

POLYWOG4: $$h_{POLYWOG\,4} = \left(1 - t^2\right) \exp\left(-\frac{t^2}{2}\right) \tag{4.25}$$

Figure 4.7 shows the four examples of derivatives of a Gaussian function POLYWOGs. Several types of POLYWOGs may also be found using many simple generators. In particular, POLYWOG wavelets may be generated from the *Hermiticity* of the derivative of the polynomial $f(t)$ and Gaussian functions $g(t)$, where $f(t)$ and $g(t)$ are square integrable $L^2(R)$. Let's review the *Hermitian* or self adjoint operator for the representation of the physical quantity in the same sense as the Fourier transform representation which is the appropriate representation for the physical quantity "*frequency*". Let a signal be expanded in the form

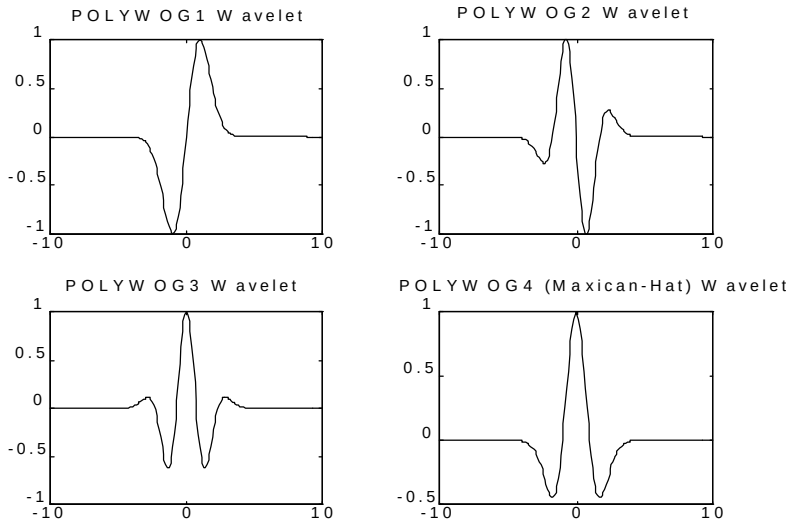


Figure 4.7 Examples of **POLY**nomials **W**ind**O**wed with **G**aussians (POLYWOG) Wavelets

$$f(t) = \int_{-\infty}^{\infty} F(a)\, u(a,t)\, da \tag{4.26}$$

where $u(a,t)$ are the basis functions and $F(a)$ are *the expansion coefficients* or the "transform of the signal". It will be shown that $F(a)$ is given by

$$F(a) = \int_{-\infty}^{\infty} f(t)\, u^*(a,t)\, dt \tag{4.27}$$

where * denotes a complex conjugate. The basis functions, $u(a,t)$ are always functions of two variables, in this case time and a. The a's are the physical quantity, and $F(a)$ gives an

indication of how important a particular value of a is for the signal at hand. If $F(a)$ is relatively large only in a particular region, then the signal is said to be concentrated at those values of a.

An operator is *Hermitian* if for any pair of functions, $f(t)$ and $g(t)$ [Coh95]

$$\int_{-\infty}^{\infty} g^*(t)\, \mathbb{H}\, f(t)\, dt = \int_{-\infty}^{\infty} f(t)\, \{\mathbb{H}\, g(t)\}^*\, dt \tag{4.28}$$

The Hermiticity of the operator guarantees that the eigenfunctions are complete and orthogonal. This means that the eigenfunctions satisfy

$$\int_{-\infty}^{\infty} u^*(a',t)\, u(a,t)\, dt = \delta(a-a') \tag{4.29a}$$

$$\int_{-\infty}^{\infty} u^*(a,t')\, u(a,t)\, da = \delta(t-t') \tag{4.29b}$$

It is these properties of the expansion functions that allow the transform between $f(t)$ and $F(a)$ as given by Eqs. (4.26) and (4.27). In particular, to see how to obtain $F(a)$, multiply Eq. (4.26) by $u^*(a',t)$ and integrate with respect to time.

$$\int_{-\infty}^{\infty} f(t)\, u^*(a',t)\ dt = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(a)\, u(a,t)\, u^*(a',t)\ da\ dt \tag{4.30a}$$

$$= \int_{-\infty}^{\infty} F(a)\, \delta(a-a')\ da \tag{4.30b}$$

$$= F(a') \tag{4.30c}$$

which proves the inverse relation, Eq. (4.27).

Also, if the operator is Hermitian the eigenvalues are guaranteed to be **real**. This is very important because in nature measurable quantities are real [Coh95].

Following the Hermiticity of the operator, now we can show that the operation of differentiation is Hermitian. Consider the operator $\mathcal{H} = \frac{d}{dt}$ defined in $L^2(R)$ and any two functions, $f(t)$ and $g(t)$ belong to $L^2(R)$. Then by integration by parts,

$$\int_{-\infty}^{\infty} g^*(t)\frac{d}{dt}f(t)\ dt = f\ g^*\Big|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} f(t)\frac{d}{dt}g^*(t)\ dt \tag{4.31}$$

The evaluation term $f\ g^*\Big|_{-\infty}^{\infty}$ is zero since $f(t)$ and $g(t) \in L^2(R)$. Alternatively, Eq. (4.31) can be written as:

$$\langle Df, g\rangle = \langle f, -Dg\rangle \tag{4.32}$$

If we let $f(t)$ and $g(t) \in L^2(R)$ be any windowed polynomial function and a Gaussian function, respectively, a useful formula (4.32) can be generated as new mother wavelets:

$$\int_{-\infty}^{\infty} \exp\left(-\frac{t^2}{2}\right)\frac{d}{dt}f(t)\ dt = -\int_{-\infty}^{\infty} f(t)\ t\ \exp\left(-\frac{t^2}{2}\right)\ dt \tag{4.33}$$

For example, the POLYWOG4 wavelet in Fig. 4.7 is widely known as the Mexican-hat wavelet introduced by Gabor and is well known as the Laplacian operator, mostly used for zero-crossing multiresolution image edge detection [SRS92]. This wavelet is in fact the second derivative of the Gaussian function:

$$h_{POLYWOG\,4} = (1-t^2)\exp\left(-\frac{t^2}{2}\right) \tag{4.25}$$

The Mexican-hat wavelet is even and real valued. The Fourier transform of the Mexican-hat wavelet is

$$H(\omega) = -\omega^2 \exp\left(-\frac{\omega^2}{2}\right) \tag{4.34}$$

which is also even and real valued and satisfies the admissible condition, $H(0) = 0$, i.e.,

$$\int_{-\infty}^{\infty} (1-t^2)\exp\left(-\frac{t^2}{2}\right) dt = 0 \tag{4.35}$$

Using the Hermiticity of the differentiation from Eq. (4.33) to Eq. (4.35), we obtain

$$\int_{-\infty}^{\infty} (1-t^2)\exp\left(-\frac{t^2}{2}\right) dt = \int_{-\infty}^{\infty} \left(t-\frac{t^3}{3}\right) t\exp\left(-\frac{t^2}{2}\right) = 0 \tag{4.36}$$

which is a new POLYWOG mother wavelet having sufficient localization properties with mean-zero. Figure 4.8 provides a plot of a new POLYWOG admissible mother wavelet.

$$h_{POLYWOG\,5} = k_5 (3t^2 - t^4)\exp\left(-\frac{t^2}{2}\right) \tag{4.37}$$

where k_5 is the normalized gain equals to 0.8244.

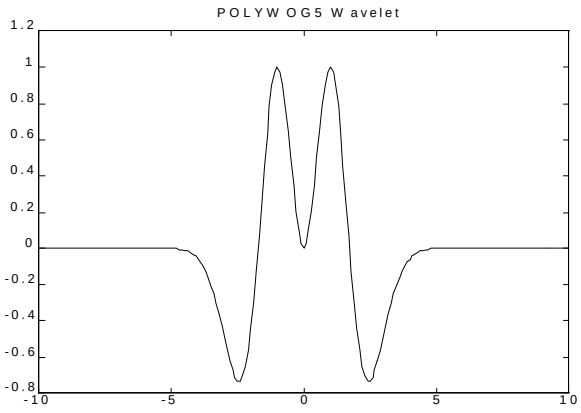


Figure 4.8 Hermiticity of Derivative **POLY**nomials **W**ind**O**wed with **G**aussians Wavelet

4.2.5 Shannon Wavelet

Shannon wavelet is a mother wavelet that generates an *orthonormal* basis for $L^2(R)$. These wavelets are discontinuous in frequency and hence spread out in time. For any function $\phi(t) \in L^2(R)$ to form an orthogonal basis, $\phi(t)$ must satisfy the orthonormality condition.

$$\langle \phi(t-k), \phi(t-l) \rangle = \delta(k-l) \tag{4.38}$$

or in Fourier domain, it is sufficient that [Zay93]

$$\sum_{k=-\infty}^{\infty} \left| \Phi(\omega + 2k\pi) \right|^2 = 1 \tag{4.39}$$

The sum of the series of the discretely translated Fourier spectrum $\left|\Phi(\omega)\right|^2$ must be unity.

For example, from the Shannon sampling theorem, the functions

$$\phi(t-k) = \frac{\sin \pi(t-k)}{\pi(t-k)} \tag{4.40}$$

constitute an orthonormal basis requiring that the Fourier transform Φ have compact support $L^2[-\pi, \pi]$, then for any function $f(t)$ can be expressed as:

$$f(t) = \sum_{k=-\infty}^{\infty} f(k)\, \phi(t-k) \tag{4.41}$$

The orthonormality can be easily demonstrated in the frequency domain. With

$$\phi(t) = \frac{\sin \pi t}{\pi t} \quad \leftrightarrow \quad \Phi(\omega) = \begin{cases} 1 & |\omega| < \pi \\ 0 \text{ otherwise} \end{cases} \tag{4.42}$$

the inner product defined in (4.5) is just

$$\int_{-\infty}^{\infty} \phi(t-k)\, \phi^*(t-l)\, dt = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-j\omega k} \Phi(\omega)\, e^{j\omega l} \Phi^*(\omega)\, d\omega \tag{4.43a}$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-j\omega(k-l)}\, d\omega = \delta(k-l) \tag{4.43b}$$

Similarly, if a basic wavelet function $h(t) \in L^2(R)$ satisfies the wavelet admissibility condition, and its Fourier transform of $h(t)$ satisfy the orthonormality condition (4.39):

$$\sum_{k=-\infty}^{\infty} \left| H(\omega + 2k\pi) \right|^2 = 1, \tag{4.44}$$

the discrete translations of the wavelets $h(t-k)$ also form an orthonormal basis.

The simplest solution for (4.44) is [Zay93]

$$H(\omega) = \begin{cases} 1 & \pi < |\omega| < 2\pi \\ 0 & \text{otherwise} \end{cases} \tag{4.45}$$

which leads to the **Shannon wavelet**

$$h_{Shannon}(t) = \frac{\sin 2\pi t - \sin \pi t}{\pi t} \tag{4.46}$$

In general, Shannon wavelet is the real part of the *Harmonic* wavelet: [New93]

$$h(t) = \frac{\exp(j4\pi t) - \exp(j2\pi t)}{j2\pi t} \tag{4.47}$$

The Shannon wavelet is drawn in Fig. 4.9.

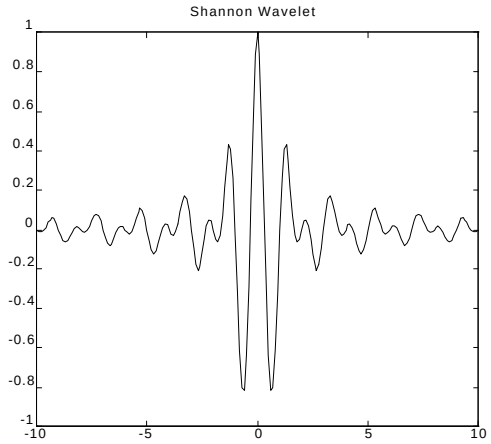


Figure 4.9 Shannon Mother Wavelet

Chapter 5

Neural Network Adaptive Wavelets (Wavenets)

Combining the wavelet transform theory of Chapter 4 with the basic concept of neural networks of Chapter 3, a new mapping network called *neural network adaptive wavelets* or *wavenets* is proposed as an alternative to feedforward neural networks for approximating arbitrary nonlinear functions. The *wavenet* algorithms consist of two processes: the self-construction of networks and the minimization of error. In the first process, the network structures applied for representation are determined by using wavelet analysis. The network gradually recruits hidden units to effectively and sufficiently cover the time-frequency region occupied by a given target. Simultaneously, the network parameters are updated to preserve the network topology and take advantage of the later process. In the second process, the approximations of instantaneous errors are minimized using an adaptation technique based on the LMS algorithms described in chapter 2. The parameter of the initialized network is updated using the steepest gradient-descent method of minimization. Each hidden unit has a square window in the time-frequency plane. The optimization rule is only applied to the hidden units where the selected point falls into their windows. Therefore, the learning cost can be reduced.

5.1 *Wavenet* Algorithms

The *wavenets* architecture shown in Figure 5.1 approximates any desired signal $y(t)$ by generalizing a linear combination of a set of daughter wavelets $h_{a,b}(t)$, where $h_{a,b}(t)$ are generated by dilation, a, and translation, b, from a mother wavelet $h(t)$:

$$h_{a,b}(t) = h\left(\frac{t-b}{a}\right) \tag{5.1}$$

with the dilation factor $a > 0$. Note that equation (5.1) is similar to that of equation (4.3), but without the energy normalization.

The inversion formula of (4.7) cannot be expressed directly by finite neural networks (NN), but can be approximately realized using NNs topology with finite hidden units. This is so because most targets are restricted in both the time and frequency domains. Assume that the network output function satisfies the admissibility condition and the network sufficiently approximates the target, i.e., the time-frequency region is effectively covered by their K windows. The approximated signal of the network $\hat{y}(t)$ can be represented by:

$$\hat{y}(t) = u(t)\sum_{k=1}^{K} w_k h_{a_k,b_k}(t) \tag{5.2}$$

where K is a number of windowing wavelets, and w_k is the weight coefficients.

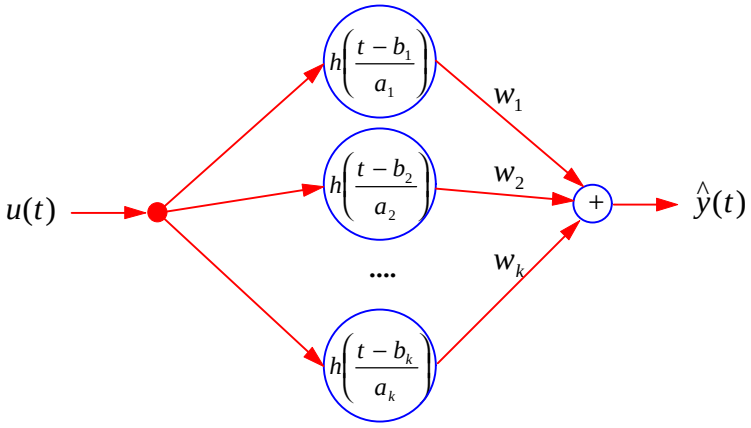


Figure 5.1 Adaptive *Wavenets* Structure

The neural network parameters w_k, a_k, and b_k can be optimized in the LMS sense by minimizing a cost function or the energy function, E, over all time t. Thus by denoting

$$e(t) = y(t) - \hat{y}(t) \tag{5.3}$$

be a time-varying error function at time t, where $y(t)$ is the desired (target) response. The energy function is defined by

$$E = \frac{1}{2}\sum_{t=1}^{T} e^2(t) \tag{5.4}$$

To minimize E we may use the method of steepest descent, which requires the gradients $\frac{\partial E}{\partial w_k}$, $\frac{\partial E}{\partial a_k}$, and $\frac{\partial E}{\partial b_k}$ for updating the incremental changes to each particular parameter w_k, a_k, and b_k, respectively. For any mother wavelet given in Table 5.1, gradients of E are

$$\frac{\partial E}{\partial w_k} = -\sum_{t=1}^{T} e(t)\, h(\tau) u(t) \tag{5.5}$$

$$\frac{\partial E}{\partial b_k} = -\sum_{t=1}^{T} e(t) u(t)\, w_k \frac{\partial h(\tau)}{\partial b_k} \tag{5.6}$$

$$\frac{\partial E}{\partial a_k} = -\sum_{t=1}^{T} e(t) u(t)\, w_k \tau \frac{\partial h(\tau)}{\partial b_k} = \tau \frac{\partial E}{\partial b_k} \tag{5.7}$$

where $\tau = \dfrac{t - b_k}{a_k}$. (5.8)

The incremental changes of each coefficient are simply the negative of their gradients,

$$\Delta w = -\frac{\partial E}{\partial w}, \quad \Delta b = -\frac{\partial E}{\partial b}, \quad \Delta a = -\frac{\partial E}{\partial a} \tag{5.9}$$

thus each coefficient $\underline{w}$, $\underline{b}$, and $\underline{a}$ of the network is updated in accordance with the rule

$$\underline{w}(n+1) = \underline{w}(n) + \mu_w \Delta \underline{w} \tag{5.10}$$

$$\underline{b}(n+1) = \underline{b}(n) + \mu_b \Delta \underline{b} \tag{5.11}$$

$$\underline{a}(n+1) = \underline{a}(n) + \mu_a \Delta \underline{a} \tag{5.12}$$

where μ is the fixed learning rate parameter.

Following the previous chapter, the derivatives with respect to its translation of the various wavelet filter, $\dfrac{\partial h(\tau)}{\partial b}$, are given in Table 5.1.

Table 5.1 Wavelet Filters and Their Derivatives

Name	$h(\tau)$	$\dfrac{\partial h(\tau)}{\partial b}$
Morlet	$\cos(\omega_0 \tau) \exp\left(-0.5\tau^2\right)$	$\dfrac{1}{a}\left[\omega_0 \sin(\omega_0 \tau) \exp\left(-0.5\tau^2\right) + \tau h(\tau)\right]$
RASP1	$\dfrac{\tau}{\left(\tau^2+1\right)^2}$	$\dfrac{1}{a}\dfrac{\left(3\tau^2-1\right)}{\left(\tau^2+1\right)^3}$
RASP2	$\dfrac{\tau \cos(\tau)}{\tau^2+1}$	$\dfrac{\tau\left(\dfrac{\tau^2+1}{a}\right)\sin(\tau) + \left(\dfrac{\tau^2-1}{a}\right)\cos(\tau)}{\left(\tau^2+1\right)^2}$

Table 5.1 Wavelet Filters and Their Derivatives (Continued)

Name	$h(\tau)$	$\frac{\partial h(\tau)}{\partial b}$
RASP3	$\frac{\sin(\pi\tau)}{\tau^2-1}$	$\frac{\left(\frac{2\tau}{a}\right)\sin(\pi\tau)-\pi\left(\frac{\tau^2-1}{a}\right)\cos(\pi\tau)}{(\tau^2-1)^2}$
SLOG1	$\frac{1}{1+e^{-\tau+1}}-\frac{1}{1+e^{-\tau+3}}$ $-\frac{1}{1+e^{-\tau-3}}+\frac{1}{1+e^{-\tau-1}}$	$\frac{1}{a}\left[-\frac{e^{-\tau+1}}{(1+e^{-\tau+1})^2}+\frac{e^{-\tau+3}}{(1+e^{-\tau+3})^2}\right.$ $\left.+\frac{e^{-\tau-3}}{(1+e^{-\tau-3})^2}-\frac{e^{-\tau-1}}{(1+e^{-\tau-1})^2}\right]$
SLOG2	$\frac{3}{1+e^{-\tau-1}}-\frac{3}{1+e^{-\tau+1}}$ $-\frac{1}{1+e^{-\tau-3}}+\frac{1}{1+e^{-\tau+3}}$	$\frac{1}{a}\left[-\frac{3e^{-\tau-1}}{(1+e^{-\tau-1})^2}+\frac{3e^{-\tau+1}}{(1+e^{-\tau+1})^2}\right.$ $\left.+\frac{e^{-\tau-3}}{(1+e^{-\tau-3})^2}-\frac{e^{-\tau+3}}{(1+e^{-\tau+3})^2}\right]$
POLYWOG1	$\tau\exp\left(-\frac{\tau^2}{2}\right)$	$\frac{1}{a}(\tau^2-1)\exp\left(-\frac{\tau^2}{2}\right)$
POLYWOG2	$(\tau^3-3\tau)\exp\left(-\frac{\tau^2}{2}\right)$	$\frac{1}{a}(\tau^4-6\tau^2+3)\exp\left(-\frac{\tau^2}{2}\right)$
POLYWOG3	$(\tau^4-6\tau^2+3)\exp\left(-\frac{\tau^2}{2}\right)$	$\frac{1}{a}(\tau^5-10\tau^3+15\tau)\exp\left(-\frac{\tau^2}{2}\right)$
POLYWOG4	$(1-\tau^2)\exp\left(-\frac{\tau^2}{2}\right)$	$\frac{1}{a}(3\tau-\tau^3)\exp\left(-\frac{\tau^2}{2}\right)$
POLYWOG5	$(3\tau^2-\tau^4)\exp\left(-\frac{\tau^2}{2}\right)$	$\frac{1}{a}(-\tau^5+7\tau^3-6\tau)\exp\left(-\frac{\tau^2}{2}\right)$
Shannon	$\frac{\sin 2\pi\tau-\sin\pi\tau}{\pi\tau}$	$\frac{\pi}{a}\frac{(-\pi\tau\cos\pi\tau-2\pi\cos 2\pi\tau+\sin\pi\tau+\sin 2\pi\tau)}{(\pi\tau)^2}$

5.2 *Wavenets* With IIR Block Structure

As previously mentioned, a *wavenet* is a *local network* in which the output function is well localized in both time and frequency domains. In addition, a *double local network* can be achieved by combining this *wavenet* architecture cascaded with a local infinite impulse response (IIR) synopsis network [YL93]. The IIR recurrent loop creates a local structure that provides a computationally efficient method of training the system, and accordingly resulting in even quicker convergence proving much less time to reach minimal error criterion. Figure 5.2 demonstrates the structure approximating any desired signal $y(t)$ by generalizing a linear combination of a set of daughter wavelets $h_{a,b}(t)$ cascaded with the local IIR recurrent networks. The approximated signal of the network $\hat{y}(t)$ can be modeled by:

$$\hat{y}(t) = \sum_{i=0}^{M} c_i z(t-i) u(t) + \sum_{j=1}^{N} d_j \; \hat{y}(t-j) v(t) \tag{5.13}$$

where

$$z(t) = \sum_{k=1}^{K} w_k h_{a_k, b_k}(t) \tag{5.14}$$

K is the number of wavelets, w_k is the *k*th weight coefficient, *M* and c_i are the number of feedforward delays and coefficients of the IIR filter, respectively, *N* and d_j are the number of feedback delays and recursive filter coefficients, respectively. The signals *u*(*t*) and *v*(*t*) are the input and co-input to the system at time *t*, respectively. Input *v*(*t*) is usually kept small for feedback stability purposes.

The gradient estimates of equations (5.5) - (5.7) can now be expressed as:

$$\frac{\partial E}{\partial w_k} = -\sum_{t=1}^{T} u(t) e(t) \sum_{i=0}^{M} c_i h(\tau - i) \tag{5.15}$$

$$\frac{\partial E}{\partial b_k} = -\sum_{t=1}^{T} u(t) e(t) \sum_{i=0}^{M} c_i \, w_k \frac{\partial h(\tau - i)}{\partial b_k} \tag{5.16}$$

$$\frac{\partial E}{\partial a_k} = -\sum_{t=1}^{T} u(t) e(t) \sum_{i=0}^{M} c_i \tau w_k \frac{\partial h(\tau - i)}{\partial b_k} = \tau \frac{\partial E}{\partial b_k} \tag{5.17}$$

$$\frac{\partial E}{\partial c_i} = -\sum_{t=1}^{T} u(t)e(t)z(t-i) \tag{5.18}$$

$$\frac{\partial E}{\partial d_j} = -\sum_{t=1}^{T} v(t)e(t)\ \hat{y}(t-j) \tag{5.19}$$

Again, the incremental changes of each parameter are simply the negative of their gradients,

$$\Delta w = -\frac{\partial E}{\partial w},\quad \Delta b = -\frac{\partial E}{\partial b},\quad \Delta a = -\frac{\partial E}{\partial a},\quad \Delta c = -\frac{\partial E}{\partial c},\quad \text{and}\quad \Delta d = -\frac{\partial E}{\partial d} \tag{5.20}$$

thus each coefficient vector $\underline{w}$, $\underline{a}$, $\underline{b}$, $\underline{c}$, and $\underline{d}$ of the network is updated as shown in equations (5.10) - (5.12), and in accordance with the rule as

$$\underline{c}(n+1) = \underline{c}(n) + \mu_c \Delta \underline{c} \tag{5.21}$$

$$\underline{d}(n+1) = \underline{d}(n) + \mu_d \Delta \underline{d} \tag{5.22}$$

where the subscripted μ-values are fixed learning rate parameters.

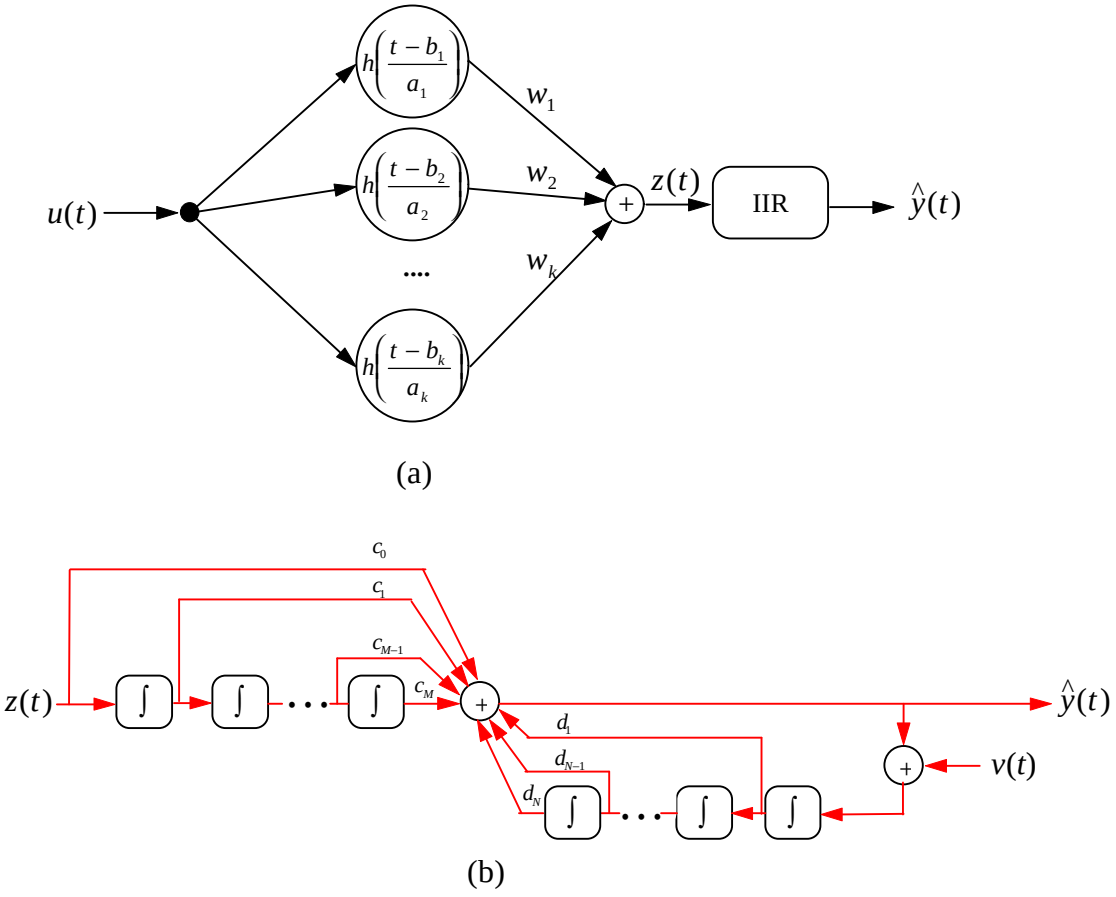


Figure 5.2 IIR Adaptive Wavelet Network Structure: (a) Local Network (b) IIR Model

5.3 Stability of the IIR Adaptive Network

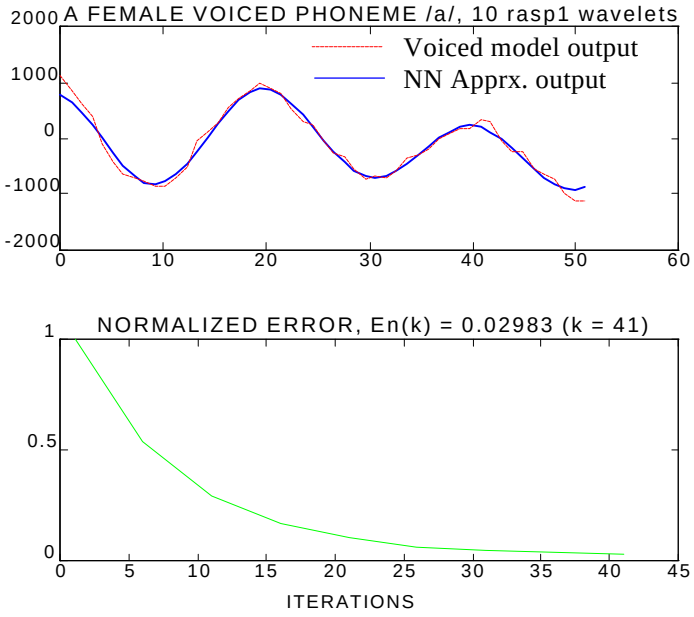
A common problem of IIR adaptive networks is the problem of guaranteeing stability and convergence. In particular, IIR adaptive networks are prone to instability as a consequence of unbounded growth of the adaptation coefficients. Furthermore, recursively adapting coefficients creates movement in the location of poles from the origin, causing the network to become unstable, even if the adaptation is stable. Finally, the convergence of the steepest descent gradient algorithm that is applied to minimize the error sometimes becomes stuck in a local minimum. In spite of these drawbacks the potential of IIR adaptive networks is such that interest in such structures remains high. In particular, IIR adaptive networks may be most useful in situations where the system has lightly damped modes (sinusoidal or near sinusoidal components) [Cla93].

The problem of the potential adaptive instability can be solved by avoiding any update which would lead to an unstable system or by successively reducing the learning rate factor. Systematically, the authors in [DCM92] proposed a new method of automatically creating adaptive learning rates to achieve the optimal rate of convergence.

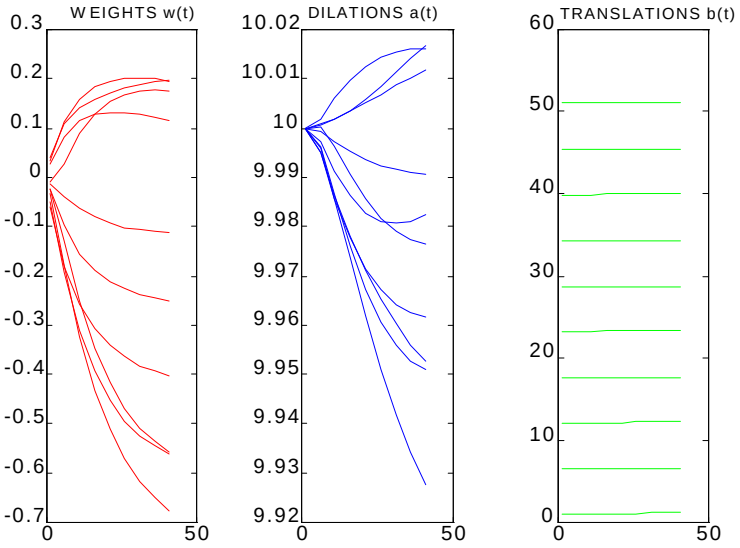
5.4 IIR Learning Comparisons

In this section, the network performance between the *wavenets* with IIR recurrent network and the *wavenets* without the IIR recurrent network is compared. A case of a female voiced phoneme /a/ (as in “had”) is being considered as an example of the comparison to the learning processes between the *wavenet* network with the IIR block structure and without the IIR block structure. Both structures have the same sets of initial parameters as shown in Table 5.2 and 5.3. Learning rate parameters for weights, dilations, translations, IIR feedforward coefficients, and feedback coefficients are fixed at 0.01, 0.05, 0.05, 0.02, and 0.02, respectively. A batch-mode learning of 50 sample data is adapted until the desired error of 0.03 is reached. Ten RASP1 mother wavelet filters are employed in both networks. Note there are no particular reasons governing the choice of these parameters. These parameters were obtained from the past experience of many simulations that were found to

be efficient. Table 5.2 and 5.3 provide the final parameter update results to each network and Figure 5.3 and 5.4 illustrate the simulation results. Note: All plots are sampled at every 5 iterations to provide less dense and smaller memory of data.



(a)

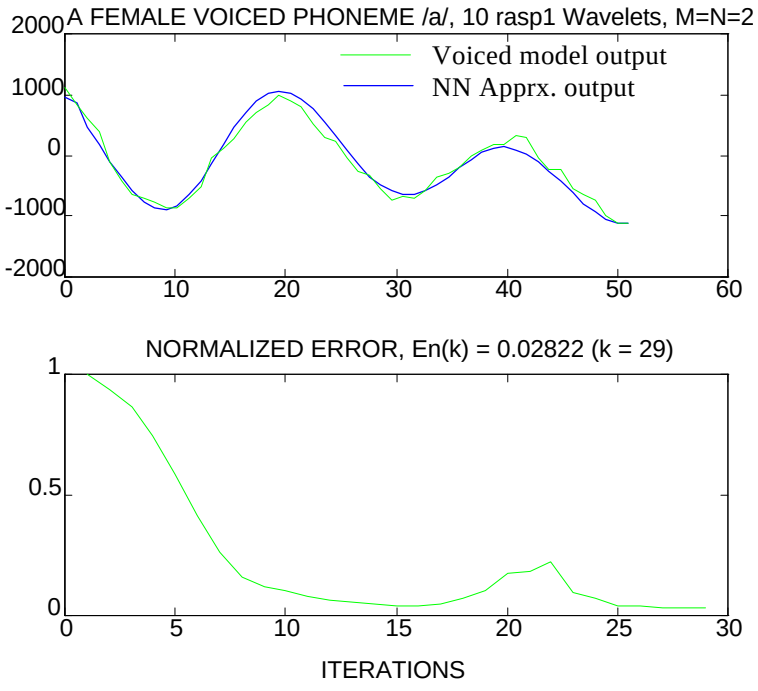


(b)

Figure 5.3 *Wavenet* Without IIR Blocks:
(a) Learning Process (b) NN Parameter Time Histories

Table 5.2 Weight, Dilation, and Translation Parameters Without IIR Blocks

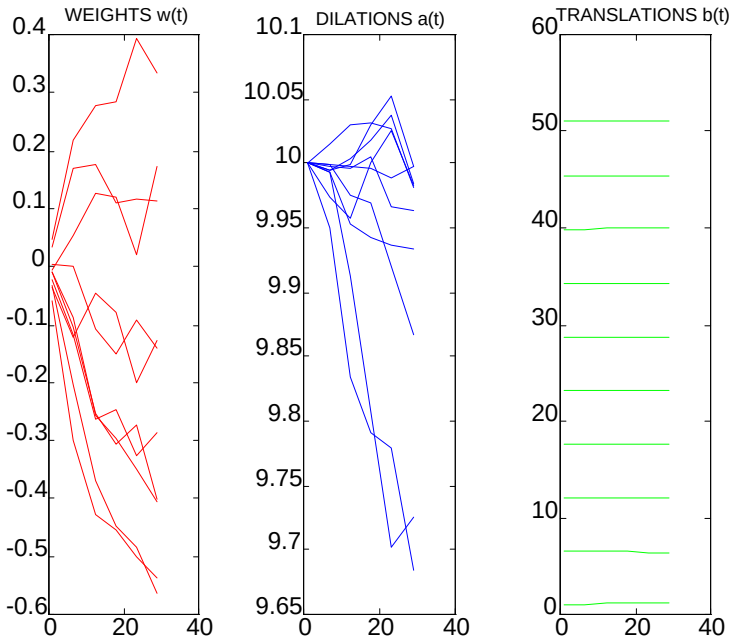
Initial			Final		
w	*a*	*b*	*w*	*a*	*b*
0	10.0000	1.0000	-0.2541	10.0160	1.1240
0	10.0000	6.5556	-0.6818	9.9264	6.4449
0	10.0000	12.1111	0.2002	9.9519	12.1928
0	10.0000	17.6667	0.1946	9.9614	17.5843
0	10.0000	23.2222	-0.5632	9.9825	23.2888
0	10.0000	28.7778	-0.4034	10.0173	28.6582
0	10.0000	34.3333	0.1747	9.9763	34.3482
0	10.0000	39.8889	-0.1116	10.0119	39.9264
0	10.0000	45.4444	-0.5615	9.9507	45.4442
0	10.0000	51.0000	0.1138	9.9907	51.0732



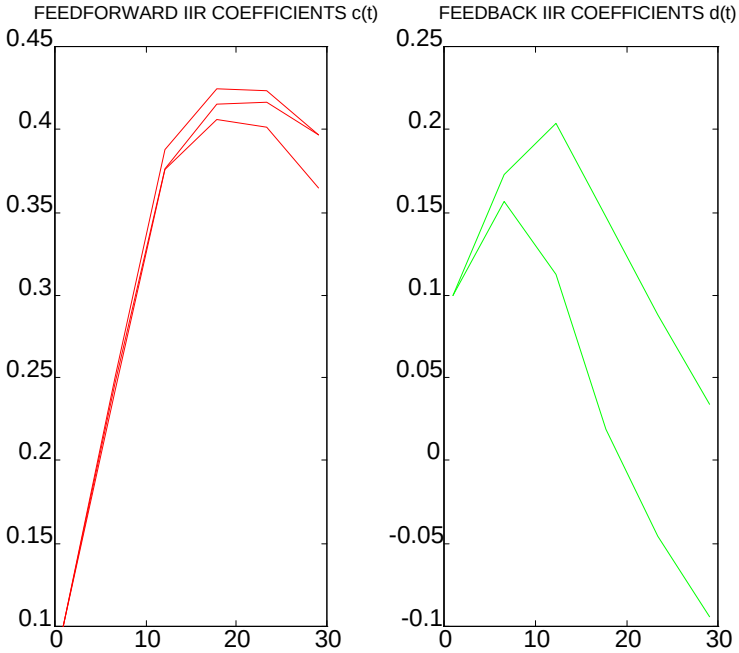
(a)

Figure 5.4 *Wavenet* With IIR Blocks:

(a) Learning Process (b) NN Parameter Updates (c) IIR Coefficients



(b)



(c)

Figure 5.4 *Wavenet* With IIR Blocks:

(a) Learning Process (b) NN Parameter Updates (c) IIR Coefficients

Table 5.3 Weights, Dilations, Translations and IIR Coefficients

Initial					Final				
$\underline{w}$	$\underline{a}$	$\underline{b}$	$\underline{c}$	$\underline{d}$	$\underline{w}$	$\underline{a}$	$\underline{b}$	$\underline{c}$	$\underline{d}$
0	10.0000	1.0000	0.1	0.1	-0.5695	9.7086	1.2366	0.3652	0.0272
0	10.0000	6.5556	0.1	0.1	-0.4040	9.9568	6.3951	0.3998	-0.0855
0	10.0000	12.1111	0.1	0.1	0.3462	9.6807	12.1260	0.4003	
0	10.0000	17.6667			-0.1370	9.9983	17.7108		
0	10.0000	23.2222			-0.5481	9.9865	23.1781		
0	10.0000	28.7778			-0.1478	9.9727	28.7076		
0	10.0000	34.3333			0.1633	9.8536	34.3356		
0	10.0000	39.8889			-0.2986	10.0050	39.9949		
0	10.0000	45.4444			-0.4254	9.9810	45.4225		
0	10.0000	51.0000			0.1150	9.9301	51.0440		

Table 5.4 Error Comparison for the *Wavenet* Network With 10 RASP1 Wavelets

Number of Iterations	Desired Normalized Error						
	0.8	0.5	0.3	0.1	0.05	0.04	0.03
With IIR Blocks	4	6	7	11	14	27	29
Without IIR Blocks	4	6	10	20	29	34	41

Figures 5.3 and 5.4 show that the *wavenet* network cascaded with the IIR recurrent network structure can achieve faster convergence (at 29th iteration) to reach a desired normalized error of 0.03 than the network without the IIR recurrent blocks reaching target error at 41st iteration. The normalized error to be evaluated as a performance measure is defined as $E_n(t) = (\underline{e}^T \underline{e})/(\underline{y}^T \underline{y})$, where T is a transpose of the vector. It has also been shown through several of my previous simulation studies that the *wavenet* with the IIR blocks can reach the desired error faster than the network without the IIR structure when the training data is large and/or the size of wavelet filters (neurons) is small. From the Tables 5.2 and 5.3, during the adaptation process the weight and the dilation parameter updates are smoother and less varied to the network without the IIR blocks than the one with the IIR blocks, but spreads out from the initial positions at about the same deviation. Table 5.4

confirms the convergent performance as the number of iterations of the *wavenets* network with the IIR structure becomes smaller to reach a finer normalized error than the *wavenets* structure without the IIR network.

5.5 MLP Learning Comparisons

In this section, the learning performance of the conventional multilayer perceptron (MLP) networks is provided for a comparison to the proposed *wavenets* structures. MLP is a global network which updates its network weights with each training sequence at any given point in the input space. Usually, backpropagation (BPP) learning algorithm is employed in the training method. However, MLP with BPP is well known for its long training times, slow convergence, and its susceptibility to a local minima. Simulations of MLP learning with BPP algorithm and the *wavenets* algorithm will be investigated and compared for their learning abilities. Figures 5.5 and 5.6 show the results of the two networks approximating the piecewise defined function [ZB92]:

$$f(x) = \begin{cases} -2.186x - 12.864 & \text{if } -10 \le x < -2 \\ 4.246x & \text{if } -2 \le x < 0 \\ 10e^{-0.05x-0.5} \cdot \sin\big((0.03x + 0.7)x\big) & \text{if } 0 \le x \le 10 \end{cases} \tag{5.23}$$

BPP with adaptive learning rate was used to obtain the best results for the MLP. Three layer feedforward neural network, i.e., two hidden layer and the output layer was employed. The activation functions employed were the hyperbolic tangent, initial learning rate was 0.01, the momentum constant was 0.75. The initial weights of adaptation were set at random. The performance of the BPP is described in Figure 5.5 where the sum-squared error (SSE) at 3000th iteration was 0.7872. Figure 5.6 illustrates the learning results of the *wavenets* network without the double local IIR network. Forty Morlet wavelets were used in the hidden neurons. Fixed learning rates of 0.01, 0.001 and 0.001 were used for updating weights, scales, and shifts, respectively. SSE of 0.06332 was reached for only 100 iterations.

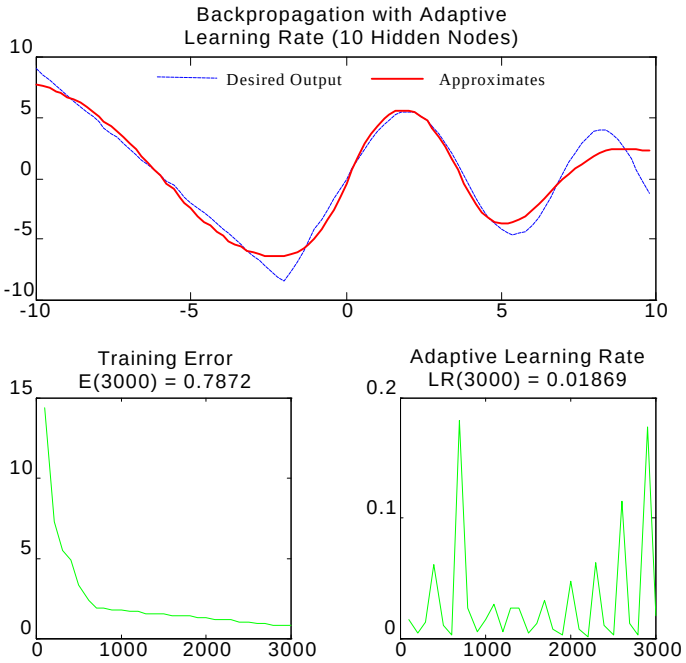


Figure 5.5 MLP with Backpropagation Adaptive Learning Rates

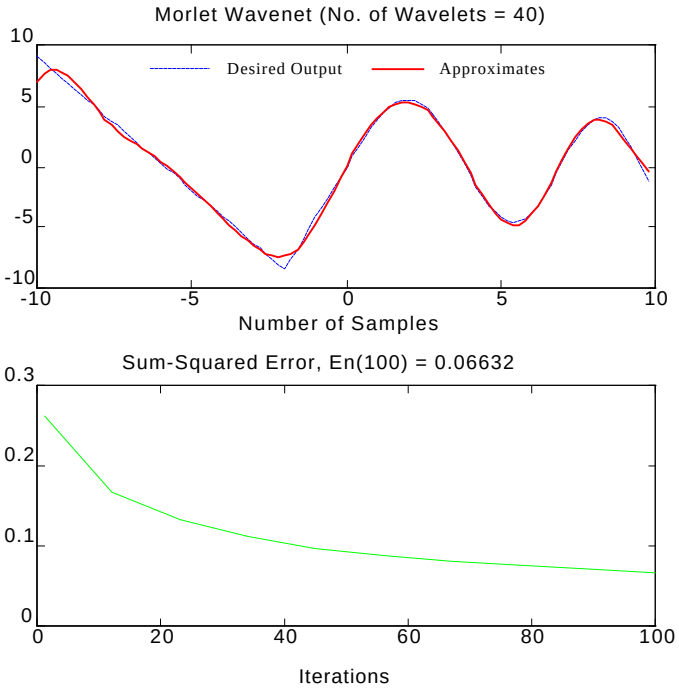


Figure 5.6 *Wavenet* Network Without the IIR Local Network

5.6 Simulation Configuration and Results

5.6.1 Approximation of an Unknown Function

In this section, it will be shown through experimental simulations how various types of mother wavelet basis functions perform their learning ability. Using the same set of data from the previous female voiced phoneme /a/ extracted from the TIMIT database from a file "fmem0sa1.wav" at bit allocations: 16300-16350, the *wavenet* network with different size and type of mother wavelets is employed to approximate the voiced data. IIR block structure with feedforward coefficients $m = 2$ and feedback coefficients $n = 2$ is also implemented. As we know from Chapter 4, wavelets are orthogonal basis functions, thus they can be added or removed one at a time without having to update the parameters of the previously placed basis functions resulting to an incrementally improvement at low computational cost. Moreover, wavelets are local basis functions that provide less interfering than global ones, leading to a noncomplex dependency in the NN parameters. The following sections will confirm this idea by providing several observations derived from the results of the MATLAB® simulations. Assuming the training data are stationary and sufficiently rich, good performance can usually be achieved with a small learning rate. Thus, all learning rate parameters for weights, dilations, translations, IIR feedforward coefficients, and feedback coefficients are fixed at 0.01, 0.05, 0.05, 0.02, and 0.02, respectively. All initial weights w_k and dilations a_k are set to 0 and 10, respectively. Note that if the dilation parameters are set too wide, they can cause several overlapping partitions and thus cannot be realized. Setting a_k too narrow may result in longer convergence. Initial translation parameters b_k are spaced equally apart throughout the training data to provide non-overlapping partitions throughout the neighboring intervals. Finally, the initial IIR coefficents c and d should be set so that the system has poles inside the unit circle, thus both are set to 0.1. The number of coefficents for each feedforward and feedback m and n are both set to 2 as well. The learning epoch will terminate when the desired normalized error of 0.03 is reached. The following simulations will describe the results of the *wavenet* network performance employing Morlet, Mexican Hat, RASP1, POLYWOG5, Shannon, and SLOG2 super-mother wavelets.

5.6.1.1 Morlet Mother Wavelet Basis Functions

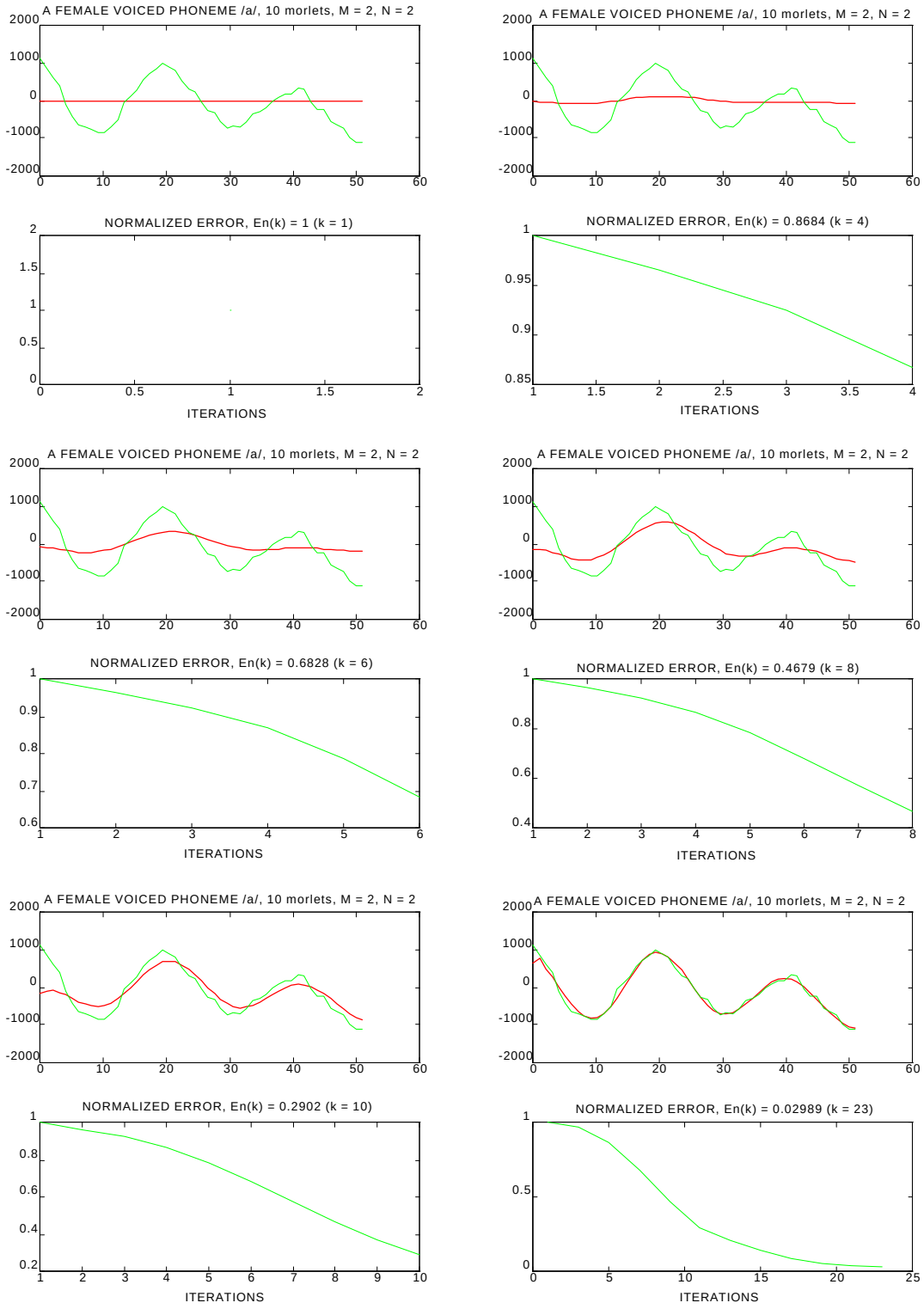


Figure 5.7a *Wavenet* Simulations with 10 Morlet Wavelets:

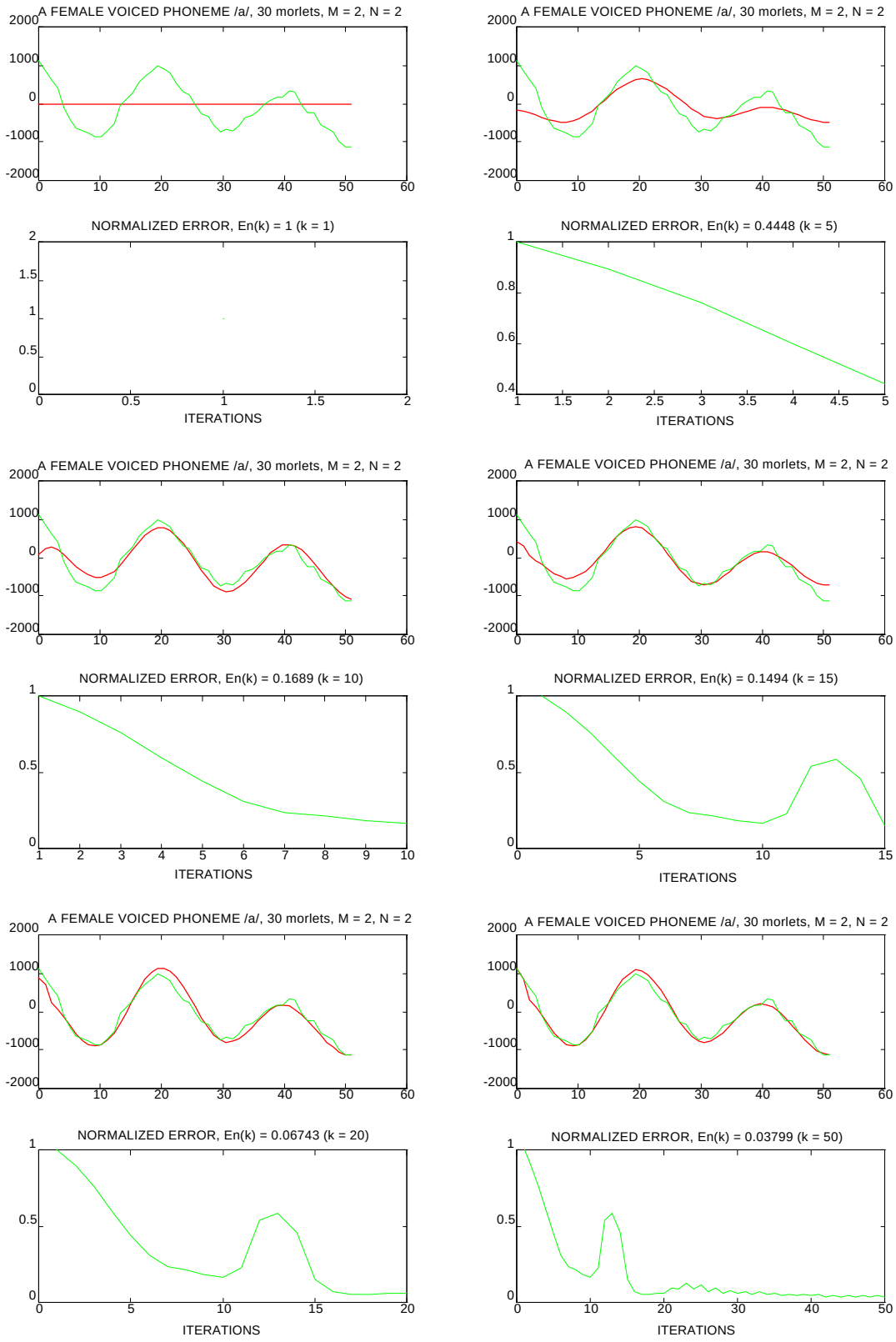


Figure 5.7b *Wavenet* Simulations with 30 Morlet Wavelets:

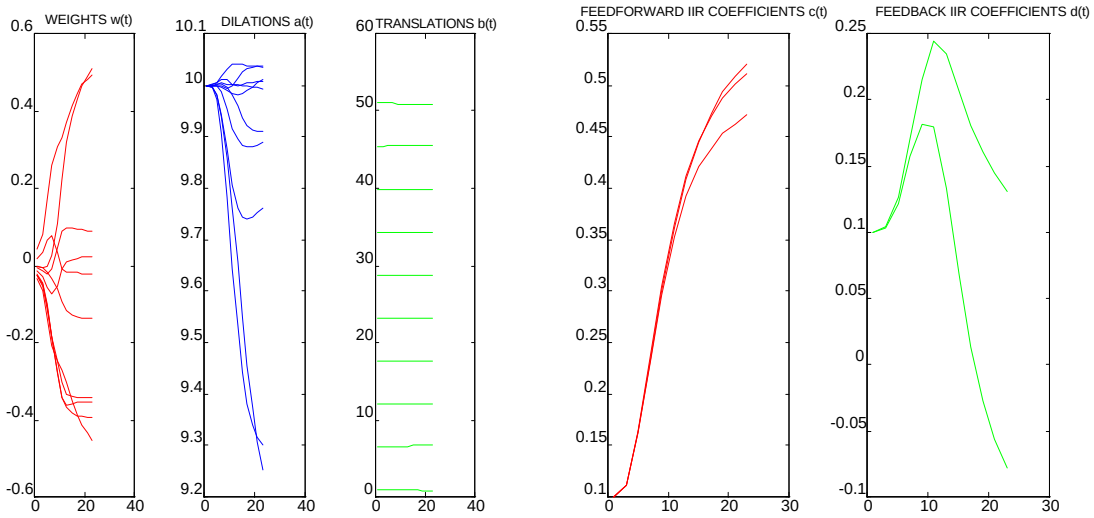


Figure 5.8a *Wavenet* Parameter Updates with 10 Morlet Wavelets

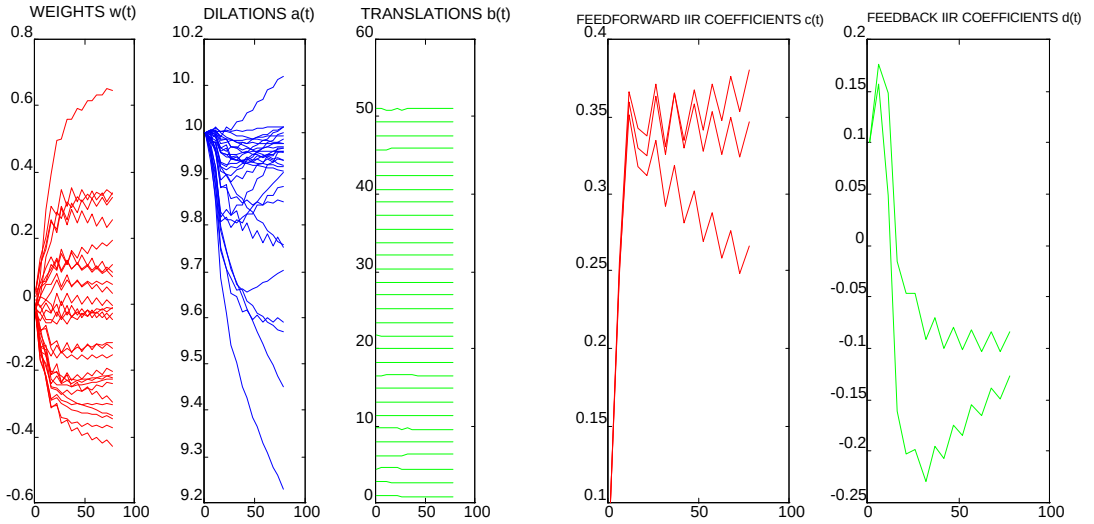


Figure 5.8b *Wavenet* Parameter Updates with 30 Morlet Wavelets

Figures 5.7a and 5.7b capture the learning performance of the *wavenet* network using 10 and 30 Morlet wavelets, respectively. We can conclude that the *wavenet* network composed of more wavelets can reach initial convergence with reference to the number of iterations very rapidly. However, to reach the desired error goal 0.03, networks with a large number of wavelets cannot converge easily and the error performance starts to oscillate. This behavior may be caused by the rate of learning stepsize not being small enough, causing the

iteration process to bounce between the two opposite sides of a valley rather than following the natural gradient contour (as shown in Figure 5.8b). Figure 5.9 and Table 5.5 provide information on the Morlet *wavenet* characteristic. As we can see, when the number of wavelets K is small, for example, for K = 3, it takes 28 iterations to reach error of 0.8 while it takes 3 iterations for K = 35, but when the error of 0.03 is the target, K = 8 takes 23 iterations while K = 30 takes 432 iterations. Large K is also undesirable to the expense of more coefficients to be updated. Small K can also takes a large amount of time to compute as for K = 3 takes more than 2000 iterations to reach error of 0.04. In conclusion, the number of Morlet wavelets between K = 8 to K = 16 is sufficient to approximate the unknown voice model.

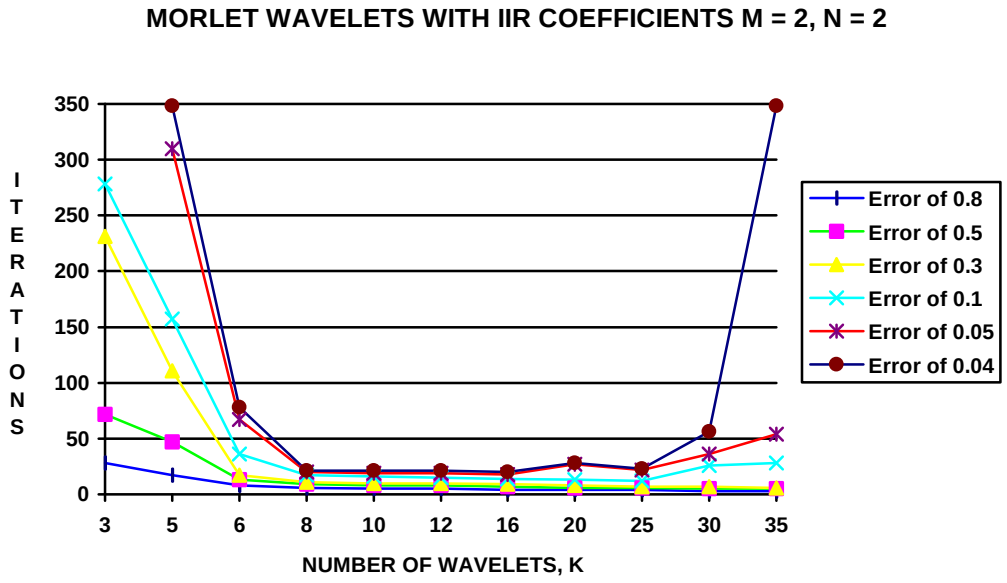
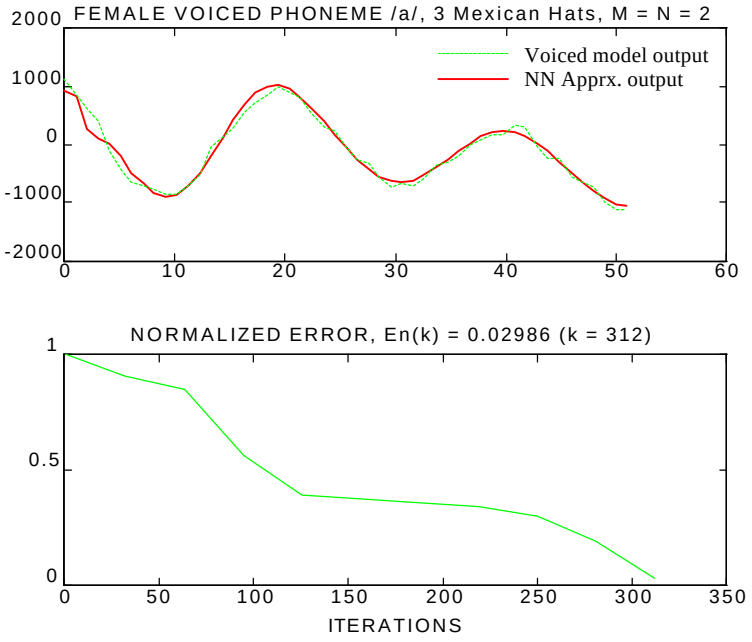


Figure 5.9 Iterations vs. Number of Morlet Wavelets Per Normalized Errors

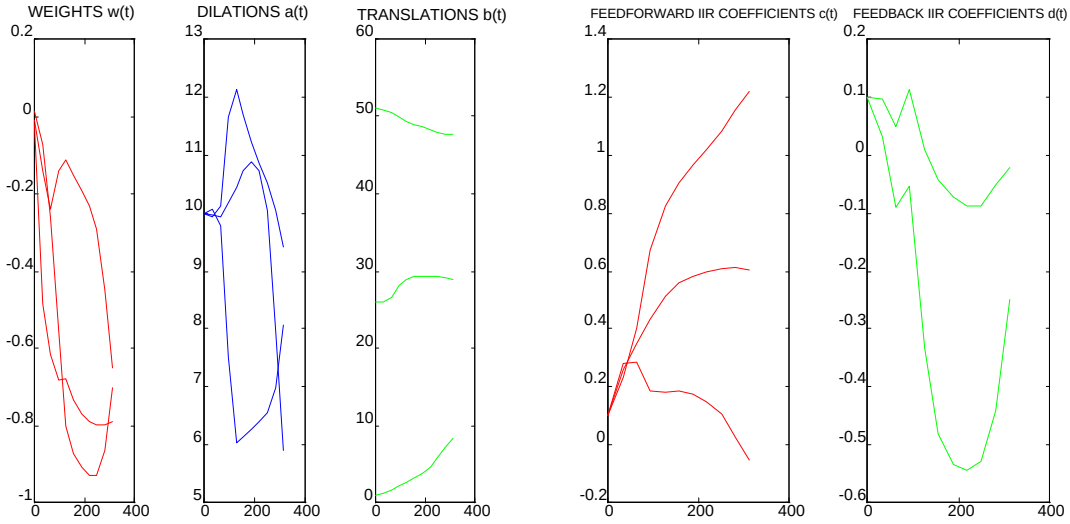
Table 5.5 Number of Iterations vs. Number of Morlet Wavelets Employed

Number of Iterations	***Number of Wavelets, K***										
	3	**5**	**6**	**8**	**10**	**12**	**16**	**20**	**25**	**30**	**35**
Error of 0.8	28	17	8	6	5	5	4	4	4	3	3
Error of 0.5	72	47	13	9	8	8	7	6	6	5	5
Error of 0.3	231	111	17	11	10	10	9	8	7	7	6
Error of 0.1	278	157	36	17	16	15	14	13	12	26	28
Error of 0.05	853	310	67	20	19	19	18	27	22	36	54
Error of 0.04	2000+	348	78	21	21	21	20	28	23	56	348
Error of 0.03		671	94	23	23	24	23	29	27	78	432

5.6.1.2 Mexican Hat Mother Wavelet Basis Functions



(a)



(b) (c)

Figure 5.10 *Wavenet* Simulations Using 3 Mexican Hat Wavelets

(a) Simulation Results at 312th Iteration

(b) *Wavenet* Parameter Updates

(c) IIR Block Parameter Updates

Figures 5.10a - 5.10c describe the wavelet network composed of 3 Mexican Hat wavelets, the network parameters adaptation, and the IIR coefficients adaptation, respectively. Normalized error of 0.02986 is reached at 312th iteration (comparing this to the previous Morlet wavelets at 2000+ iteration). Here we can see some adjustments in the adaptation parameter b_k . These changes is caused by the small amount of wavelets that in turns providing narrowed receptive fields to partition the neighboring intervals. Table 5.6 demonstrates the values of error vs. the number of wavelets vs. the number of iterations. For K = 16 and 20, respectively, the learning takes 17 and 16 steps to reach the error of 0.3, but then the errors of 0.1 are never reached, because the system becomes diverge shooting up the error to infinity. From Figure 5.11, we can choose a K that is sufficient to approximate the unknown data by searching for the steady valley of the graph (i.e., K = 8 and 10).

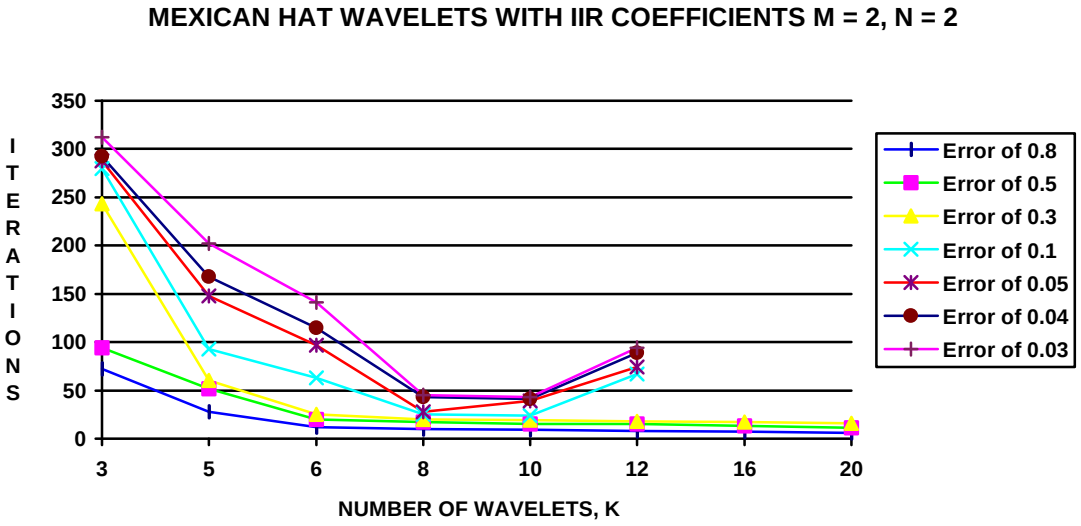


Figure 5.11 Iterations vs. Number of Mexican Hat Wavelets Per Normalized Errors

Table 5.6 Number of Iterations vs. Number of Mexican Hat Wavelets Employed

Number of Iterations	***Number of Wavelets, K***							
	3	**5**	**6**	**8**	**10**	**12**	**16**	**20**
Error of 0.8	72	28	12	10	9	8	7	6
Error of 0.5	94	52	20	17	15	15	13	11
Error of 0.3	243	60	25	20	19	18	17	16
Error of 0.1	280	93	63	25	24	67	∞	∞
Error of 0.05	288	148	97	28	39	74		
Error of 0.04	292	168	115	43	41	89		
Error of 0.03	312	202	141	45	43	94		

5.6.1.3 RASP1 Mother Wavelet Basis Functions

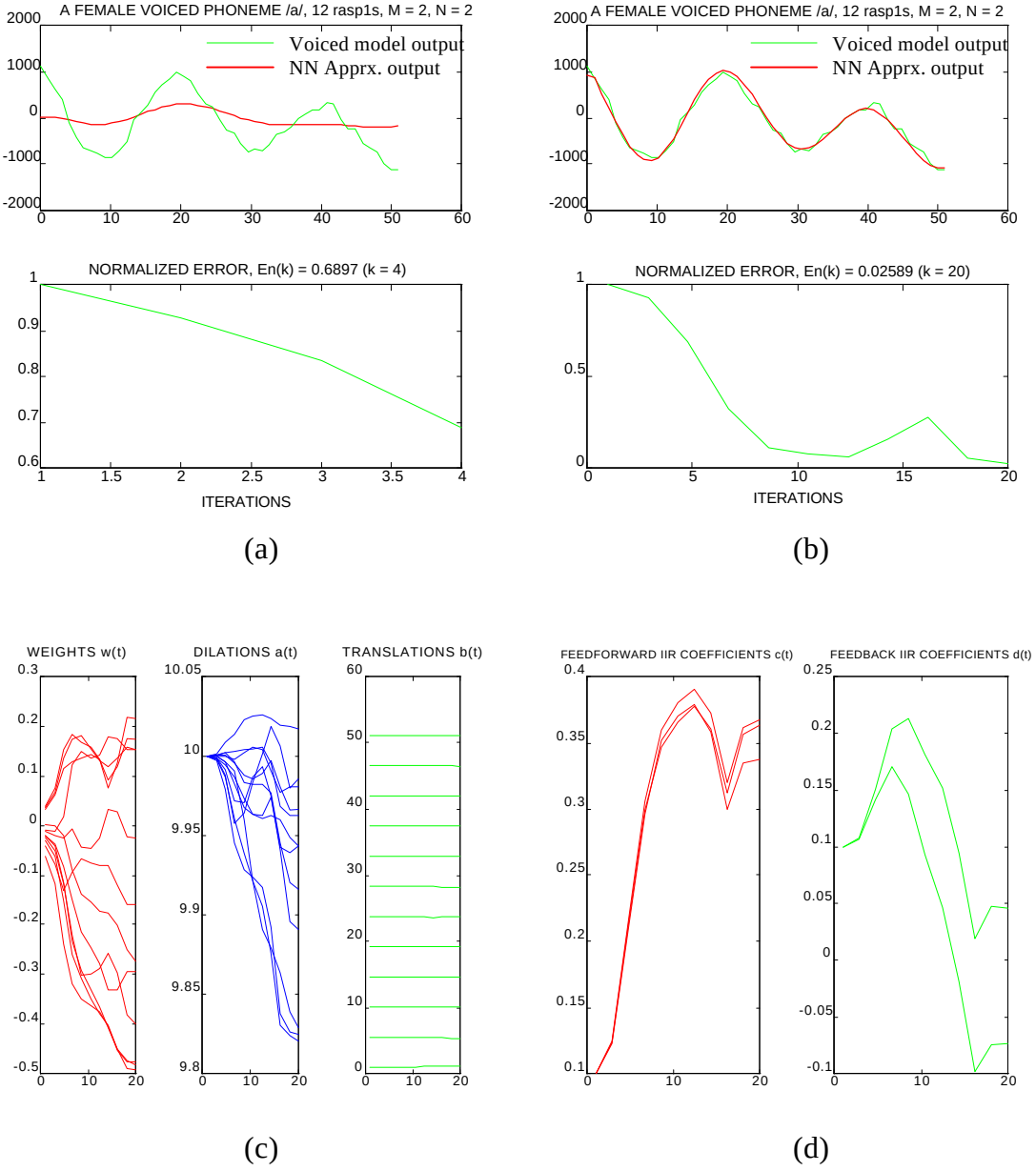


Figure 5.12 *Wavenet* Simulations Using 12 RASP1 Wavelets

(a) Simulation Results at 4th Iteration

(b) Simulation Results at 20th Iteration

(c) *Wavenet* Parameter Updates

(d) IIR Block Parameter Updates

Figure 5.12 illustrates the RASP1 performance. Figure 5.13 and Table 5.7 provide the graphical and numerical values of the experiment, respectively. The results show that *wavenet* employing 12 RASP1 wavelets can reach the error goal of 0.3 in the fastest time among all mother wavelets experimented at 20 iterations. However, again when we oversize the number of wavelets to K = 20 and 25, the normalized error starts to oscillate steadily resulting in prolong time consumption to reach the fine error target. From Figure 5.13, the best number of wavelets to employ in the *wavenet* network for approximation of the unknown voice model turns out to be 12.

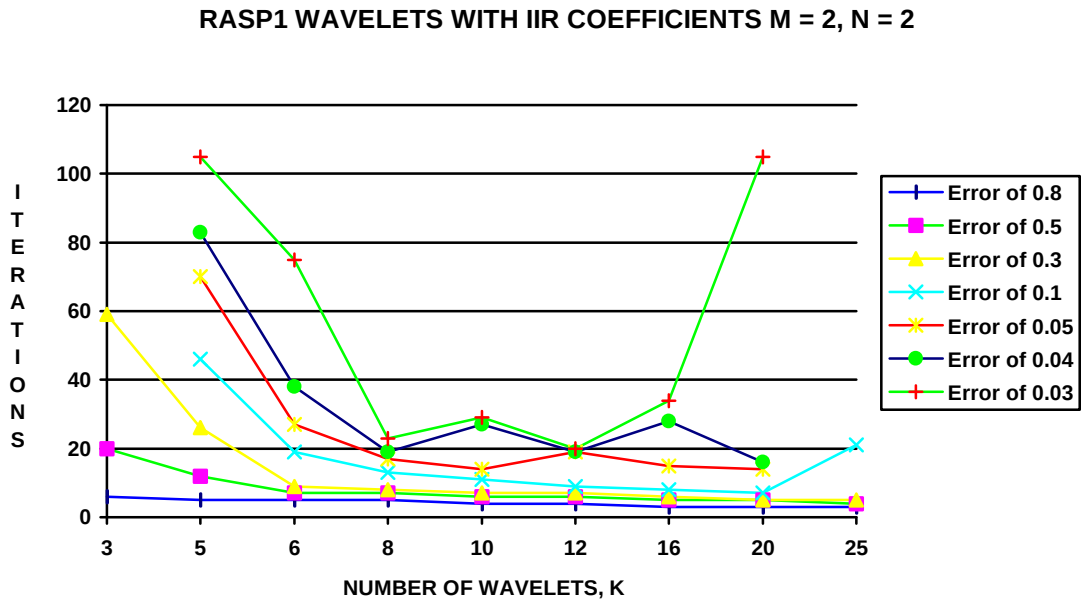


Figure 5.13 Iterations vs. Number of RASP1 Wavelets Per Normalized Errors

Table 5.7 Number of Iterations vs. Number of RASP1 Wavelets Employed

Number of Iterations	***Number of Wavelets, K***								
	3	**5**	**6**	**8**	**10**	**12**	**16**	**20**	**25**
Error of 0.8	6	5	5	5	4	4	3	3	3
Error of 0.5	20	12	7	7	6	6	5	5	4
Error of 0.3	59	26	9	8	7	7	6	5	5
Error of 0.1	715+	46	19	13	11	9	8	7	21
Error of 0.05		70	27	17	14	19	15	14	100+
Error of 0.04		83	38	19	27	19	28	16	(Oscillate)
Error of 0.03		105	75	23	29	20	34	100+	

5.6.1.4 POLYWOG5 Mother Wavelet Basis Functions

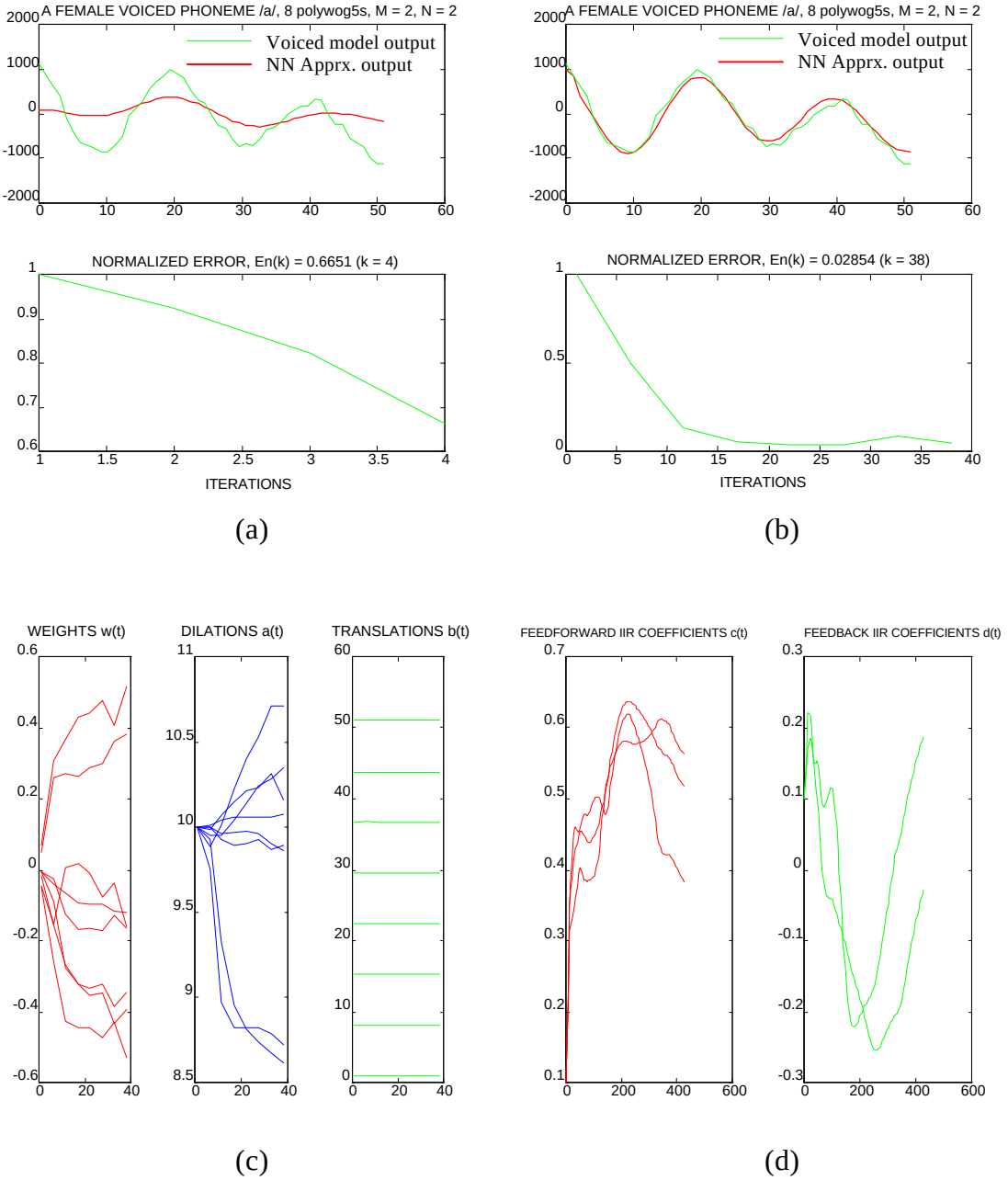


Figure 5.14 *Wavenet* Simulations Using 8 POLYWOG5 Wavelets

(a) Simulation Results at 4th Iteration

(b) Simulation Results at 38th Iteration

(c) *Wavenet* Parameter Updates

(d) IIR Block Parameter Updates

Figure 5.14 provides the simulation results for a network composed of 8 POLYWOG5 wavelets. The weight adaptations seem to spread out equally between -0.4 to 0.4. Almost no adjustment at all is shown for the shift parameters. This means either the dilation values are large enough or the number of wavelets are large. Again, when *K* is too big, the error starts to oscillate that is caused by the large learning rates. The error goal of 0.03 consumes longer time to converge. Finally, from Figure 5.15 and Table 5.7, the number of POLYWOG5 wavelets equal to 8 is best chosen in this experiment.

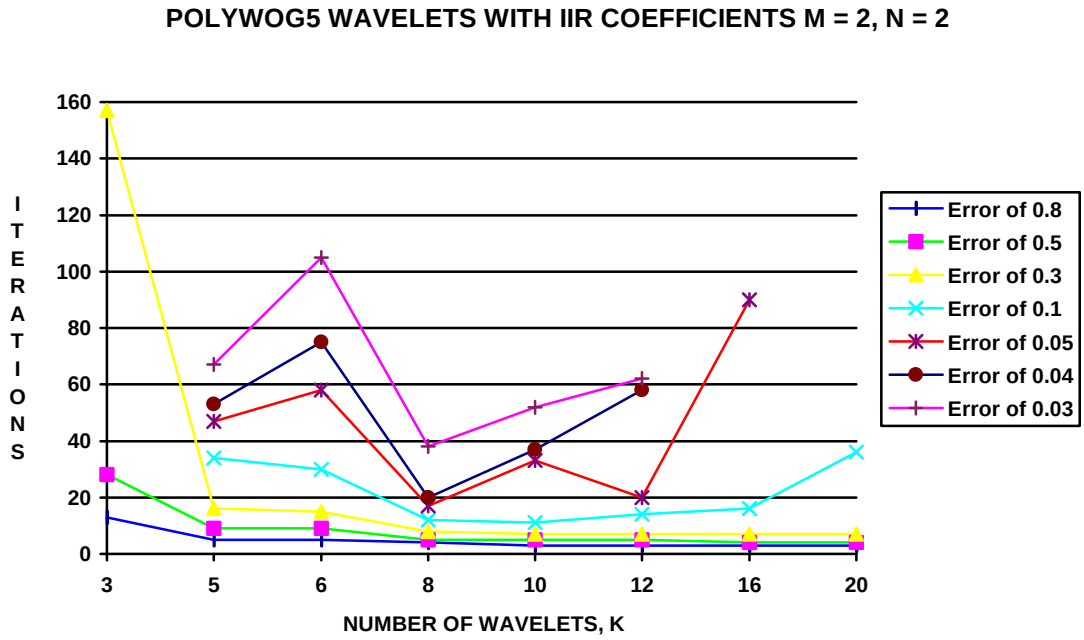


Figure 5.15 Iterations vs. Number of POLYWOG5 Wavelets Per Normalized Errors

Table 5.7 Number of Iterations vs. Number of POLYWOG5 Wavelets Employed

Number of Iterations	***Number of Wavelets, K***							
	3	**5**	**6**	**8**	**10**	**12**	**16**	**20**
Error of 0.8	13	5	5	4	3	3	3	3
Error of 0.5	28	9	9	5	5	5	4	4
Error of 0.3	157	16	15	8	7	7	7	7
Error of 0.1	1600+	34	30	12	11	14	16	36
Error of 0.05		47	58	17	33	20	90+	(Oscillate)
Error of 0.04		53	75	20	37	58	(Oscillate)	
Error of 0.03		67	105	38	52	62		

5.6.1.5 Shannon Mother Wavelet Basis Functions

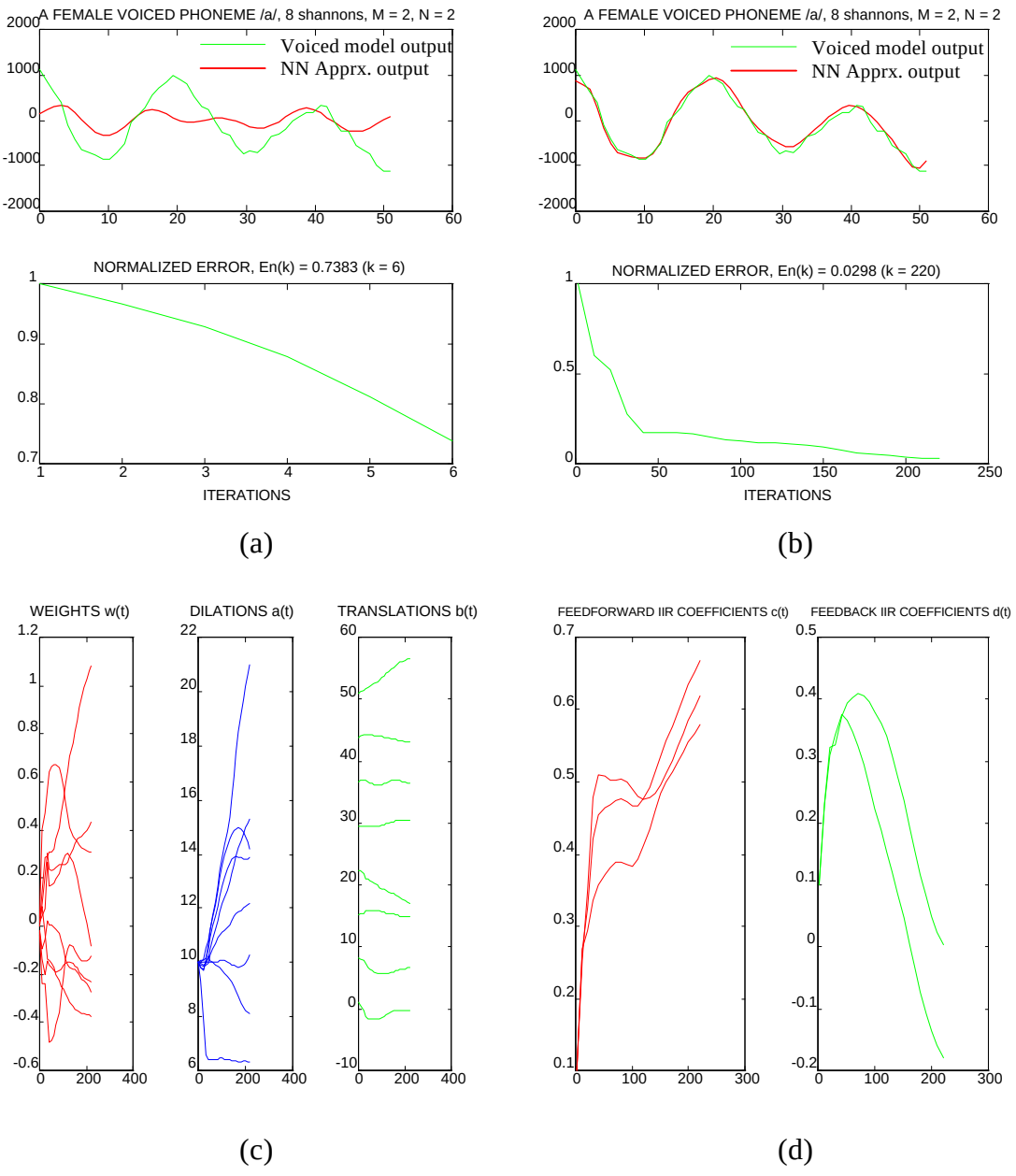


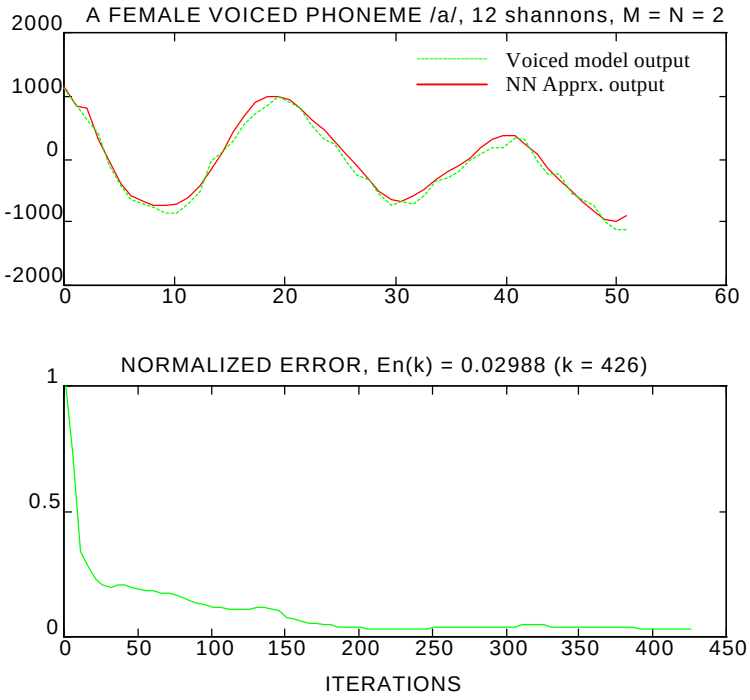
Figure 5.16 *Wavenet* Simulations Using 8 Shannon Wavelets

(a) Simulation Results at 6th Iteration

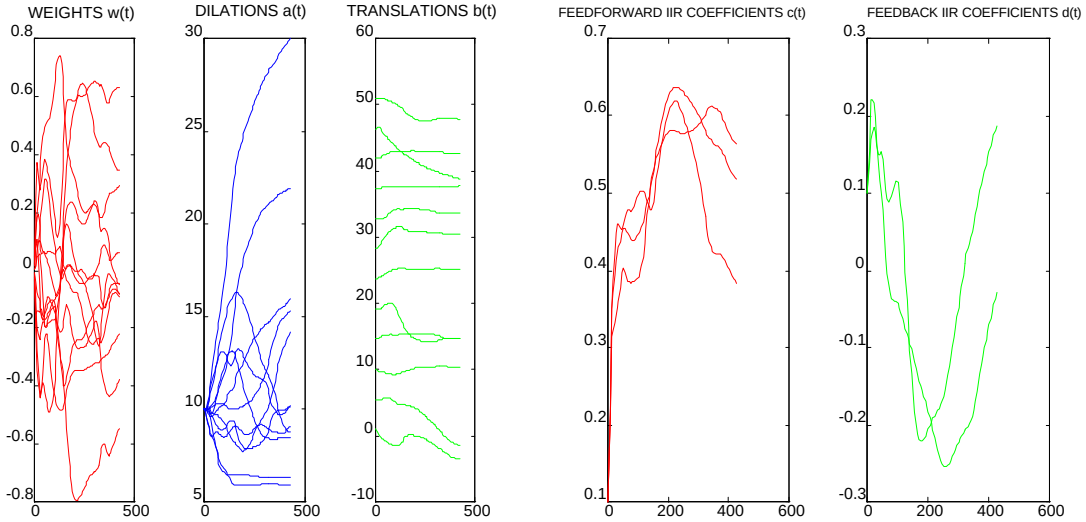
(b) Simulation Results at 220th Iteration

(c) *Wavenet* Parameter Updates

(d) IIR Block Parameter Updates



(a)



(b) (c)

Figure 5.17 *Wavenet* Simulations Using 12 Shannon Wavelets

(a) Simulation Results at 426th Iteration

(b) *Wavenet* Parameter Updates

(c) IIR Block Parameter Updates

Figures 5.16 and 5.17 provide simulation results of the *wavenet* performance using 8 and 12 Shannon wavelets, respectively. A Shannon mother wavelet as a basis function to the *wavenet* networks has a unique characteristic compared to the others in the sense that their shift parameters change as time varies, even if the number of super-mother wavelets that covers the univariate receptive fields is increased. This unique learning behavior is due to the discontinuity in the Shannon wavelet's frequency domain that corresponds to several oscillations in the time domain. Figure 5.18 and Table 5.8 correspond to numerical values of the learning process. To reach a finer grid (smaller error target), the learning also takes longer time.

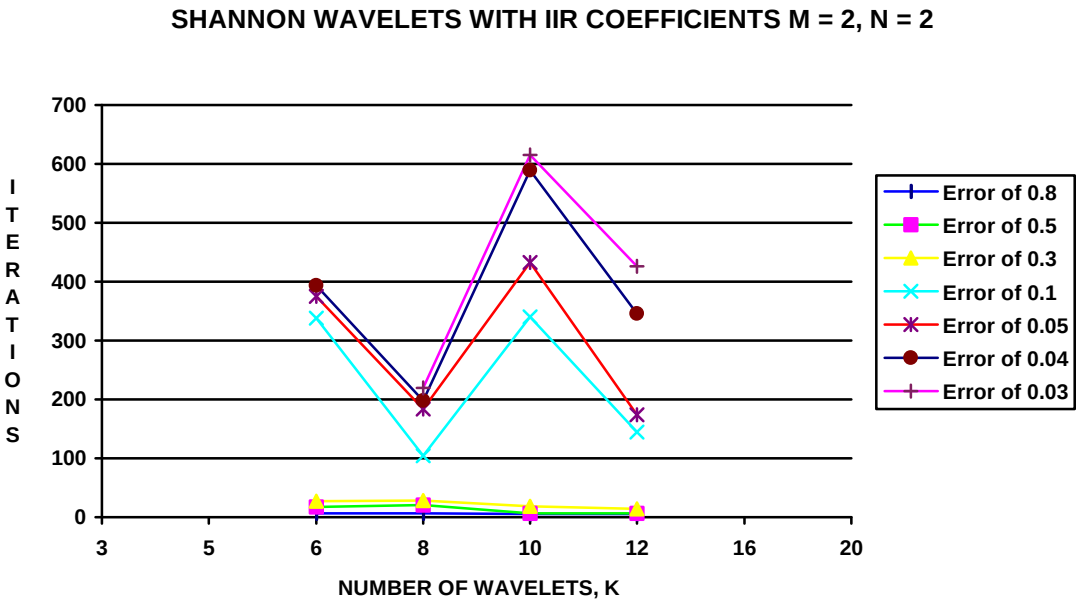


Figure 5.18 Iterations vs. Number of Shannon Wavelets Per Normalized Errors

Table 5.8 Number of Iterations vs. Number of Shannon Wavelets Employed

Number of Iterations	***Number of Wavelets, K***			
	6	**8**	**10**	**12**
Error of 0.8	7	6	5	5
Error of 0.5	17	21	7	7
Error of 0.3	27	28	18	14
Error of 0.1	338	104	340	145
Error of 0.05	375	184	433	174
Error of 0.04	393	198	589	346
Error of 0.03	1675+	220	615	426

5.6.1.6 SLOG2 Mother Wavelet Basis Functions

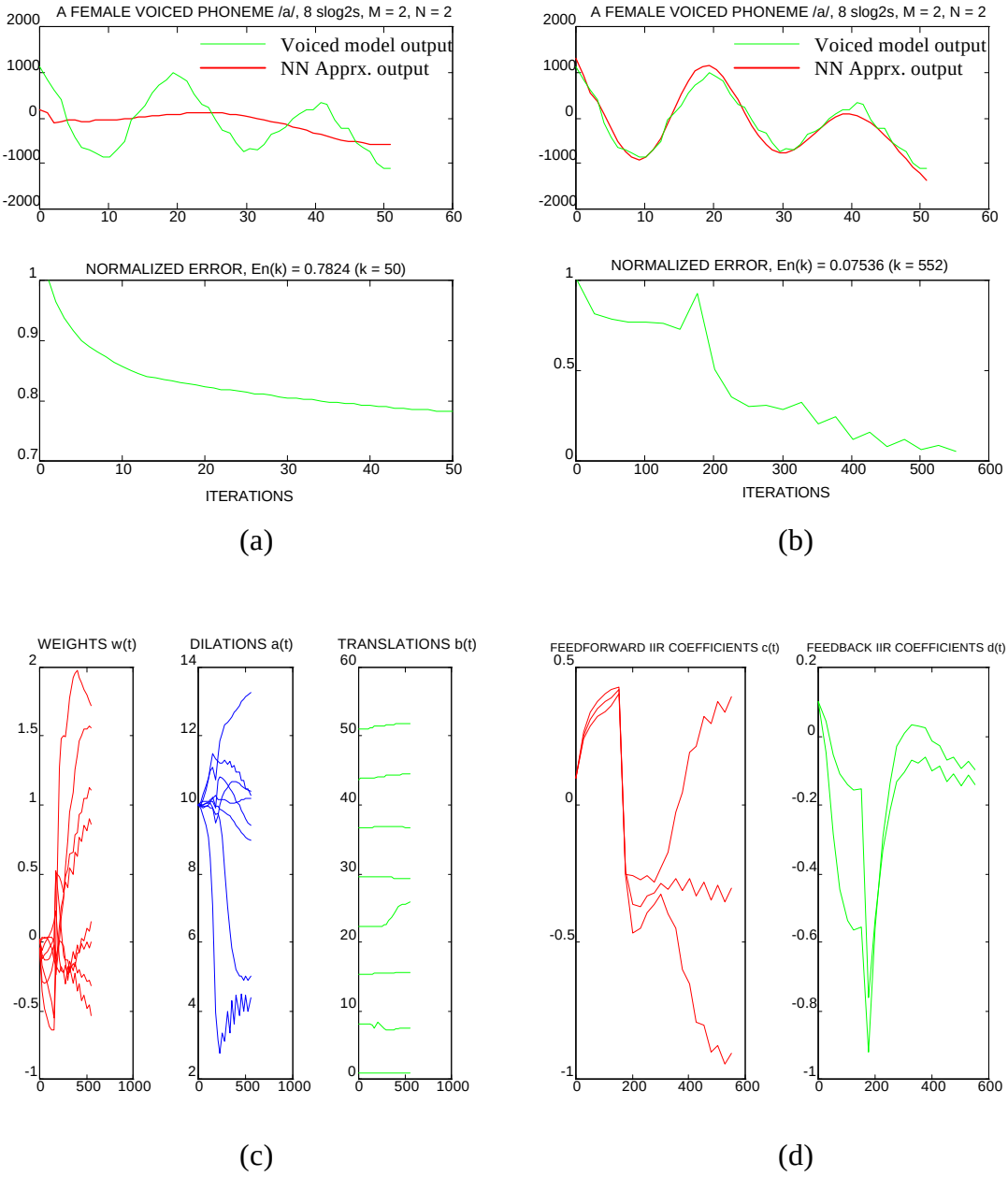


Figure 5.19 *Wavenet* Simulations Using 8 SLOG2 Wavelets

(a) Simulation Results at 50th Iteration

(b) Simulation Results at 552st Iteration

(c) *Wavenet* Parameter Updates

(d) IIR Block Parameter Updates

Figure 5.19 shows *wavenet* simulations of 8 SLOG2 wavelets. Figure 5.20 and Table 5.9 provide numerical error values. This mother wavelet may not be a good choice for approximation in a neural network realization; the error diverges when the network adapts for several iterations. This could be the stepsize problem itself or the sensitivity of the superposed logistic.

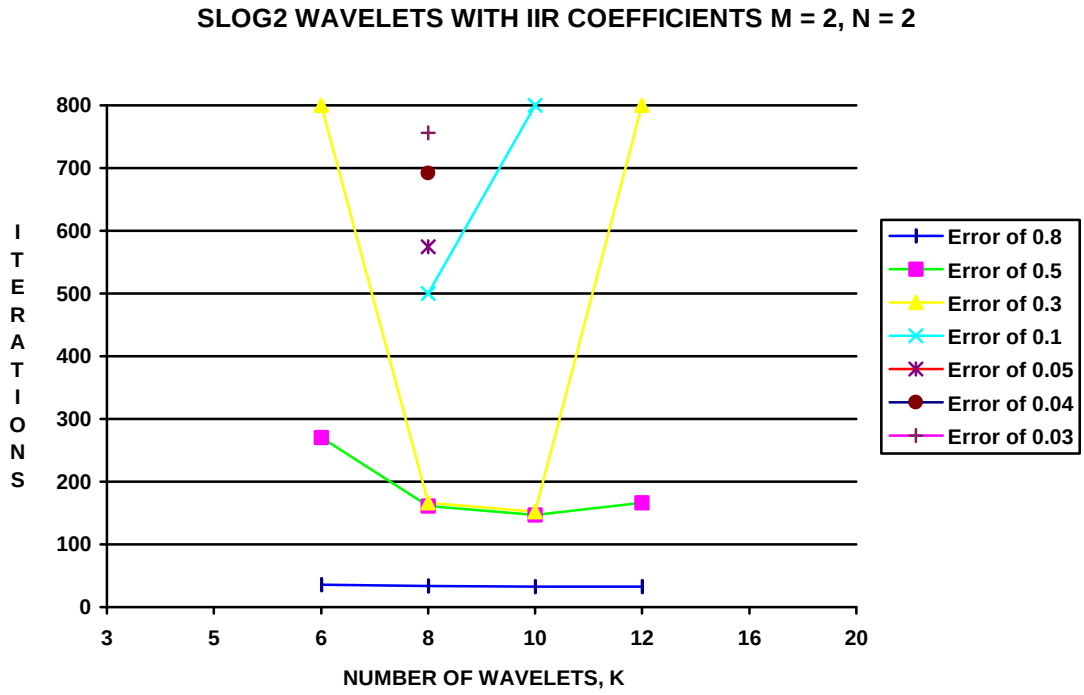


Figure 5.20 Iterations vs. Number of SLOG2 Wavelets Per Normalized Errors

Table 5.9 Number of Iterations vs. Number of SLOG2 Wavelets Employed

Number of Iterations	***Number of Wavelets, K***			
	6	**8**	**10**	**12**
Error of 0.8	36	34	33	33
Error of 0.5	270	161	147	166
Error of 0.3	(275) ∞	166	152	∞ (172)
Error of 0.1		500	∞ (157)	
Error of 0.05		574		
Error of 0.04		692		
Error of 0.03		756		

Note: The values in the parenthesis are the number of iterations when the error starts to diverge

5.6.2 Static Nonlinear Mappings

Figure 5.21 shows the identification structure of a static nonlinear mapping (similar to that of Figure 2.2b). The *wavenet* algorithm is implemented to approximate the nonlinear function:

$$f(x) = x^3 + 0.3x^2 - 0.4x \tag{5.24}$$

The simulation results of using 20 hidden nodes (or 20 mother wavelets) with (2,1) order IIR coefficients to fit the nonlinear function $f(x)$ is shown in Figure 5.22. The Mexican Hat wavelets were employed in the network. All initial weights w_k were set to zeros, all dilations a_k of each wavelet were initialized to 16, the initial translations b_k were equally spaced throughout the data, and the IIR synopsis feedforward and feedback coefficients c_k , and d_k were initially set to -0.2. Learning rates for each parameter, w_k , a_k , b_k , c_k , and d_k , to be updated are 0.01, 0.05, 0.05, 0.02, 0.02, respectively. From the results we can compare with Narendra's paper [NP89] suggesting that the more singular points exist (the points where the derivative of the function is zero), the more difficult it is to fit with a neural network. From Mars' results [MCN96] to the function $f(x)$ using the two hidden layer back-propagation algorithm with 20 and 10 hidden nodes, it takes approximately 50,000 iterations to reach about the same results obtained in Figure 5.22 for 100 iterations with 0.02427 normalized error. This experiment thus confirms how quick the *wavenet* networks can learn comparing to the MLP with backpropagation networks.

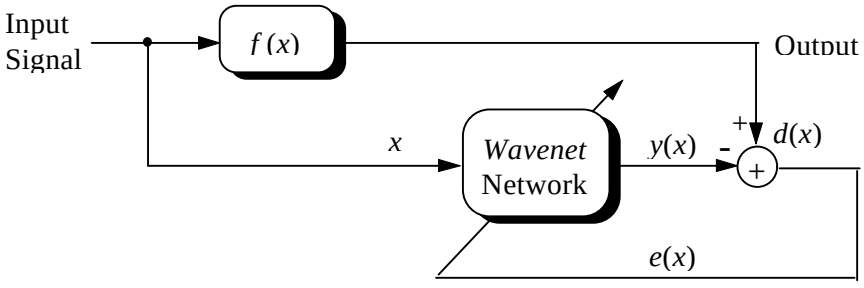
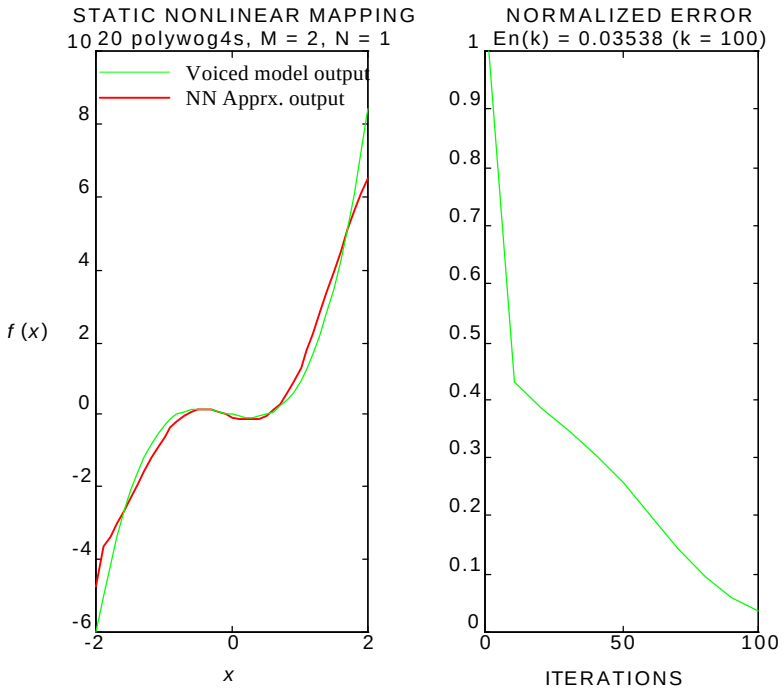
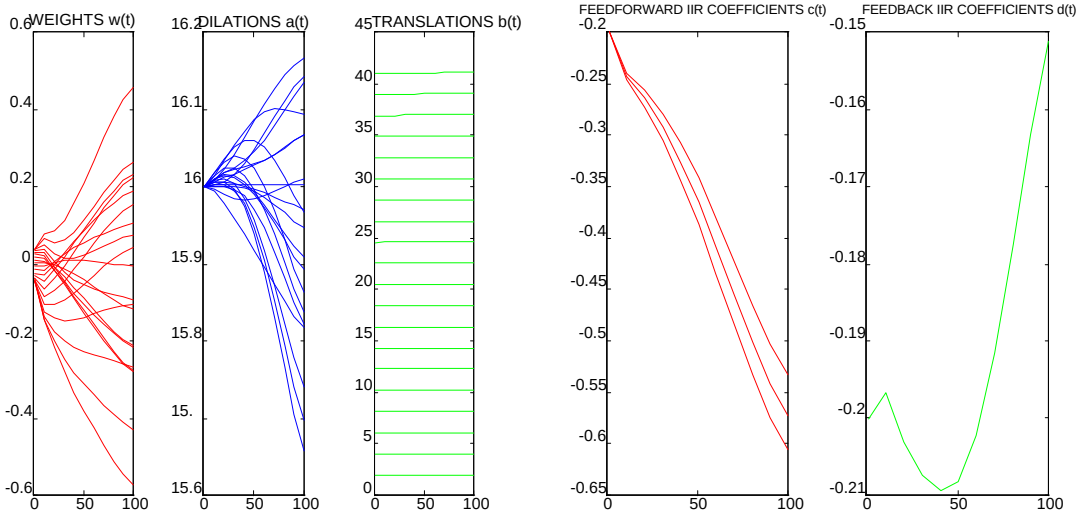


Figure 5.21 Neural Network Identification Model



(a)



(b) (c)

Figure 5.22 Static Mapping Identification of $f(x)$:

(a) Simulation Results for 100 Iterations

(b) NN Parameter Updates

(c) IIR Coefficient Updates

5.6.3 Dynamic Systems With Only Static Nonlinearity

A second order system with only a static nonlinearity can be described by the difference equation of the form:

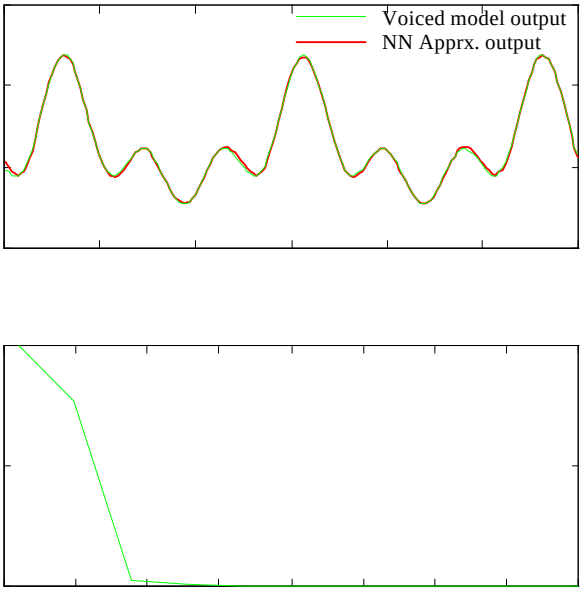
$$y(n) = 0.3y(n - 1) - 0.6y(n - 2) + f(u(n)) \tag{5.25}$$

which is a linear dynamic system with a nonlinear input mapping $u(n)$ as the exciting signal. The nonlinear function $f(.)$ has the form:

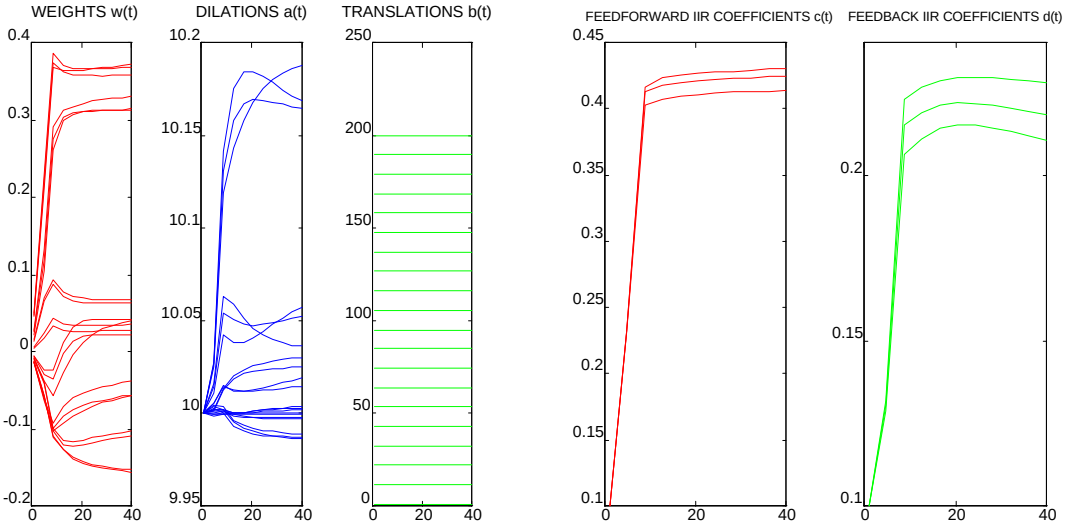
$$f(u) = u^3 + 0.3u^2 - 0.4u \tag{5.26}$$

where $u(n) = \sin(2\pi n/250)$, $n = 0, 1, 2, ..., 600$.

Using the identification structure of Figure 5.21, the "unknown" system $y(n)$ to be identified with the IIR *wavenet* network model $\hat{y}(n)$ of Eq. (5.13) can be reconstructed using 20 Morlet wavelets and (2,3) order IIR coefficients. Learning rates for each parameter, w_k, a_k, b_k, c_k, and d_k, to be updated are 0.01, 0.05, 0.05, 0.02, 0.02, respectively. All initial weights were set to 0, all dilations were initialized to 10, the initial translations were equally spaced throughout the data, and the IIR feedforward and feedback coefficients were initially set to 0.1. Figure 5.23 shows the *wavenet* identification results of the system dynamics of Eq. (5.25) with a nonlinear input of Eq. (5.26). The convergence with reference to the number of iterations is very rapid, showing that the normalized error of 0.0008069 is obtained within 40 iterations. This is a property of *wavenets* in that they rapidly find regions of near optimal solutions. In comparison to the MLP neural network where Mars stated that "randomness can actually lead to a better identification and irregular excitation should be used to break down the tracing by the weight updating and force identification process into action" [MCN96], *wavenets* do not require irregularity in the excitation to precipitate their actions. The *wavenet* networks have their own excitations in the structure themselves based on the super-mother wavelet basis functions. Particular inputs to the *wavenet* network are not really necessary as can be seen from Figures 5.1 and 5.2a, the most important input parameter of the *wavenet* structure (in the hidden layer) is the *time, t*. Thus, one of the advantages of using the *wavenet* structures is to have one less constraint condition to decide on an appropriate input excitation.



(a)



(b) (c)

Figure 5.23 Dynamic Systems With Only Static Nonlinearity

(a) *Wavenet* Simulation Results for 40 Iterations

(b) NN Parameter Updates

(c) IIR Coefficient Updates

5.6.4 Identification of Systems With Nonlinear Dynamics

This is an example of a nonlinear system that has nonlinear dynamics but linear input excitation. The system can be described by a nonlinear difference equation as

$$y(n) = \frac{y(n-1)}{1+y^2(n-1)} + u(n) + w(n) \tag{5.27}$$

where $w(n)$ is the noise with uniform distribution (variance $\sigma^2 = 0.05$) and

$u(n) = 0.8\sin(2\pi n/50)$, $n = 0, 1, 2, ..., 200$.

Figure 5.24 provides the input excitation to the nonlinear dynamic system of Eq. (5.27). Twenty Morlet wavelets were used with two IIR feedforward coefficients and two IIR feedback coefficients. Learning rates for each parameter, w_k , a_k , b_k , c_k , and d_k , to be updated were 0.005, 0.01, 0.01, 0.01, 0.01, respectively. All initial parameter values were kept the same as in the previous section. The identification process takes approximately 40 iterations for the normalized error of 0.01894, as shown in Figure 5.25. Because the wavelet transforms have a more interpolated or a spline-like characteristic, the approximated signal is more smooth and more resistant to noise. After the identification part of the "unknown" system is done, the hard part is to control the system, particularly in the nonlinear dynamic case, according to the reference or set-point model. The next chapter will describe the control problems in more details.

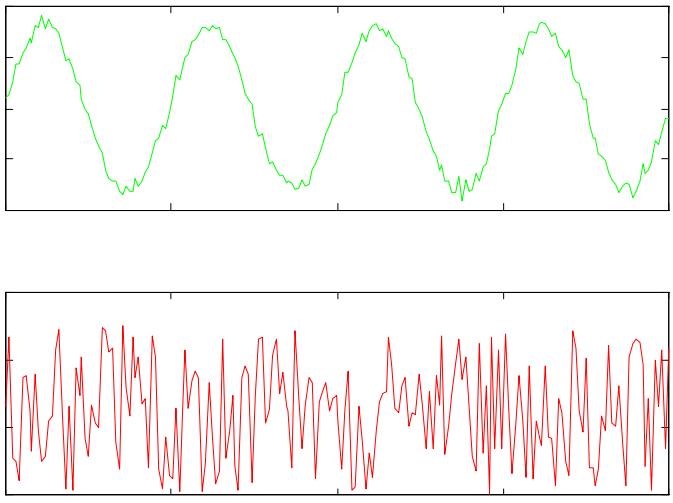


Figure 5.24 Input Excitation to the Nonlinear Dynamic System

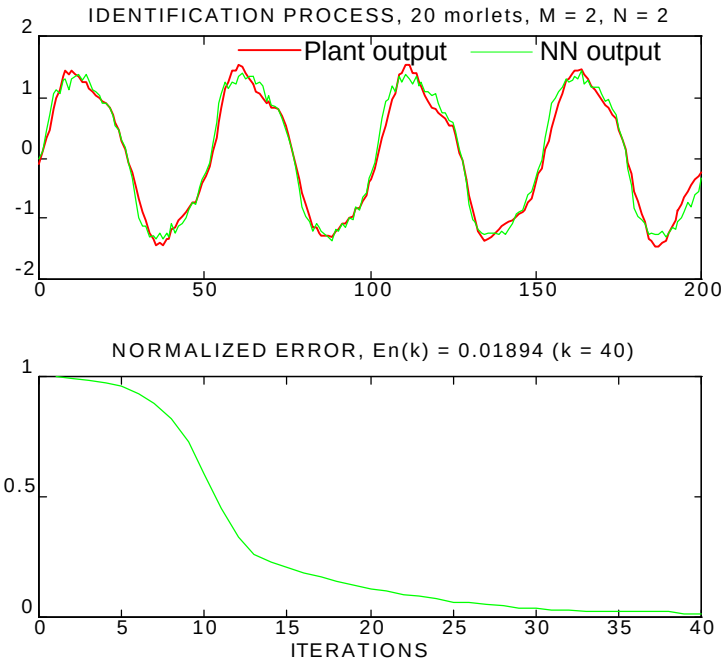


Figure 5.25 Simulation Results for the Nonlinear Dynamic System

5.6.5 Identification of a Noisy Impulse Corrupted Model

Another advantage of using the neural network adaptive wavelets is their robustness to interference such as when signals are contaminated by a non-Gaussian noise or when signal are characterized by outliers, leverages, and long-tailed effects, etc. To demonstrate these effects, the voiced case /a/ is resumed with 10 dB SNR (noise $\sigma^2 = 0.1$) and an outlier at $t = 50$. Six Morlet wavelets were employed without the IIR block structure. All parameters w_k, and a_k, were initialized to 0, and 8, respectively. Translations b_k of each wavelet were set to 5, 25, 46, 70, 90, and 100, respectively. The learning rates μ of all parameters were fixed at 0.005. The normalized error is calculated with respect to the original clean speech data and is found to be 0.2316 at 200th iteration. Figure 5.26 presents the plots of the original voiced phoneme /a/, the noisy impulse corrupted voiced data, and the resynthesized speech signal using the estimated optimum parameters corresponding to six Morlet wavelets. As a result, the network can tolerate some noisy effects and impulse corruption while maintaining the quality of speech.

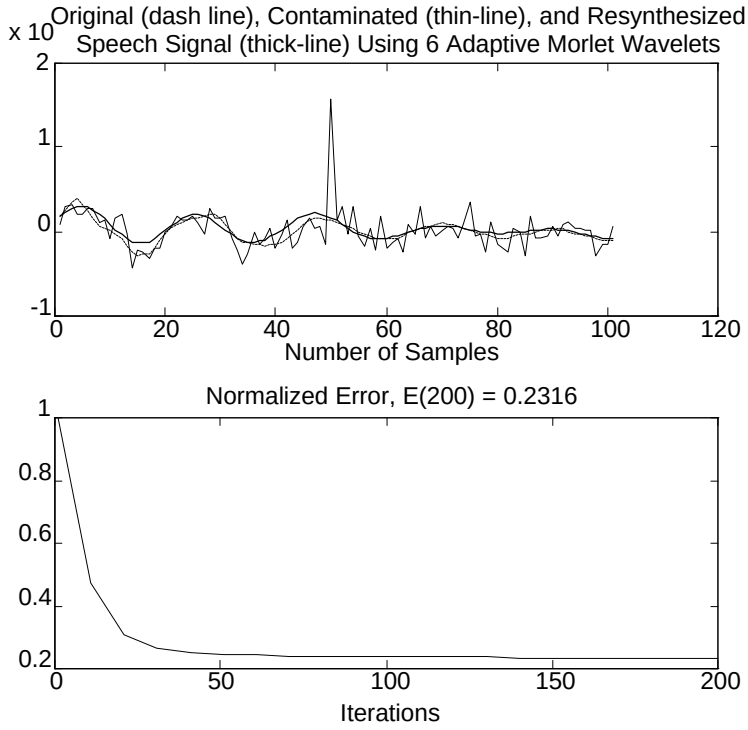


Figure 5.26 Speech Signal of a Noisy Impulse Corrupted Voiced Phoneme /a/

5.6.6 Input Noise Immunity Study

To study the noise immunity of the *wavenet* algorithms for the identification of the system with nonlinear dynamics:

$$y(n) = \frac{y(n-1)y(n-2)}{1+y^2(n-1)+y^2(n-2)} + u(n) \tag{5.28}$$

where $u(n) = 0.8\sin(2\pi n/50)$, random noise with normal distribution $\sigma^2 = 0.1$ is added to the input port of the system, as shown in Figure 5.27.

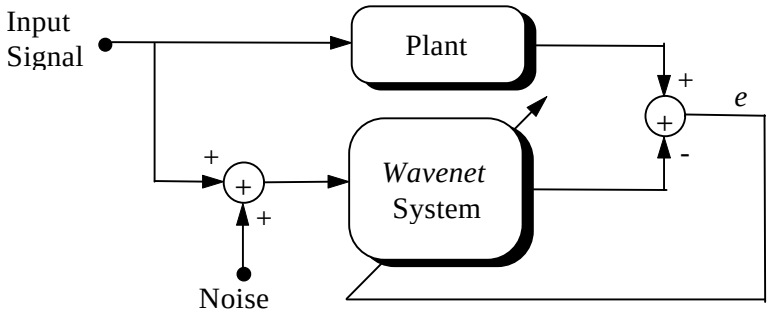


Figure 5.27 Identification to Input Noise Immunity System

Twelve RASP1 *wavenet* network with (1,1) IIR coefficients were employed. Learning rates for each parameter, w_k, a_k, b_k, c_k, and d_k, to be updated are 0.01, 0.05, 0.05, 0.02, and 0.02, respectively. All initial weights were set to 0, all dilations were initialized to 10, the initial translations were equally spaced throughout the data, and the IIR feedforward and feedback coefficients were initially set to 0.1. For comparison, the identification performance of a noiseless system of Eq. (5.28) was simulated first and demonstrated in Figure 5.28. The normalized error found was 0.02352 at the 50th iteration. Figure 5.29 provides the results to the RASP1 *wavenet* representation of the nonlinear dynamic system with noisy input. After about 50 iterations, the normalized error was 0.05744. The simulation results indicate that the identification becomes poor when the input to the *wavenet* network is contaminated with noise that increases in variance. Unlike the previous example that the model data itself is contaminated with noise, yet the *wavenet* network can represent the true voice phoneme and resist to noise and impulse corruption. A detailed description of the output noise immunity study will be described next, in the following section. Note that an equivalent input noise immunity block diagram can be modeled and interpreted by adding the measurement noise input to the output port of the NN output responses rather than at the input port of the network (See Section 6.3.2).

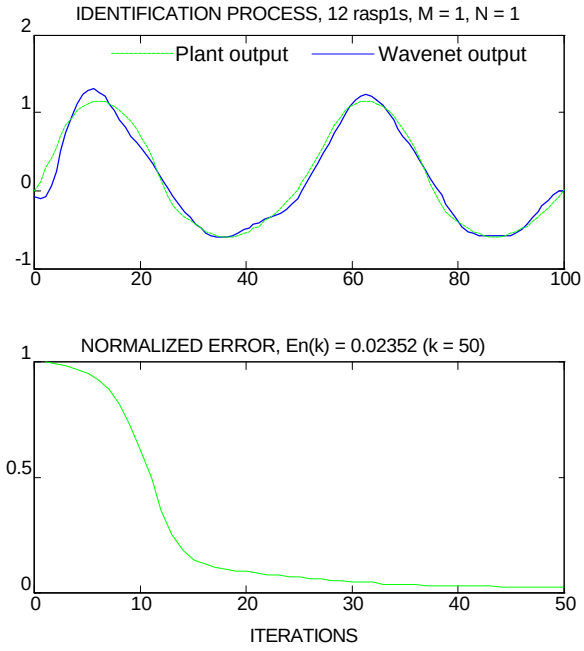


Figure 5.28 Identification to Nonlinear Dynamic System

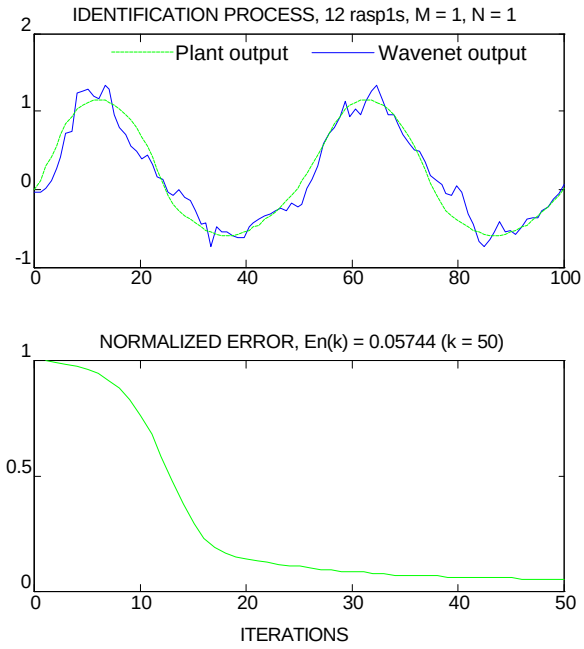


Figure 5.29 Identification to Input Noise Immunity Network

5.6.7 Identification to Output Noise Immunity

If the noise with normal distribution $\sigma^2 = 0.1$ is added at the output port of the same nonlinear plant system of Eq. (5.28), rather than at the input of the neural network, as described in Figure 5.30. All network parameters were initially kept the same as before. By inspection, the identification with respect to the true clean model is more satisfactory than the previous case of Section 5.6.6 where the normalized error between the clean plant output and the reconstructed network output responses was 0.0426. The normalized error of 0.2057 shown in Figure 5.31 was with respect to the error between the noisy plant response and the neural network emulator output. This simulation again emphasizes over the network superiority to robustness and noise interference resistance. Equivalent output noise immunity model is also given in Section 6.3.3.

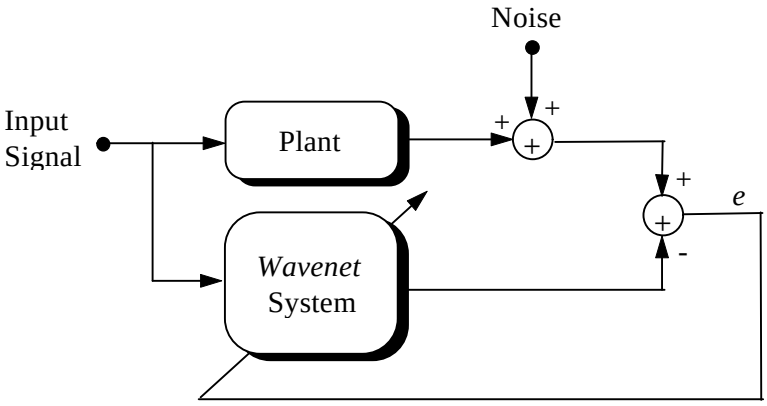


Figure 5.30 Identification to Output Noise Immunity System

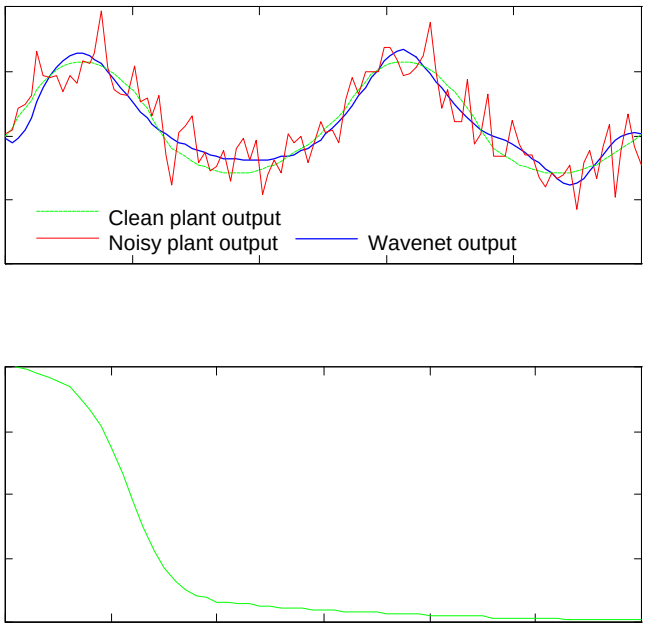


Figure 5.31 Identification to Output Noise Immunity Network

Chapter 6

Wavenet Controllers Design

Traditional self-tuning adaptive control approaches are limited in that they cannot deal with complex nonlinear systems. Typically, these techniques assume that the control model is operating in a linear region. The parameters of a *linearized* plant model are estimated recursively and used to update the controllers. Generally, it is not possible to design a controller based on mathematical analysis for such plants that consist of the nonlinearity and the uncertainty. The problem is exacerbated when the functions describing the plants are unknown and change with time. Such nonlinear time-varying adaptive control problems are arising with increasing frequency in today's technology. It is important to develop an effective technique in which the structure of the unknown, linear/nonlinear plant models can be identified as an adaptive process; and controllers have to be designed which act rapidly, accurately and in a stable fashion.

Unlike the previous Chapter where the identification process in which the parameters of the IIR synopsis *wavenet* network are adjusted off-line (batch mode learning), the parameters of the proposed predictive controls are adjusted on-line within a feedback loop. The computation of a gradient in this case is considerably more involved and computationally intensive.

Two major neuro *wavenet*-based control structures will be presented. The first one is self-tuning neuro *wavenet* controllers that approximate the unknown uncertain dynamics of the plant systems, then the on-line predictive controls are based on the approximated algebraic computations following the sampled information. The second one is adaptive self-tuning PID controllers using *wavenet*s. In this method, the *wavenet* network is needed to learn the characteristics of the plant dynamic systems and make use of it to determine the future inputs that will minimize error performance index so as to compensate the PID controller parameters. Advantageous and disadvantageous of the two proposed controllers will be discussed in each section with illustrations. A well-known traditional backpropagation training will also be included in the control design for a base line

comparison. In the remainder of this Chapter, the *wavenet* models will be based on Chapter 5. Next, the principal ideas involved in each design will be outlined in each section.

6.1 Self-Tuning Neuro *Wavenet* Controllers

6.1.1 Structure and Definition

Consider a general SISO dynamical system to be represented by the state equations

$$x(k+1) = f(x(k), u(k), k) \tag{6.1}$$

$$y(k) = g(x(k), k) \tag{6.2}$$

where $x(k) \in \mathsf{R}^n$, and $u(k), y(k) \in \mathsf{R}$. Further, let the unknown functions $f, g \in C^1$. The only accessible data are the input u and output y. It has been shown [LN95] that if the linearized system around the equilibrium state is observable, an input-output representation exists which has the form

$$y(k+1) = y(\, y(k), y(k\text{-}1), ..., y(k\text{-}n+1), u(k), u(k\text{-}1), ..., u(k\text{-}n+1)) \tag{6.3}$$

i.e., a function $y(\cdot)$ exists that maps $y(k)$ and $u(k)$, and their n-1 past values, into $y(k+1)$. In view of this, a *wavenet* network model $\hat{y}$ can be trained to approximate y over the domain of interest. Practically even if an *exact* model of the plant is available, approximate models are adapted to update the control parameters on-line. The considerations are based on the neural network controller design of the control system. The following alternative model of an unknown plant that can simplify the computation of the control input is described by the equation [NBK95]:

$$\begin{aligned} y(k+1) = {} & F\big(y(k), y(k-1), ..., y(k-n+1), u(k), u(k-1), ..., u(k-n+1)\big) + \\ & G\big(y(k), y(k-1), ..., y(k-n+1), u(k), u(k-1), ..., u(k-n+1)\big) \cdot u(k) \end{aligned} \tag{6.4}$$

For the sake of conciseness and simplicity in the discussion, the complex system representations described above will be generalized for a discrete-time process of dimension 1 of the form [KSS91]:

$$y(k+1) = F\big(y(k)\big) + G\big(y(k)\big) \cdot u(k) \tag{6.5}$$

where $y(k)$ and $u(k)$ denote the input and the output at the kth instant of time.

If the nonlinearity terms $F(\cdot)$ and $G(\cdot)$ are known exactly, the required control $u(k)$ for tracking a desired output $r(k+1)$ can be computed at every time instant using the formula

$$u(k) = \frac{r(k+1) - F(y(k))}{G(y(k))} \tag{6.6}$$

However, if $F(\cdot)$ and $G(\cdot)$ are unknown, the idea is to use the neural network adaptive wavelets model to approximate the system dynamics i.e.,

$$\hat{y}(k+1) = \hat{F}(y(k), Q_F) + \hat{G}(y(k), Q_G)\, u(k) \tag{6.7}$$

Comparing the model of Eq. (6.7) with the one of Eq. (5.13), we can conclude that

$$\hat{F}(y(k), Q_F) = \sum_{j=1}^{N} d_j\, \hat{y}(k-j)\, v(k) \tag{6.8}$$

$$\hat{G}(y(k), Q_G) = \sum_{i=0}^{M} c_i z(k-i) \tag{6.9}$$

where

$$z(k) = \sum_{l=1}^{K} w_l h\left(\frac{k-b_l}{a_l}\right) \tag{6.10}$$

After the nonlinearities $F(\cdot)$ and $G(\cdot)$ are approximated by the two distinct neural network functions $\hat{F}(\cdot)$ and $\hat{G}(\cdot)$ with adjustable parameters (including weights w_k, dilations a_k, translations b_k, IIR feedforward coefficients c_k, and IIR feedback coefficients d_k), represented by Q_F and Q_G respectively, the control $u(k)$ for tracking a desired output $r(k+1)$ can be obtained from

$$u(k) = \frac{r(k+1) - \hat{F}(y(k), Q_F)}{\hat{G}(y(k), Q_G)} \tag{6.11}$$

The neuro controller for self-tuning control system is provided in Figure 6.1.

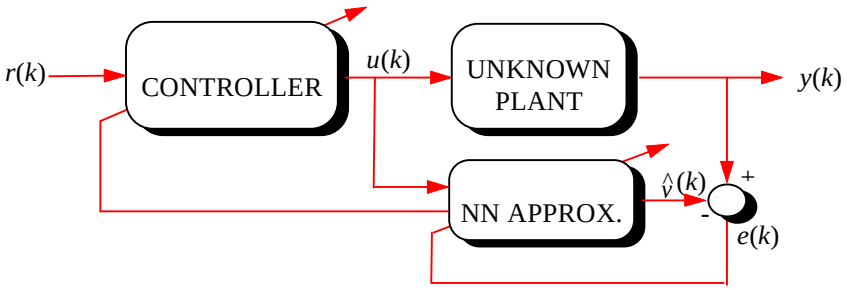


Figure 6.1 Self-Tuning Neuro Controller System

6.2 Adaptive PID Controllers Using *Wavenets*

6.2.1 Structure and Algorithms

The *Proportional-Integral-Derivative* (PID) controller is one of the simplest of the traditional feedback controller schemes. Nevertheless, the linear PID algorithm might be difficult to deal 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 *wavenet* 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 of disadvantages to the linear P-I-D 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 P-I-D 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). Several tuning components determine the contribution of the weights of the error that suits a cost function $E = \frac{1}{2}\sum_{k=1}^{T}(r(k) - \hat{y}(k))^2$, where $r(k)$ is the desired set-point and $\hat{y}(k)$ is the *wavenet* output. The digital PID controller can be expressed in discrete-time as follows [WWZZ95], [O95]:

$$u(k) = u(k\text{-}1) + P\,[\,\varepsilon(k) - \varepsilon(k-1)\,] + I\,\varepsilon(k) + D\,[\,\varepsilon(k) - 2\varepsilon(k-1) + \varepsilon(k-2)\,] \qquad (6.12)$$

where P, I, and D are proportional, integral, and differential gains, $u(k)$ is a plant input at kT, where T is a sampling interval, and

$$\varepsilon(k) = r(k) - y(k) \qquad (6.13)$$

P, *I*, *D* parameters are considered as part of the function of *E* and can be optimized and updated according to the cost function *E* of Eq. (5.4),

$$P(k) = P(k-1) + \mu_P \, e(k) G(k) \big(\varepsilon(k) - \varepsilon(k-1)\big) \tag{6.14}$$

$$I(k) = I(k-1) + \mu_I \, e(k) G(k) \varepsilon(k) \tag{6.15}$$

$$D(k) = D(k-1) + \mu_D \, e(k) G(k) \big(\varepsilon(k) - 2\varepsilon(k-1) + \varepsilon(k-3)\big) \tag{6.16}$$

where $e(k) = y(k) - \hat{y}(k)$, $\hat{G}$ comes from Eq. (6.9), and $m(k)$ is the fixed learning rate of each adaptive PID parameter. Figure 6.2 depicts the block diagram of the resulting network topology based on the PID controller paradigm.

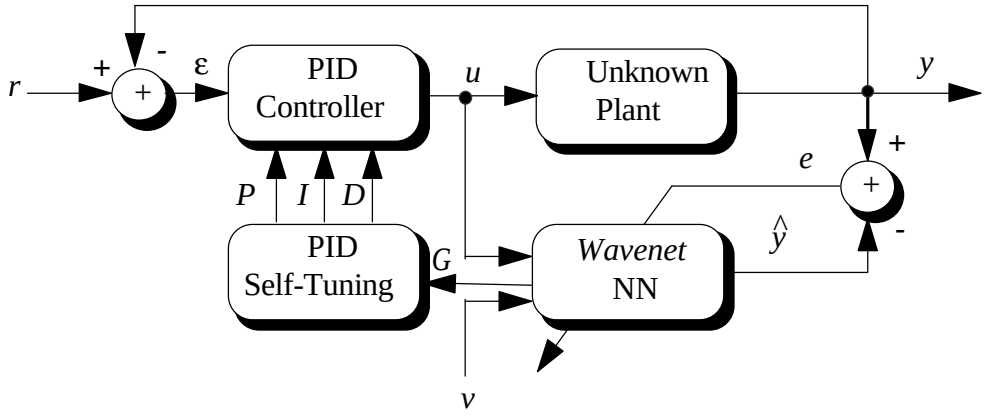


Figure 6.2 Adaptive PID Self-Tuning Neuro Strategy Scheme

6.3 Simulation Study

6.3.1 Control of the Nonlinear Dynamic Systems

Using the same noisy nonlinear difference equation of Eq. (5.27), the adaptation process according to the approximated model of Eq. (6.7) was identified from the previous Chapter as shown in Figure 5.25. The task now is to find a control $u(k)$ that can track the setpoint reference:

$$r(k) = u(k\text{-}200) - 1.5u(k\text{-}300) + 1.5u(k\text{-}400) \tag{6.17}$$

for k = 200, 201, ..., 500, where $u(k)$ represents the unit step function

$$u(k) = \begin{cases} 1 & \text{for } k \geq 0 \\ 0 & \text{for } k < 0 \end{cases} \tag{6.18}$$

6.3.1.1 Neuro *Wavenet* Controller

Figure 6.3 illustrates the results of the setpoint control using the proposed self-tuning neuro *wavenet* controller with 20 Morlet wavelets. Dash line represents the response from the NN emulator used in the adaptation of the self-tuning control process. Thick line denotes the output response from the actual plant of the system. The same control $u(k)$ is fed to both the actual plant and the neural network emulator. The mean squared error (MSE) with respect to the setpoint reference $r(k)$ of the actual plant $y(k)$ and of the NN output response $\hat{y}(k)$ were 0.01252 and 0.005962, respectively. The response to the control input of the unknown nonlinear plant is what we aim for and its performance displays fairly good results to this type of control scheme. Figure 6.4 shows the control input $u(k)$. Figure 6.5a provides the network parameter updates. Note that the super wavelet parameters w_k, a_k, b_k, were merely adjusted during the tracking period while the IIR coefficients c_k, and d_k were adjusted very often. This demonstrates that after the identification process of the *wavenet* network, the network parameters become very stable around the equilibrium state. Figure 6.5b shows the update of the nonlinearity terms $\hat{F}(\cdot)$ and $\hat{G}(\cdot)$ during the identification period (k = 0 to 200) and during the tracking period (k = 200 to 500).

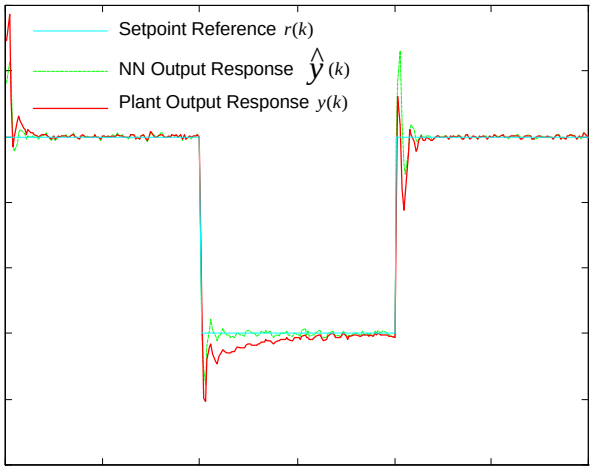


Figure 6.3 Self-Tuning Neuro *Wavenet* Controller Responses to Set-Point Reference

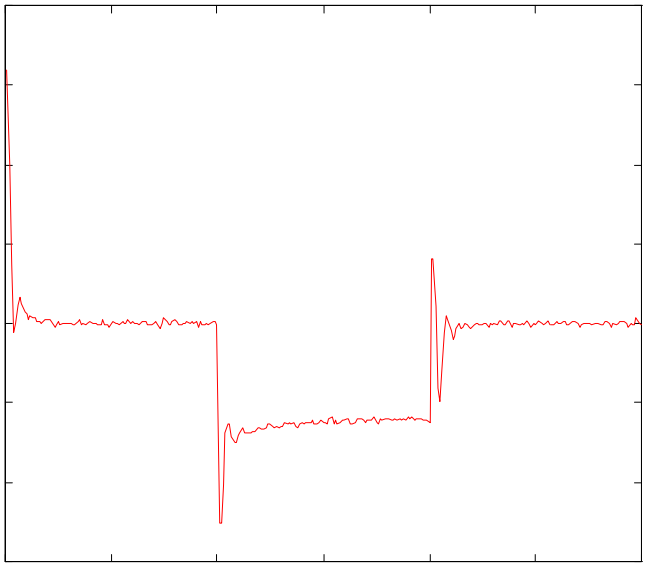
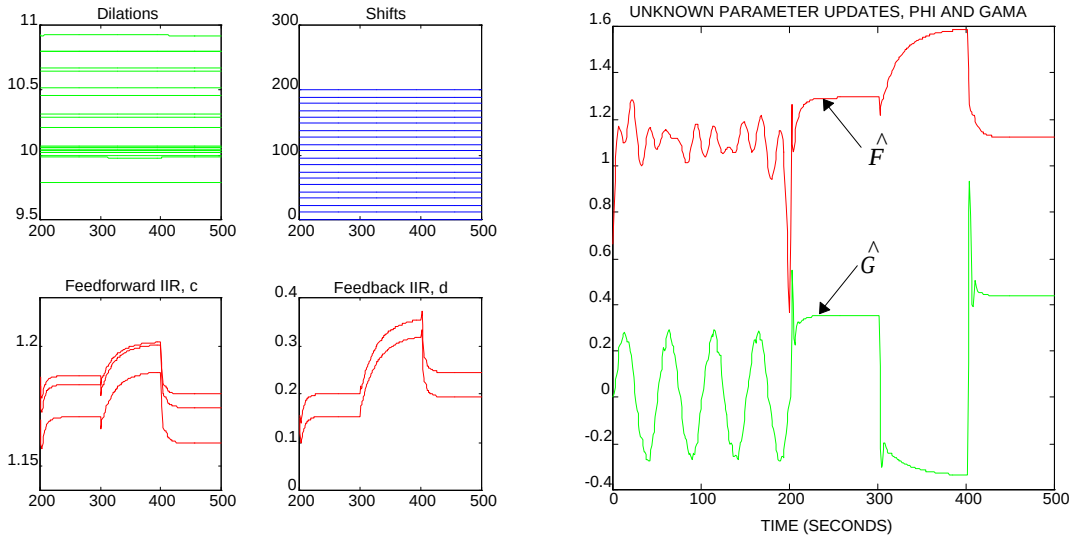


Figure 6.4 Self-Tuning Neuro *Wavenet* Control Input



(a) (b)

Figure 6.5 Self-Tuning *Wavenet* Parameters Tracking to Set-Point Reference

(a) *Wavenet* Parameter Updates (b) Nonlinearity Term Updates, $\hat{F}(\cdot)$ and $\hat{G}(\cdot)$

6.3.1.2 Adaptive PID Controller Using *Wavenets*

Figure 6.6 shows the simulation results to the set-point control using the proposed adaptive self-tuning PID controller with self-adjustable *wavenet* parameter adaptation. The network utilizes the same 20 Morlet mother wavelets with (2,2) IIR coefficents. Controller parameters P, I, and D were initially set at 0.2, 0.1, and 0.005, respectively which then later vary with local control NN conditions. Again the emphasis is on the plant output responses to the reference setpoint. Unlike the previous study where the NN output responses track the set-point reference faster than the actual plant responses, adaptive PID control of the actual plant output follows the desired set-point optimally quicker than the NN responses. The MSE with respect to the desired unit step reference $r(k)$ of the actual plant $y(k)$ and of the NN output response $\hat{y}(k)$ were 0.03492 and 0.06433, respectively. Control response to the adaptive PID controller is shown in Figure 6.7 and Figure 6.8 provides adaptive updates to each parameter applied to the control scheme.

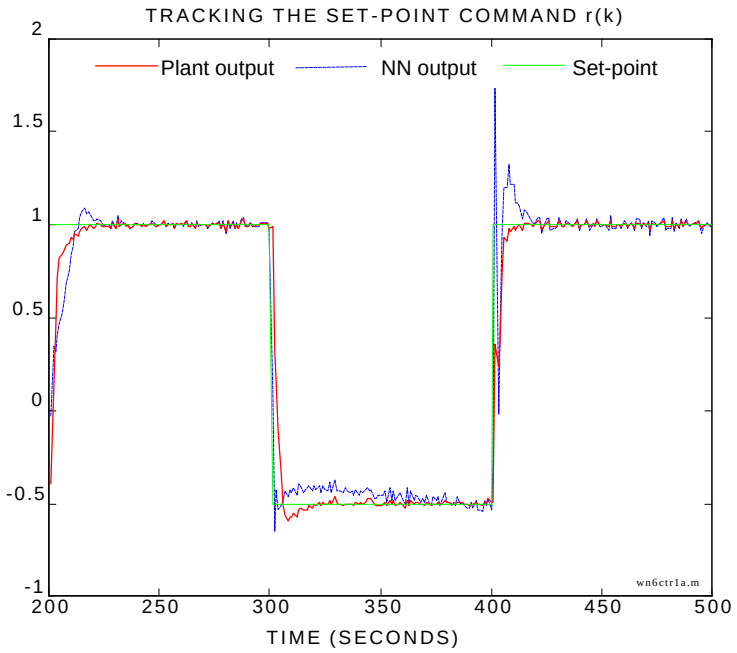


Figure 6.6 Adaptive Self-Tuning PID Controller Responses to Set-Point Reference

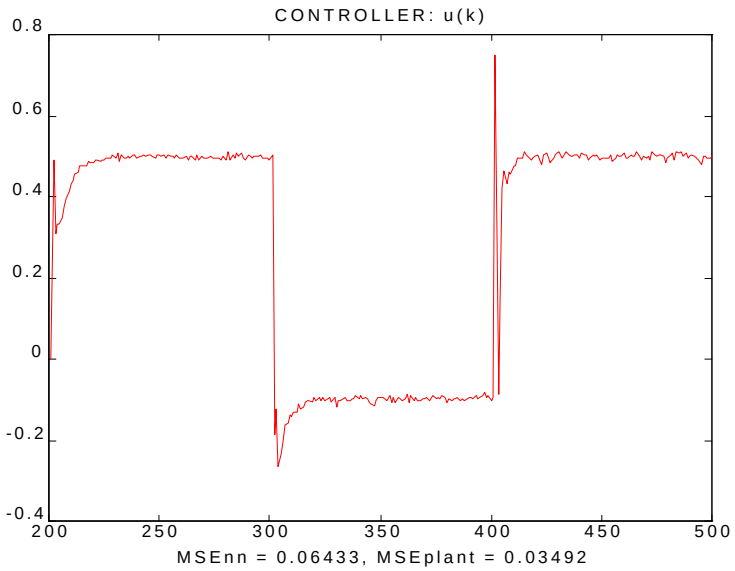


Figure 6.7 Adaptive Self-Tuning PID Control Input

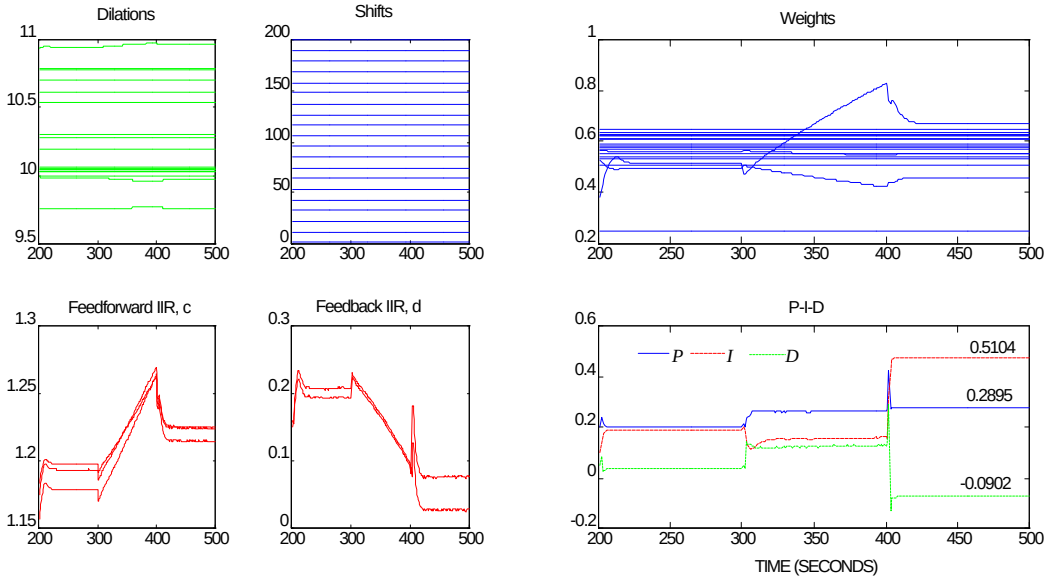


Figure 6.8 Adaptive Self-Tuning PID Parameter Updates

6.3.1.3 Adaptive PID Controller Without Prior NN Training

One can ask why the identification process is necessary for the PID controller scheme since the controller partially depends on the *wavenet* model. Figure 6.9 provides the answer.

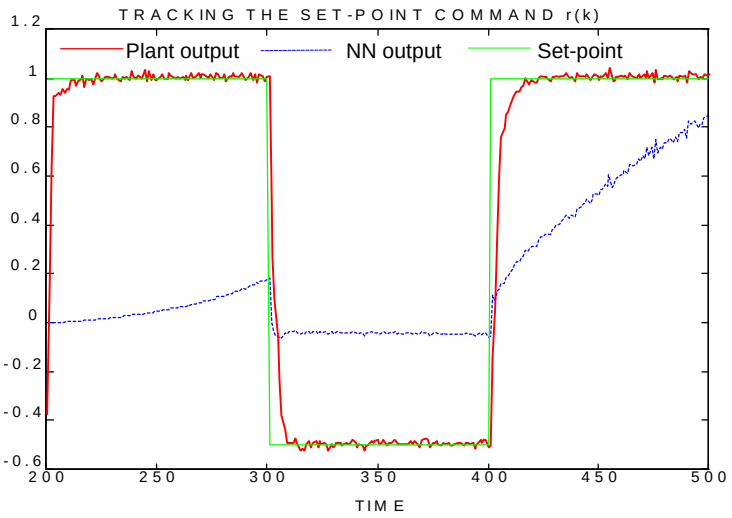


Figure 6.9 Adaptive PID Controller Responses Without NN Training

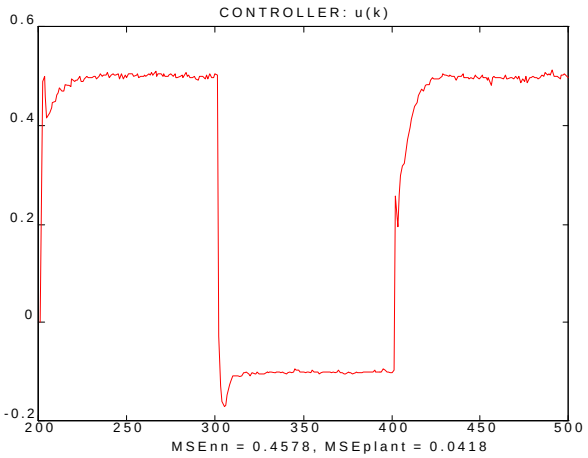


Figure 6.10 Corresponding Adaptive PID Control Action Without NN Training

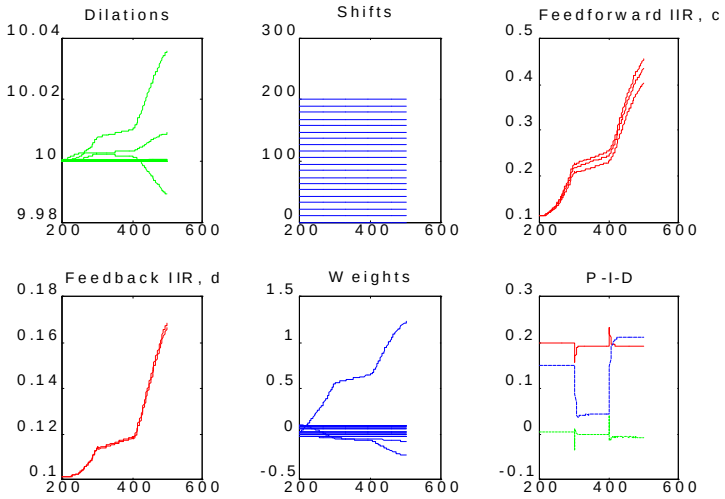


Figure 6.11 Corresponding Adaptive PID Parameter Updates Without NN Training

Without first training or identifying the *wavenet* model, the plant is still able to track the desired reference but it takes a longer time to reach (MSE of 0.0418 compared to the previous MSE of 0.03492). With the *wavenet* network cascading to the IIR synopsis structure providing double local structure, it does not consume excess time to train the network to emulate the plant dynamics, thus it can also be learned online and provides better tracking performance. Note since there is no learning to the NN, the NN output to the control action of Figure 6.10 stays unmodeled. Figure 6.11 shows the major parameter adjustments adapting to the control changes. The updated values of the same initial *P*, *I*, and *D* parameters from the previous section were found to be 0.1921, 0.2110, and -0.0068, respectively.

6.3.1.4 Fixed PID Controller Using *Wavenets*

Now if we set the *P*, *I*, and *D* parameters constant at 0.2, 0.1, and 0.05, respectively without self-adjusting from the *wavenet* emulator, the plant takes a little longer time (MSE of 0.04709) to reach the target than the variable PID parameters as illustrated in Figure 6.12.

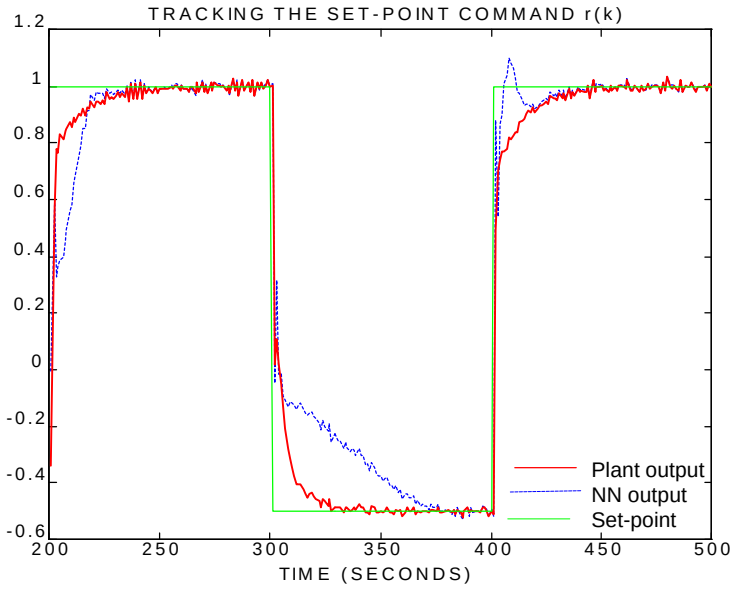


Figure 6.12 Corresponding Static PID Controller Responses

6.3.1.5 Neuro-Controller Based on the Backpropagation Algorithms

A base line comparison is demonstrated in this section by providing a traditional feedforward MLP neural network structure based on the backpropagation algorithms. Two-layer perceptron network (1 hidden layer, 1 output layer) is simulated with 20 hidden nodes. The bias for each node is set at 0.5. Initial weights of both layers are set at random. Adaptive learning rate and its momentum coefficient are initially set at 0.005 and 0.95, respectively. Hyperbolic tangents are employed in the hidden nodes as an activation function. Figure 6.13 illustrates the results of the setpoint control using the BPP algorithms. As previously mentioned, the backpropagation learning does not perform well on-line because of its global characteristic. The tracking response of the plant needed a longer time to reach the desired target and the MSE with respect to the reference $r(k)$ of the actual plant output was found to be 0.1298. The neural network emulator in Figure 6.13 indicates that its response could not capture the dynamic changes on real time and thus, could never provide the good one step predictive control performance to the tracking operation. Batch mode adaptation is needed for this type of algorithms. Obviously, they are not suitable for this particular type of control applications.

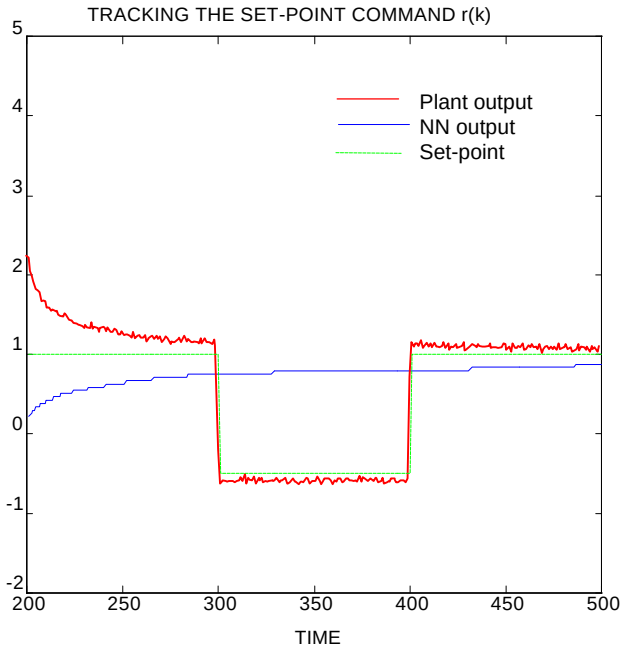


Figure 6.13 Neuro-Controller with BPP Learning Responses to Set-Point Reference

6.3.2 Control of the Input Noise Immunity Problem

The block diagram of an input noise immunity model of Figure 5.27 can be equivalently interpreted the same as the block diagram of Figure 6.14 with the measurement noise input is added to the output port of the NN output responses. The similarity of the two models can be easily shown from the proposed model of Equations (6.5) and (6.7). The *wavenet* output response with noisy input $w(k)$ added to the input port of the network is represented by

$$\hat{y}(k+1) = \hat{F}(\cdot) + \hat{G}(\cdot)(u(k) + w(k)) = \hat{F}(\cdot) + \hat{G}(\cdot)u(k) + \hat{G}(\cdot)w(k)$$

$$= \hat{F}(\cdot) + \hat{G}(\cdot)u(k) + \tilde{w}(k) \tag{6.19}$$

and the *wavenet* response to the noise added to the network output is represented by

$$\hat{y}(k+1) = \hat{F}(\cdot) + \hat{G}(\cdot)u(k) + w(k) \tag{6.20}$$

Thus the two block diagrams provide the same results to the identification performance.

Note: The alternative model above can only be applied to nonlinear dynamic systems with linear input excitation.

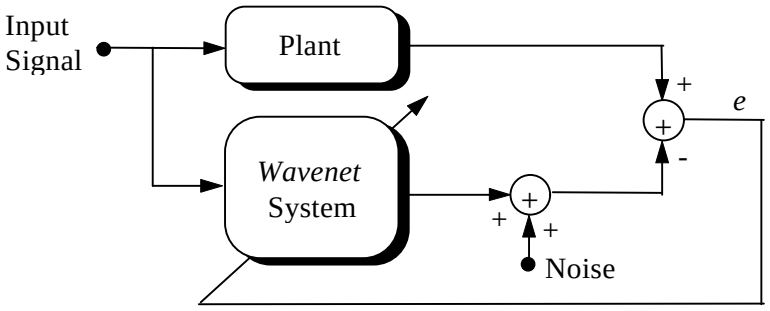


Figure 6.14 Equivalent Identification to Input Noise Immunity System

After the identification process for the unknown nonlinear system of Eq. (5.28) has been identified as shown in Figure 5.29, control action is activated to track the desired set-point reference:

$$r(k) = 0.5u(k-100) - u(k-150) \tag{6.21}$$

for k = 100, 101, ..., 200, where $u(k)$ represents the unit step function.

The measurement noise $w(k)$ of random distribution with a variance of 0.01 is still inserted at the input port of the *wavenet* network. Simulations of each control methods are shown below.

6.3.2.1 Neuro *Wavenet* Controller

All parameters stay the same as in the identification process of Section 5.6.6. The simulation runs of the plant and NN output responses when the neuro *wavenet* controller was applied is shown in Figure 6.15. Figure 6.16 demonstrates the corresponding control action to the actual plant and the noisy control to the *wavenet* network emulator. Figure 6.17 provides the necessary parameter updates of the network and control. From the results, it can be seen that when noises are mixed in the control process, the tracking operation of the actual plant to the desired reference is inferior proportional to the noise level. Note that the measurement noise is only added to the input to the NN emulator, but not to the actual plant. However, the plant responses to the control action is noisy. The *wavenet* network emulator responses to better setpoint tracking than the plant resulting to the corresponding MSE of 0.02739 and 0.0617, respectively. It can be concluded that with this type of control scheme, noises are the main distraction to the tracking operation for the actual plant system.

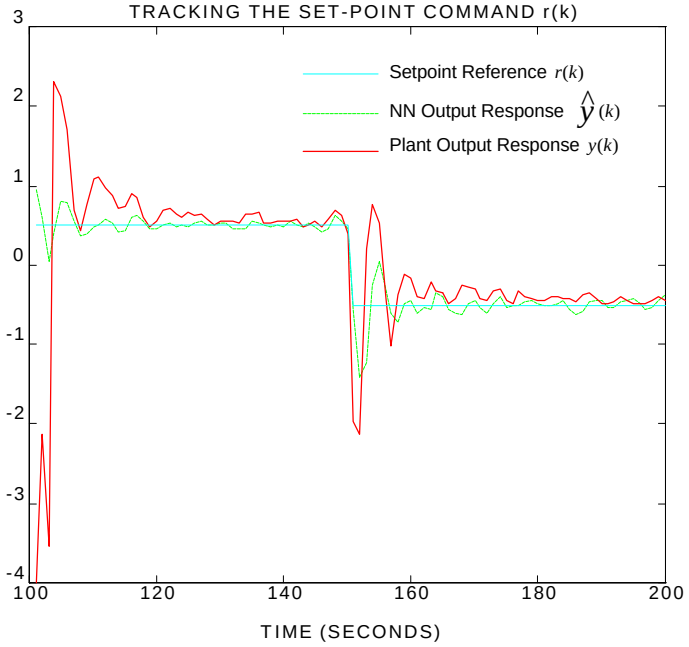


Figure 6.15 *Wavenet* Controller Responses to Set-Point Control of Input Noise Immunity

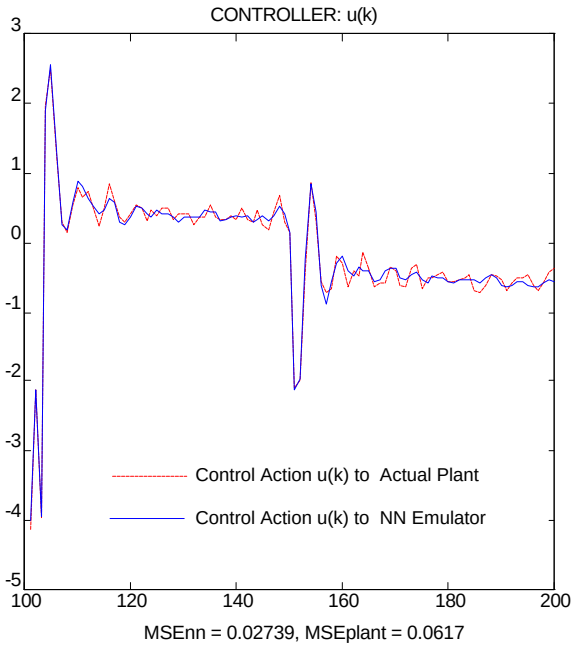


Figure 6.16 *Wavenet* Control Actions to Set-Point Control of Input Noise Immunity

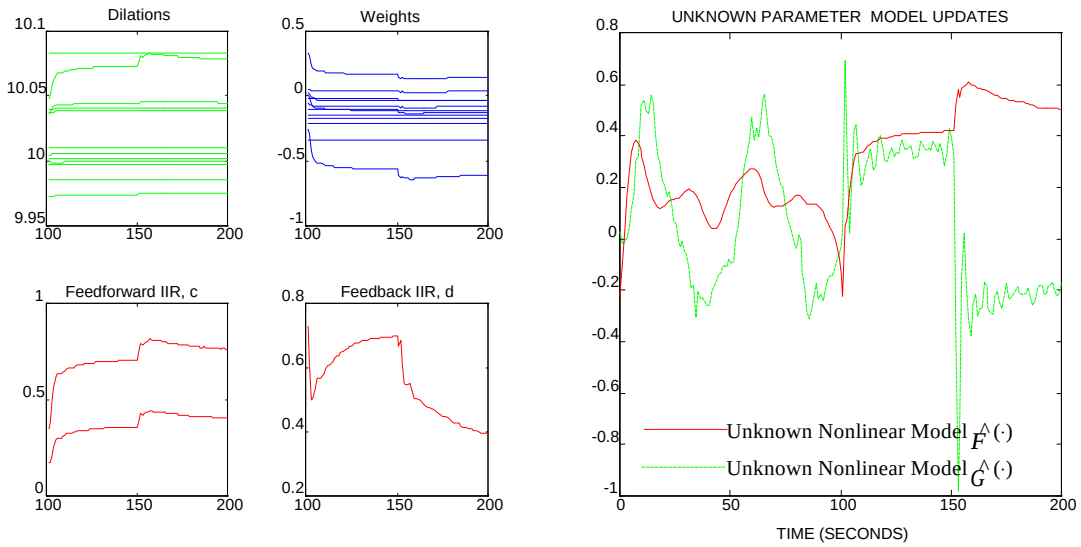


Figure 6.17 Adaptive *Wavenet* Parameters to Set-Point Control to Input Noise Immunity

(a) Super Mother Wavelet Parameter and IIR Coefficient Updates

(b) Nonlinear Term Updates

6.3.2.2 Adaptive PID Controller Using *Wavenets*

Figure 6.18 shows the simulation results to the set-point control using the proposed adaptive self-tuning PID controller with self-adjustable *wavenet* parameter adaptation. The network utilizes the same 12 RASP1 mother wavelets with (1,1) IIR coefficients as in the Section 5.6.6. Controller parameters P, I, and D were initially set to 0.2, 0.1, and 0.005, respectively. Learning rate step size for the variable P, I, and D are all fixed at 0.05. Control performance and adaptive parameters involved are shown in Figures 6.19 and 6.20, respectively. In this proposed control method, the plant output responds to the desired setpoint much better and quicker than in the previous control algorithms. The MSE for the actual plant and the NN output with respect to the reference $r(k)$ are 0.01717 and 0.3142, respectively. The control action to the actual plant is now less noisy and with less overshoot than the previous method. The NN response to the noisy control was varied slowly and did not reach the setpoint. However, the clean control based on the NN self-tuning scheme is able to capture the tracking mode. The NN emulator during the tracking period is only used for the P, I, D adaptation processes. It will be shown later that when the characteristic of unknown nonlinear systems has a slow transient or dynamic responsiveness to changes, the self-tuning PID control scheme is superior to the instant algebraic dynamic control of the previous case.

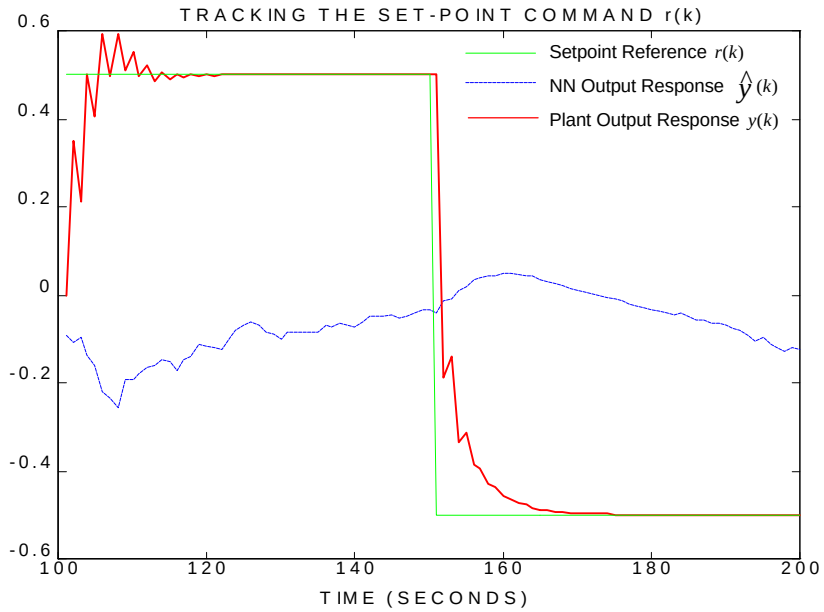


Figure 6.18 PID Controller Responses to Set-Point Control of Input Noise Immunity

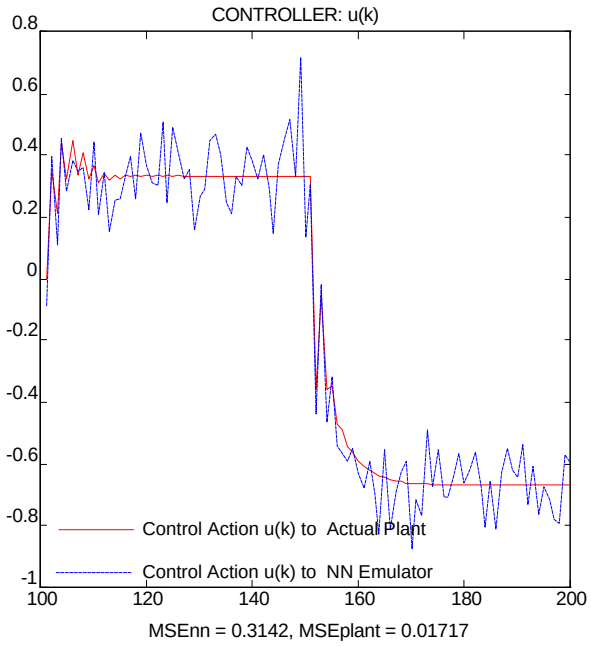
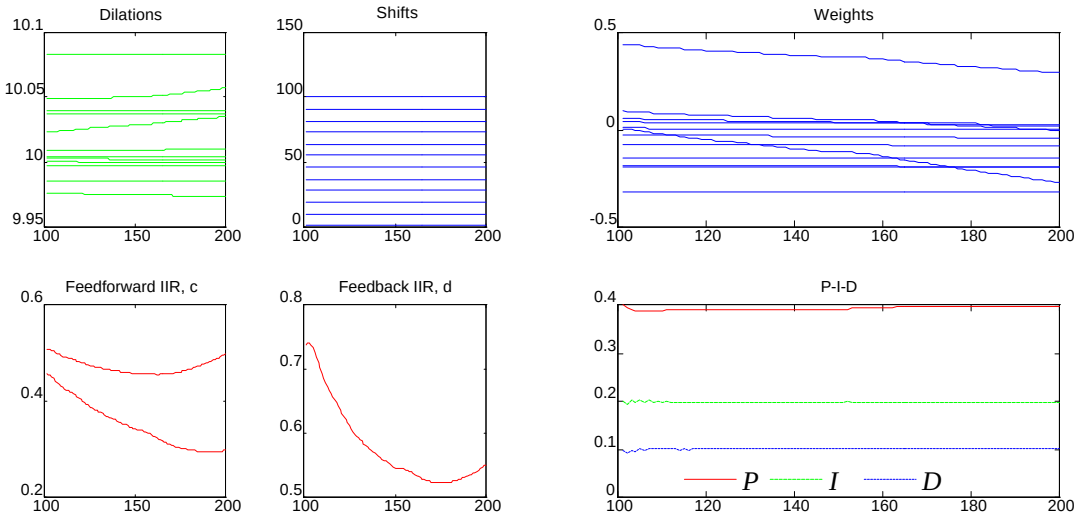


Figure 6.19 PID Control Actions to Set-Point Control of Input Noise Immunity



(a) (b)

Figure 6.20 Adaptive PID NN Parameters to Set-Point Control of Input Noise Immunity

6.3.2.3 Neuro-Controller Based on the Backpropagation Algorithms

With the same initialization as in Section 6.3.1.5, the simulation result of the input noise immunity study for the two-layer feedforward NN with backpropagation algorithms is shown in Figure 6.21. MSE of the plant was found to be 0.0531. The neural network output hardly responds to the tracking operation, and the plant output is noisy. Unlike *wavenets*, the backpropagation algorithms do not signify the interpolation characteristic, thus the NN responses will capture the noisy interference effect.

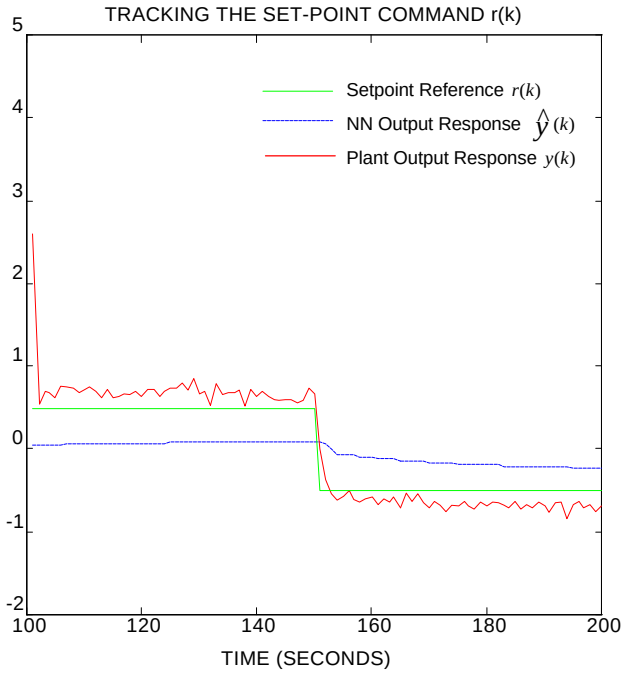


Figure 6.21 Neuro-Controller with BPP Learning Responses to Input Noise Immunity

6.3.3 Control of the Output Noise Immunity Problem

Equivalent output noise immunity model of Figure 5.30 can be implemented as in Figure 6.22 where the noise disturbance is added to the input of the actual plant system, rather than at the output of the plant. The proof can be verified from the instantaneous error of the learning algorithm. When the model of Figure 5.30 is used, the error becomes

$$e(k) = (y(k) + w(k)) - \hat{y}(k)$$

$$= [F(\cdot) - \hat{F}(\cdot)] + [G(\cdot)\,u(k) - \hat{G}(\cdot)u(k)] + w(k) \qquad (6.22)$$

and if the model of Figure 6.22 is implemented, the error is

$$e(k) = [F(\cdot) - \hat{F}(\cdot)] + [G(\cdot)\,u(k) - \hat{G}(\cdot)u(k)] + G(\cdot)w(k)$$

$$= [F(\cdot) - \hat{F}(\cdot)] + [G(\cdot)\,u(k) - \hat{G}(\cdot)u(k)] + \tilde{w}(k) \qquad (6.23)$$

Thus, the two identification models provide similar algorithmic approximation.

Note: The alternative model above can only be applied to nonlinear dynamic systems with linear input excitation.

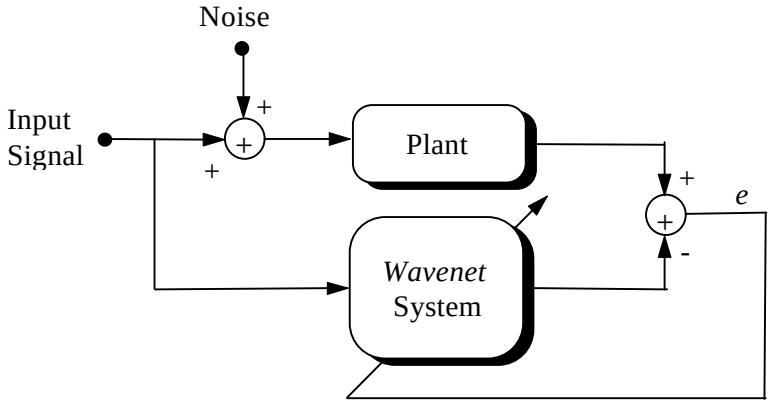


Figure 6.22 Equivalent Identification to Output Noise Immunity System

After the identification process for the unknown nonlinear system of Eq. (5.28) has been identified as shown in Figure 5.31, control action is activated to track the same desired set-point reference of Eq. (6.21). The measurement noise $w(k)$ of random distribution with a variance of 0.01 is still inserted at the output port of the nonlinear plant or to the alternative model above. Simulations of each control methods are shown below.

6.3.3.1 Neuro *Wavenet* Controller

The simulation result of the plant and NN output responses when the neuro *wavenet* controller was applied is shown in Figure 6.23. Figure 6.24 demonstrates the corresponding control action to both the actual plant and the *wavenet* network emulator. Figure 6.25 provides the necessary parameter updates of the network. The results show that there is an offset of the plant response to the set-point due to the disturbance at the output port of the

plant. Initial oscillation to both plant and NN responses is suspected to come from the stability of the NN adaptation caused by the scaling to the input port of the NN emulator. With this control scheme, the NN performance is noisy, but quick to the set point, while the plant output performance is affected by the disturbance of the noise.

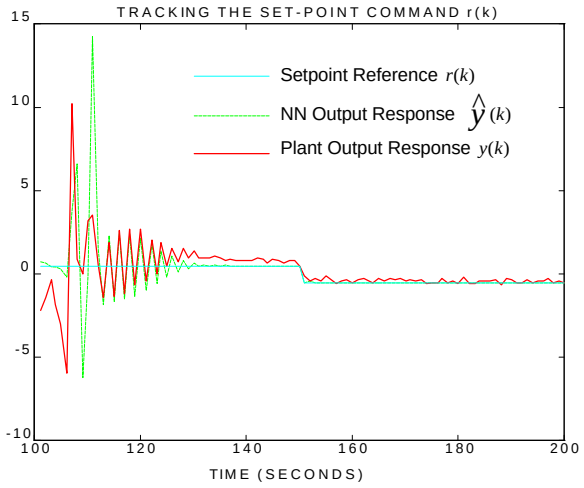


Figure 6.23 *Wavenet* Controller Responses to Set-Point Control of Output Noise Immunity

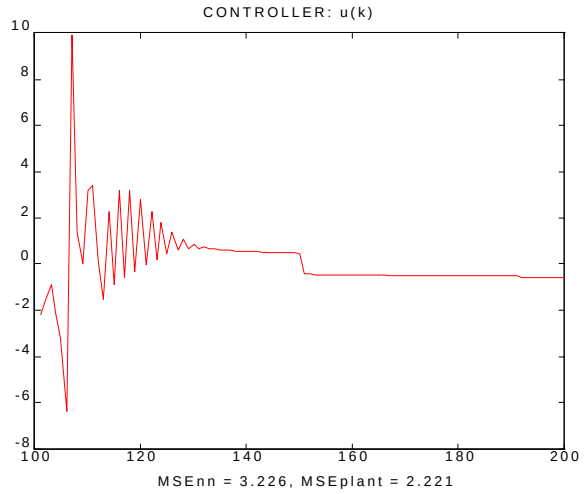


Figure 6.24 *Wavenet* Control Actions to Set-Point Control of Output Noise Immunity

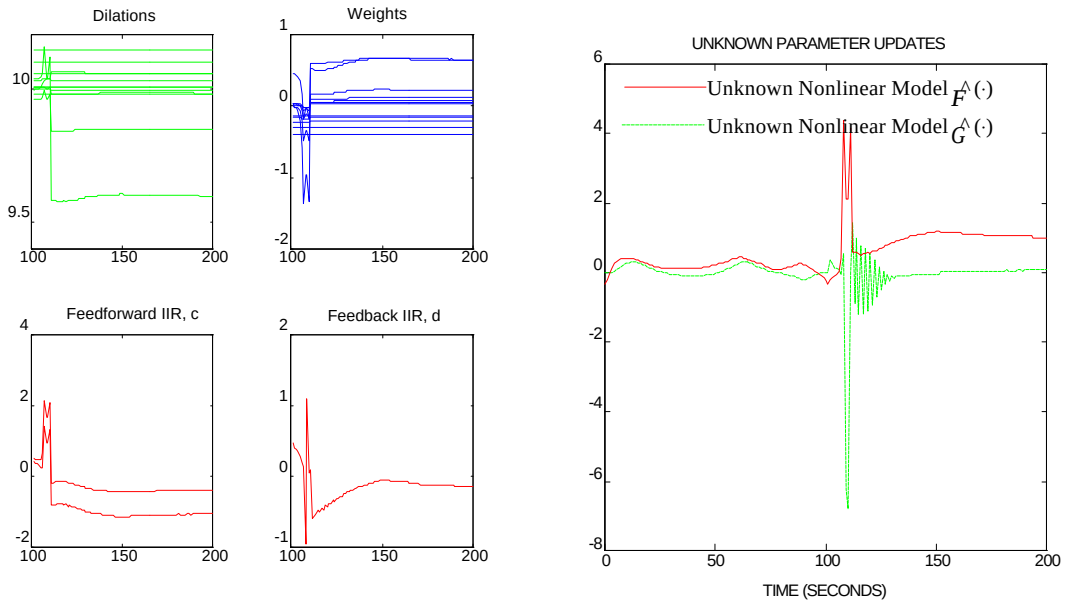


Figure 6.25 Adaptive *Wavenet* Parameters to Set-Point Control of Output Noise Immunity

6.3.3.2 Adaptive PID Controller Using *Wavenets*

Control input of Figure 6.26 is used for the set-point control simulation results in Figure 6.27 using the proposed adaptive self-tuning PID controller with self-adjustable *wavenet* parameter adaptation. Controller parameters P, I, and D were initially set at 0.2, 0.1, and 0.005, respectively. Learning rate step size for the variable P, I, and D are all fixed at 0.05. Figure 6.28 shows all the necessary parameter updates to the network and control.

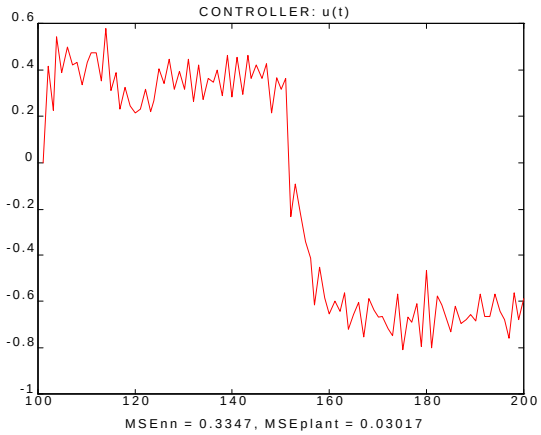


Figure 6.26 PID Control Actions to Set-Point Control of Output Noise Immunity

Again the plant response is noisy but achieve the setpoint tracking while the NN response is only used for the P, I, D adaptation processes. The MSE of the actual plant and of the *wavenet* network found with respect to the desired reference were 0.03017 and 0.3347 respectively.

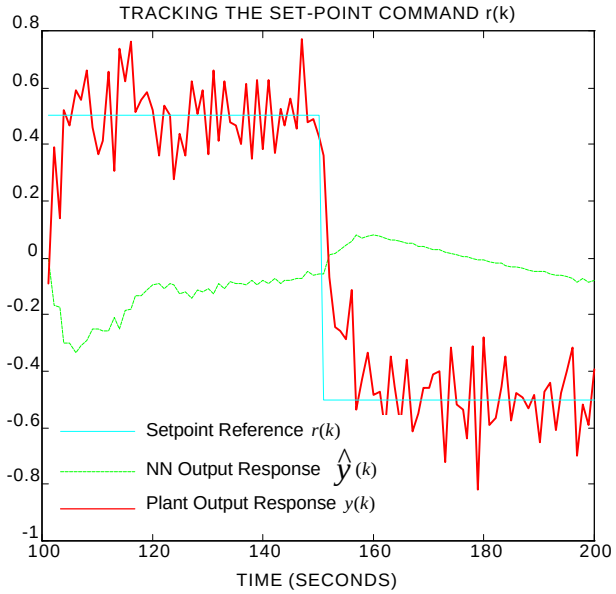
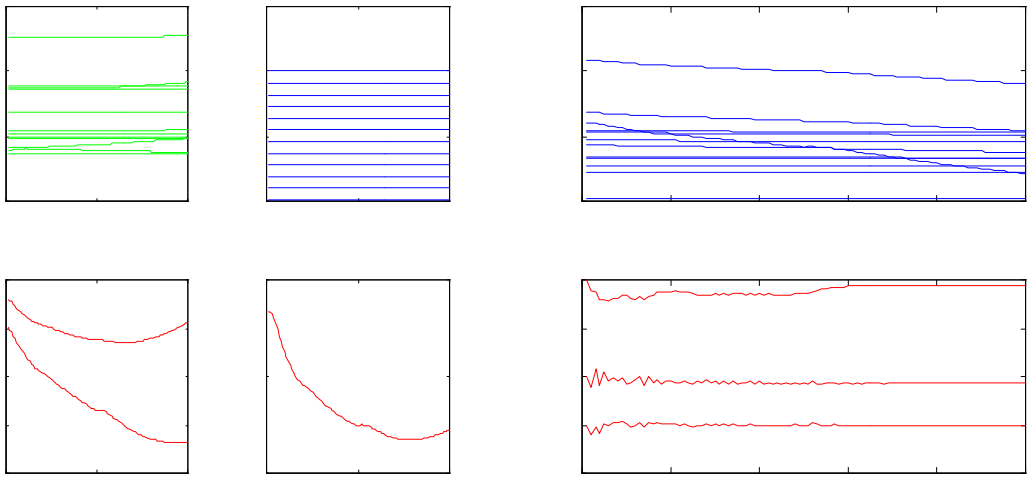


Figure 6.27 PID Controller Responses to Set-Point Control of Output Noise Immunity



(a) (b)

Figure 6.28 Adaptive PID Parameters to Set-Point Control of Output Noise Immunity

6.3.3.3 Neuro-Controller Based on the Backpropagation Algorithms

Figure 6.29 shows the result of the tracking responses to the setpoint control of the output noise immunity using the same two-layer perceptron network with BPP algorithms as before. The plant responds with MSE of 0.0567. The control $u(k)$ fed to the plant is illustrated in Figure 6.30. With this conventional algorithm, the plant response is noisy with a small offset and the NN response never adjust to the changes. Finally, the output noise immunity studies show the worst scenario in the adaptive self-tuning control.

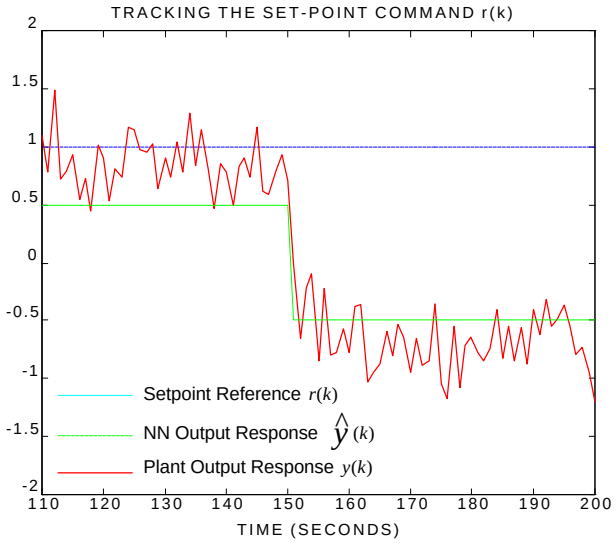


Figure 6.29 Neuro-Controller with BPP Learning Responses to Output Noise Immunity

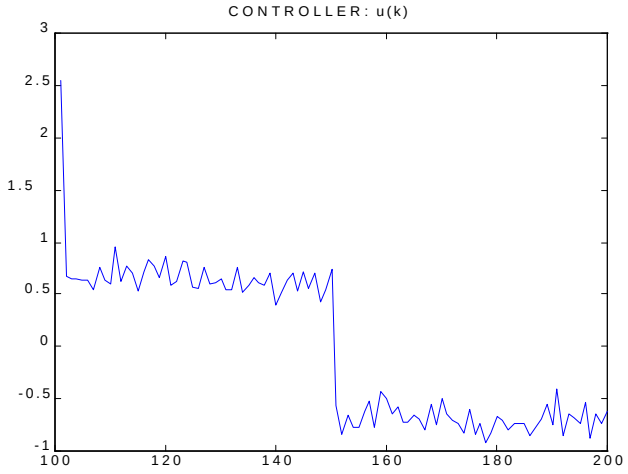


Figure 6.30 Control Actions to Set-Point Control of Output Noise Immunity

Chapter 7

Future Work

In modeling the *wavenet* identifiers, it is found that extra hidden units/mother wavelet basis functions would not necessarily improve performance, but rather may deteriorate the network's stability (by not giving the network "room" outside of the given receptive fields to generalize between neighboring training samples). The minimum number of wavelets, depends on the different types of wavelets and sizes of data. Determining the minimum number is still an open research problem. An efficient network consists of a minimal number of parameters. In this paper, the minimum number of all wavelets found in the particular identification process of 50 discrete voice data was approximately 8 wavelets. Moreover, the appropriate order of the IIR structure is worth investigation to provide minimal networks for identifying and self-tuning purposes.

In control modeling it is difficult to specify which wavelet functions are best in providing quick tracking to various uncertainty. In [LV97], we demonstrated an example comparing simulation studies of the nonlinear control systems to various frame wavelets. More research is still needed to reach absolute conclusions. The control stability of the neuro control structure is another ongoing research topic. It is known that the stability of a closed-loop system is not implied by the stability of the open loop system. This situation recurs in a more complex form in adaptive neuro control: a well behaved off-line parameter estimation algorithm can become unstable when operating on-line via a feedback controller. This could happen when the identification of the actual process contains "unmodeled high-frequency dynamics" and is thus of higher order than the control model. For example, when a rapidly varying (high frequency) control signal hits the process or when the plant dynamics are unmodeled resulting to slow response to the output process, etc. Also the size of the learning rate is a crucial factor to the stability of the *wavenet* based controller.

Chapter 8

Conclusions

This paper emphasizes self-tuning control applications of an efficient neural network architecture based on a wavelet theory called *wavenets*. The *wavenet* algorithms demonstrate a relevant network topology without initially resorting to trial-and-error methods. The *wavenet* based controllers improve the performance of the trained network for fast convergence, minimum variability between runs, robustness to noise interference, and high complex ability to learn and track of *unknown*/undefined complex systems. Three control schemes were shown. One is based on an assumed plant model and a neuro-identifier that is used to construct adaptive controllers. The second is a PID controller based on the self-tuning *wavenet* adaptations. The other is a traditional neuro-control schemes based on the feedforward neural network structure with backpropagation (BPP) algorithms. This scheme is used for a base line comparison to *wavenets*. The first two control schemes have different advantages. The PID controller is simple to configure since it does not require a process model. *Wavenet* algorithms can be easily adapted to the existing PID controller and plants which provide easy transition changes to plant modifications. The first controller requires less parameters than the PID controller which requires additional PID parameters. Simulation studies indicate similar results from both schemes, but the PID scheme tends to be more robust and less sensitive, and the first scheme tends to provide quicker tracking adjustment to control changes. Finally, the conventional scheme with BPP shows that it requires a longer time adapting to changes and performs poorly to noise added systems. From the research studies, the worst scenario to all of the control schemes, in terms of MSE, occurs when noise is contaminated to the output port of the plant systems.

In this research, significant contribution is dedicated to the algorithms and the network topology. Controlling of *unknown* complex systems is a very difficult problem, especially when the *black-box* systems are highly nonlinear and masking with interference. The author has shown that *wavenets* can overcome this problem and hopes to have contributed to setting the foundation for the more development of neural network adaptive wavelet based control system methodology.

References

[AH92] A. N. Akansu and R. A. Haddad, *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets*, Academic Press, Inc., San Diego, CA, 1992.

[BA80] R. R. Bitmead and B. D. O. Anderson, "Performance of Adaptive Estimation Algorithms in Dependent Random Environments," *IEEE Trans. Autom. Control*, vAC25, pp 788-94, 1980.

[Ber86] N. J. Bershad, "Analysis of the Normalized LMS Algorithm With Gaussian Inputs," *IEEE Trans. Acoust. Speech Signal Processing*, vASSP-34, pp 793-806, 1986.

[BF94] J. J. Benedetto and M. W. Frazier, *Wavelets: Mathematics and Applications*, CRC Press, Inc., Boca Raton, FL, 1994.

[BGLSR96] R. Benton, A. Ganjoo, B. Lumetta, D. Spillman, and J. Ring, "Adaptive Wavelet Transforms of Singular and Chaotic Signals," *Wavelet Applications III*, H. H. Szu, Ed., Proceedings SPIE- The International Society for Optical Engineering, v2762, pp. 136-43, 1996.

[Che90] F. C. Chen, "Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control," *IEEE Control Systems Magazine, Special Issue on Neural Networks for Control Systems*, pp 44-8, April 1990.

[CGT89] J. M. Combes, A. Grossmann, and Ph. Tchamitchian, Eds., *Wavelets: Time-Frequency Methods and Phase Space*, 2nd edition, Springer-Verlag, New York, NY, 1989.

[CJ94] D. S. Chen and R. C. Jain, "A Robust Back Propagation Learning Algorithm for Function Approximation, *IEEE Trans. on Neural Networks*, v5, n3, pp. 467-79, 1994.

[Cla93] P. M. Clarkson, *Optimal and Adaptive Signal Processing*, CRC Press, Inc., Boca Raton, FL, 1993.

[Coh95] L. Cohen, *Time-Frequency Analysis*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1995.

[Cor95] K. Corcella, "Market prediction Turns the Tables on Random Walk," *Wall Street & Technology*, pp 37-41, May 1995.

[CU93] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, John Wiley & Sons, Inc., New York, NY, 1993.

[CY94] P-R Chang and B-F Yeh, "Nonlinear Communication Channel Equalization Using Wavelet Neural Networks," *IEEE Transaction on Neural Networks*, pp. 923-34, 1995.

[Dau90] I. Daubechies, "The Wavelet Transform, Time-Frequency Localization and Signal Analysis," *IEEE Trans. Inform. Theory*, v36, n5, pp. 961-1005, Sept. 1990.

[DC91] H. Drucker and Y. L. Cun, "Double Backpropagation Increasing Generalization Performance," *IEEE International Joint Conference on Neural Networks* (*IJCNN*), pp. II145-150, 1991.

[DC92] M. Donlin and J. Child, “Is Neural Computing the Key to Artificial Intelligence?” *Computer Design*, pp 87-104, Oct. 1992.

[DCM92] C. Darken, J. Chang and J. Moody, “Learning Rate Schedules for Faster Stochastic Gradient Search,” *Neural Networks for Signal Processing 2--Proceedings of the 1992 IEEE Workshop*, IEEE Press, 1992.

[DPH93] J. R. Deller, Jr., J. G. Proakis, and J. H. L. Hansen, *Discrete-Time Processing of Speech Signals,* Macmillan Publishing Company, New York, NY, 1993.

[EB95] R. Eberhart and R. Brandmaier, “Overview of Biomedical Applications of Neural Networks,” *Applications and Science of Artificial Neural Networks*, S. K. Rogers and D. W. Ruck, Eds., Proc. SPIE, v2492, part2, pp. 642-50, 1995.

[Gar84] W. A. Gardner, “Learning Characteristics of Stochastic Gradient Descent Algorithms: A General Study, Analysis, and Critique,” *Signal Processing*, v6, pp 113-33, April 1984.

[GGM84] P. Goupillard, A. Grossmann, and J. Morlet, “Cycle-Octave and Related Transforms in Seismic Signal Analysis,’ *Geoexploration*, v23, pp. 85-102, 1984.

[Gib80] J. D. Gibson, “Adaptive Prediction in Speech Differential Encoding Systems,” Proc. IEEE, v68, pp 488-525, 1980.

[GM84] A. Grossmann and J. Morlet, “Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape,” *SIAM J. Math. Anal*., v15, n4, pp. 723-36, 1984.

[GS88] R. Gorman and T. Sejnowski, “Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Signals, *Neural Networks*, v1, pp 75-89, 1988.

[Hag90] M. Hagiwara, “Novel Back Propagation Algorithm for Reduction of Hidden Units and Acceleration of Convergence Using Aritificial Selection, *Proceedings ICNN*, pp. 625-30, 1990.

[Ham93] D. Hammerstrom, “Neural Networks at Work,” *IEEE Spectrum*, pp 26-32, June 1993.

[Hay91] S. Haykin, *Adaptive Filter Theory*, 2nd Ed., Prentice Hall, Inc., Englewood Cliffs, NJ, 1991.

[Hay94] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan, New York, NY, 1994.

[HB92] F. Hlawatsch and G. F. Boudreaux-Bartels, “Linear and Quadratic Time-Frequency Signal Representations,” *IEEE Signal Processing Mag*., pp 21-67, April 1992.

[HD91] S. Haykin and C. Deng, “Classification of Radar Clutter Using Neural Networks,” *IEEE Trans. Neural Networks*, v2, n6, pp 589-600, 1991.

[HH93] D. R. Hush and B. G. Horne, “Progress in Supervised Neural Networks: What’s New Since Lippmann?,” *IEEE Signal Proc. Mag*., pp 8-39, Jan. 1993.

[HM84] M. L. Honig and D. G. Messerschmitt, *Adaptive Filters: Structures, Algorithms, and Applications*, Kluwer Acadamic Publishers, Hingham, MA, 1984.

[HSW89] K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Network and Univerdal Approximators," *Neural Networks*, v2, pp 359-66, 1989.

[IRR95] S. A. Imhoff, D. Y. Roeum, and M. R. Rosiek, "New Classes of Frame Wavelets for Applications," *Wavelet Applications II*, H. H. Szu, Ed., Proceedings SPIE- The International Society for Optical Engineering, v2491, pp. 923-34, 1995.

[JK94] I. Jouny, M. Kanapathipillai, "Neural Network Adaptive Wavelet Classification of Radar Targets," *Proceedings of the 1994 International Geoscience and Remote Sensing Symposium*, v4, pp 1889-91, 1994.

[Joh95] R. C. Johnson, "New Techniques Rewrite Pen Computing," *Electronic engineering Times*, pp 42-44, May 22, 1995.

[JS94] B. Jawerth and W. Sweldens, "An Overview of Wavelet Based Multiresolution Analyses," *SIAM Review*, v36, n3, pp. 377-412, Sept. 1994.

[Kai94] G. Kaiser, *A Friendly Guide to Wavelets*, Birkhäuser, Boston, MA, 1994

[KB92] S. Kadambe and F. Boudreaux-Bartels, "A Comparison of the Existence of 'Cross Terms' in the Wigner Distribution and the Squared Magnitude of the Wavelet Transform and the Short Time Fourier Transform," *IEEE Trans. Signal Proc*., v40, n10, pp. 2498-2517, Oct. 1992.

[KL92] C-C Ku and K. Y. Lee, "System Identification and Control Using Diagonal Recurrent Networks," *Proceedings of American Control. Conference*, pp 545-49, June 1992.

[KS94a] S. Kadambe and P. Srinivasan, "Application of Adaptive Wavelets for Speech Coding," *Proceedings of the IEEE-SP International Symposium of Time-Frequency and Time-Scale Analysis*, pp. 632-35, 1994.

[KS94b] S. Kadambe and P. Srinivasan, "Applications of Adaptive Wavelets for Speech," *Optical Engineering*, v33, n7, pp 2204-11, 1994.

[KS94c] S. Kadambe and P. Srinivasan, "Text Independent Speaker Identification System Based on Adaptive Wavelets," *Wavelet Applications*, H. H. Szu, Ed., Proceedings SPIE- The International Society for Optical Engineering, v2242, pp 669-77, 1994.

[KS95] S. Kadambe and P. Srinivasan, "A Low Bit Rate Speech Coder Based On Gaussian Adaptive Wavelets," *Wavelet Applications II*, H. H. Szu, Ed., Proceedings SPIE- The International Society for Optical Engineering, v2491, part 1, pp. 461-72, 1995.

[KSS91] A. Karakasoglu, S. I. Sudharsanan, and M. K. Sundareshan, "Nerual Network-Based Identification and Adaptive Control of Nonlinear Systems: A Novel Dynamical Network Architecture and Training Policy," Proceedings of the 30th Conference on Decision and Control, v1, pp 180-1, December 1991.

[KT94] K. Kobayashi and T. Torioka, “A Wavelet Neural Network for Function Approximation and Network Optimization,” *Intelligent Engineering Systems Through Artificial Neural Networks, Volume 4*, C. H. Dagli, B. R. Fernandez, J. Ghosh, and R. T. Soundar Kumara, Eds., Proceedings of the Artificial Neural Networks in Engineering (ANNIE ‘94) Conference, pp. 505-10, 1994.

[LAP93] P. J. Loughlin, L. E. Atlas, and J. W. Pitton, “Advanced Time-Frequency Representations for Speech Processing,” *Visual Representations of Speech Signals*, M. Cooke, S. Beet, and M. Crawford, Eds., John Wiley & Sons, Ltd., NY, 1993.

[Lip87] R. Lippmann, “An Introduction to Computing with Neural Networks,” *IEEE ASSP Mag.*, pp 4-22, April 1987.

[LN95] A. U. Levin and K. S. Narendra, “Control of Nonlinear Dynamical Systems Using Nerual Networks, Part II: Observability, Identification and Control,” *IEEE Transactions on Neural Networks*, 1995.

[LS89] W. Li and J. J. E. Slotine, “Neural Network Control of Unknown Nonlinear Systems,” Proc. 1989 American Control Conference, pp 1136-41, 1989.

[LV97] G. Lekutai, and H. VanLandingham, “Self-Tuning Control of Nonlinear Systems Using Neural Network Adaptive Frame Wavelets,” to be presented to *1997 IEEE Int. Conf. on System, Man, and Cybernatics* in Orlando, FL.

[MA90] H. Meyr and G. Ascheid, *Synchronization in Digital Communications: Volume1 Phase-, Frequency-Locked Loops and Amplitude Control*, John Wiley & Sons, Inc., New York, NY, 1990.

[MAL91] C. A. Mead, X. Arreguit, and J. Lazzaro, “Analog VLSI Model of Binaural Hearing,” *IEEE Trans. Neural Networks*, v2, pp 232-36, 1991.

[Mey93] Y. Meyer, Wavelets: Algorithms & Applications, Translated by R. D. Ryan, *SIAM*, Philadelphia, PA, 1993.

[MCN96] P. Mars, J. R. Chen, and R. Nambiar, *Learning Algorithms*, CRC Press, Inc., Boca Raton, FL, 1996.

[MFV96] J. F. Marar, E. C. B. C. Filho, and G. C. Vasconcelos, “Functions Approximation by Polynomial Wavelets Generated From Powers of Sigmoids,” *Wavelet Applications III*, H. H. Szu, Ed., Proceedings SPIE-The International Society for Optical Engineering, v2762, pp. 365-74, 1996.

[MMG87] R. K. Martinet, J. Morlet, and A. Grossmann, “Analysis of Sound Patterns Through Wavelet Transforms,” *International J. of Pattern Recogn. and Artificial Intellig.*, v1, n2, pp. 273-302, 1987.

[MR90] B. Muller and J. Reinhardt, *Neural Networks*, Spring Verlag, Berlin, 1990.

[NBK95] K. S. Narendra, J. Balakrishnan, and M. K. Ciliz, “Adaptation and Learning Using Multiple Models, Switching, and Tuning,” IEEE Control Systems Magazine, v15, n3, pp 37-51, June 1995.

[NP90] K. S. Narendra and K. Parthasarathy, “Identification and Control of Dynamical Systems Using Neural Network,” *IEEE Transactions on Neural Networks*, v1, n1, pp 4-27, March 1990.

[New93] D. E. Newland, *An Introduction to Random Vibrations, Spectral & Wavelet Analysis*, John Wiley & Sons, Inc., New York, NY, 1993.

[New95] H. P. Newquist, "Smart Cards," *AI Expert*, pp 44-5, Jan. 1995.

[Oma95] S. Omatu, "Dynamical Systems Regulation by Neuro-Controllers," *Applications and Science of Artificial Neural Networks*, S. K. Rogers and D. W Ruck, Editors, Proceedings SPIE, v2492, pp 590-9, 1995.

[PK93] Y. Pati and P. Krishnaprasad, "Analysis and Synthesis of Feedforward Neural Networks Using Discrete Affine Wavelet Transformation," *IEEE Trans. Neural Networks*, v4, n1, pp. 73-85, Jan. 1993.

[Pro95] J. G. Proakis, *Digital Communications*, 3rd Ed., McGraw-Hill, New York, NY, 1995.

[PSY88] D. Psaltis, A. Sideris, and A. A. Yamamura, "A Multilayered Neural Network Controller," *IEEE Control Systems Magazine*, pp 17-21, April 1988.

[PV90] I. Pitas and A. N. Venetsanopoulos, *Nonlinear Digital Filters: Principles and Applicaitons*, Kluwer Acadamic Publishers, Norwell, MA, 1990.

[Res91] H. L. Resnikoff, "Wavelets and Adaptive Signal Processing," *Adaptive Signal Processing*, S. Haykin, Ed., Proceedings SPIE, v1565, pp. 370-82, 1991.

[Ril89] M. D. Riley, *Speech Time-Frequency Representations*, Kluwer Acadamic Publishers, Norwell, MA, 1989.

[RS90] S. Roy and J. J. Shynk, "Analysis of the Momentum LMS Algorithm," *IEEE Trans. on Acoust., Speech, Signal Processing*, vASSP-38, pp 2088-98, 1990.

[RV91] O. Rioul and M. Vetterli, "Wavelets and Signal Processing," *IEEE Signal Processing Mag.*, v8, n4, pp 14-38, Oct. 1991.

[SMA88] W. A. Sethares, I. M. Y. Mareels, B. D. O. Anderson, E. R. Johnson, and R. R. Bitmead, "Excitation Conditions for Signal Regressor Least Mean-Squares Adaptation," *IEEE Circuits Systems*, vCAS-35, pp. 613-24, 1988.

[SR87] T. Sejnowski and C. Rosenberg, "Parallel Networks That Learn to Pronounce English Text," *Complex Systems*, v1. pp 145-168, 1987.

[SRS92] Y. Sheng, D. Roberge, and H. H. Szu, "Optical Wavelet Transform," *Optical Engr.*, v31, n9, pp. 1840-45, Sept. 1992.

[SS91] S. I. Sudharsanan and M. k. Sundareshan, "Training of A Three-Layer Dynamical recurrent Neural Network for Nonlinear Input-Output Mapping," Proc. Int. Conf. on Nerual Networks-- *IEEE IJCNN*, v2 pp II-111-115, 1991.

[STK92] H. H. Szu, B. A. Telfer, and S. Kadambe, "Neural Network Adaptive Wavelets for Signal Representation and Classificaiton," *Optical Enginering*, v31, n9, pp 1907-16, Sept. 1992.

[TSD94] B. A. Telfer, H. H. Szu, and G. J. Dobeck, "Adaptive Wavelet Classification of Acoustic Backscatter," *Wavelet Applications*, H. H. Szu, Ed., Proceedings SPIE- The International Society for Optical Engineering, v2242, pp 661-68, 1994.

[TSD95] B. A. Telfer, H. H. Szu, and G. J. Dobeck, “Adaptive Time-Frequency Classification of Acoustic Backscatter,” *Wavelet Applications II*, H. H. Szu, Ed., Proceedings SPIE- The International Society for Optical Engineering, v2491, part 1, pp 451-60, 1995.

[Wei94] L. G. Weiss, “Wavelets and Wideband Correlation Processing,” *IEEE Signal Processing Mag.*, v11, n1, pp 13-32, Jan. 1994.

[Wel94] S. T. Welstead, *Neural Network and Fuzzy Logic Applications in C/C++*, John Wiley & Sons, Inc., New York, NY, 1994.

[Wil94] C. Wilson, “New Wireless System Offers Data Alternative (Multipoint Networks Inc.’s WaveNET Wireless Systems),” *Telephony*, v226, n1, p. 9, Jan. 3, 1994.

[WS85] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1985.

[WWZZ95] Jin Wang, Fuli Wang, Jinliang Zhang, and Jin Zhang, “Intelligent Controller Using Neural Network,” *Intelligent Manufacturing*, S-Z. Yang, J. Zhou, and C-G Li, Editors, Proceedings SPIE, v2620, pp 755-61, 1995.

[YL93] X. Ye and N. K. Loh, “Dynamic System Identification Using Recurrent Radial Basis Function Network,” *Proceedings of American Control. Conference*, v3., pp. 2912-16, June 1993.

[You93] R. K. Young, *Wavelet Theory and Its Applications*, Kluwer Acadamic Publishers, Boston, MA, 1993.

[Zay93] A. I. Zayed, *Advanced in Shannon’s Sampling Theory*, CRC Press, Inc., Boca Raton, FL, 1993.

[ZB92] Q. Zhang and A. Benveniste, “Wavelet Networks,” *IEEE Trans. Neural Networks*, v3, n6, pp 889-98, Nov. 1992.

Vita

Gaviphat Lekutai was born in Thailand on March 22, 1970. He came to the United States in 1987, and graduated from Verona High School in 1988. He graduated cum laude from West Virginia Institute of Technology with a Bachelor of Science degree in Electrical Engineering in 1991. He continued to the graduate program at Virginia Polytechnic Institute and State University and completed the Master of Science and the Doctor of Philosophy degrees in 1993 and 1997, respectively. He joined CellularOne of Washington/Baltimore as a RF engineer in July, 1996.

His current interests include neural computing, adaptive signal processing, cellular communications, control systems, and wavelets.

Mr. Lekutai was recognized by the Who's Who Among Students in America Universities & Colleges in 1991. He is also a member of the IEEE, SIAM, and Eta Kappa Nu. He enjoys listening to jazz music and playing basketball.