APPLICATION OF VARIABLE STRUCTURE SYSTEMS THEORY FOR TRAINING OF INTELLIGENT SYSTEMS

by

Uğur Yıldıran

B. S. in Control and Computer Engineering, Istanbul Technical University, 1999

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Systems and Control Engineering

Boğaziçi University

2001

ACKNOWLEDGMENTS

I would like to thank to my thesis supervisor, Prof. Dr. Okyay Kaynak for his sincere guidance, endless support and motivation. It would be so difficult to accomplish this study without his effort.

I am grateful to Assoc. Prof. H. Levent Akın and Prof. Fikret Gürgen for reading my thesis in a short period of time and being member of my thesis jury.

I would also like to thank Assoc. Prof. Yağmur Denizhan for everything she thought me during my studies in Boğaziçi University.

Thanks to Assoc. Prof. Feza Kerestecioğlu for his careful evaluation of this study.

I would like to express my thanks to Dr. M. Önder Efe for his unbelievable support and motivation. His guidance has lead to success of this thesis.

In addition, I would like to thank Serdar İplikçi for his supports during my thesis study. He shared too much with me.

I would like to express my gratitude to Nur Başürün, Metin Yıldırım, Hasip Bulut, Gülay Öke, Ersin Göse, M. Ali Ergin, Yağız Sütçü, Ayşegül Ergin and Alper Özpınar.

ABSTRACT

APPLICATION OF VARIABLE STRUCTURE SYSTEMS THEORY FOR TRAINING OF INTELLIGENT SYSTEMS

Soft computing architectures with their extensive flexibility and strong mapping capabilities have been widely used for control of nonlinear systems. In this regard, error backpropogation and its derivatives have been the most popular and frequently employed schemes for parameter adjustment of these architectures. However, these schemes bring some serious problems together, like instability of closed loop system and sensitivity to uncertainties, which must be carefully addressed by a system designer. In order to alleviate these problems, recently, Efe has proposed a control strategy in which parameters of intelligent controllers are updated by a continuous-time robust parameters adjustment mechanism in order to robustify and stabilize the closed loop system dynamics. The results obtained for a two link SCARA robot in this study show that the proposed method is successful in achieving the control objectives.

In this thesis, the methodology proposed by Efe is investigated for first order nonlinear systems. Based on the results, it has been observed that the time evolution of input-output curves of different structures show similar characteristics. Moreover, a modification is proposed for update mechanism of all architectures in order to prevent unbounded parameter evolution problem which occurs in the original algorithm. Lastly, based on the results for different systems, it has been concluded that the Adaptive Linear Element is the most suitable architecture for the control systems investigated because of its simplicity.

ÖZET

DEĞİŞKEN YAPILI SİSTEMLER KURAMININ AKILLI SİSTEMLERİN EĞİTİMİNE UYGULANMASI

İşlemsel zeka içeren sistemler sahip oldukları esneklik ve doğrusal omayan fonksiyonları gerçekleyebilme özellikleri ile doğrusal olmayan sistemlerin kontrollünde geniş bir uygulama alanı bulmuştur. Bu bağlamda hata geri yayma yöntemi ve onun türevleri en çok kullanılan eğitim algoritmaları olmuştur. Fakat bu yöntemler kapalı çevrim sistemin kararsızlığı ve parametrelerin sınırlı tutulamaması gibi problemlerden dolayı pratik ugulamalarda sistem tasarımcısı tarafından dikkate alınması gereken bazı sorunlara yol açmaktadırlar. Bu sorunların üstesinden gelebilmek için Efe denetleyici parametrelerini zamanda sürekli gürbüz bir mekanizma ile güncelleyen bir yöntem önermiştir. İki serbestlik dereceli SCARA tipi robot modeli üzerindeki çalışmalar önerilen yöntemin başarılı sonuçlar verdiğini göstermiştir.

Bu tezde, Efe tarafından önerilen metod birinci dereceden doğrusal olmayan sistemler için incelenmiştir. Elde edilen sonuçlar üzerinde yapılan incelemelere dayanarak değişik akıllı yapıların giriş-çıkış eğrilerinin zaman içindeki davranışının benzer olduğu görülmüştür. Ayrıca asıl algoritmada oluşan sınırsız parametre genişlemesi problemini önlemek için parametre güncelleme mekanizmasında bir değişiklik önerilmiştir. Son olarak, elde edilen benzetim sonuçlarına dayanarak uygulamalar için en iyi sistemin "Adaptive Linear Element" olduğu sonucuna varılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$\underline{\varphi}$	Hidden layer output vector of GRBFNN
μ_{ij}	Center of activation or membership functions
σ_{ij}	Width of activation or membership functions
f_i	Consequence of i^{th} rule of ANFIS
G	Gain matrix for sliding surface
K	Uncertainty bound
N_w	Upper bound for norm of adjustable parameter vector of an intelligent structure
N_x	Upper bound for norm of input vector of an intelligent structure
$N_{\dot{x}}$	Upper bound for norm of derivative of input vector of an intelligent structure
$N_{\dot{y}d}$	Upper bound for absolute value of time derivative of desired output
s_c	Sliding surface at the output of the controller
s_p	Sliding surface at the output of the plant
$\underline{u}$	Firing strength vector
$\underline{u}^n$	Normalized firing strength vector
$\underline{w}$	Adjustable parameters of an intelligent architecture
$\underline{x}$	Input of an intelligent architecture
y	Output of an intelligent architecture

ADALINE	Adaptive linear elements
ANFIS	Adaptive neuro-fuzzy inference systems
ANN	Artificial neural network
FL	Fuzzy logic
GRBFNN	Gaussian radial basis function neural network
SFS	Standard fuzzy systems
SMC	Sliding mode control
VSS	Variable structure systems

1. INTRODUCTION

Although, classical methods have been successfully applied to variety of engineering disciplines so far, they are not adequate to satisfy high performance requirements of today's complex systems. This stems from the fact that they utilize precise model based paradigms, which highly rely on the mathematical relations regarding the internal structure of the system in hand. In contrast to that, a novel approach to computation, soft computing, offers a set of methodologies, which can be exploited together to incorporate different types of knowledge about a system. These methodologies are especially characterized by information granulation, perception and adaptability to changes in the environment.

The basic constituents of soft computing are artificial neural networks (ANN), fuzzy logic (FL) and genetic algorithms (GA) [1]. NNs are comprised of several nodes, which are massively connected by set of synaptic weights that store the information from the past experiences of the network. Similar to those of biological neural networks, artificial neural networks have ability to learn from interactions with environment, generalize the information and adapt to changes in data. The second component of soft computing is FL. FL allows the designers to express the verbal knowledge in mathematical form by means of membership functions. This is especially useful for mid-level applications, which provide an interface between high-level supervisory systems and low-level components, relying on basic mathematical methods. GA is the last constituent of the soft computing. GA mimics the evaluation of living organisms to find optimal solutions in a parallel fashion. This approach eliminates some limitations of classical optimization methods like converging to local minimas and need for a heuristic knowledge.

One of the most important characteristics of intelligent systems is their ability to learn from examples. This procedure includes storing information gained by training in an appropriate form. In soft computing architectures, the information is stored in form of adjustable parameters of nonlinear input-output mappings. Thus, learning in these systems can be boiled down to a parameter optimization problem in which a parameters set that best fits to given samples are to be found. There exist different training algorithms in the literature for this purpose. But most of them suffer from parameter convergence problem,

that is, reaching to optimal parameter set may take too long or even it may not be possible to attain the optimal set. These limitations cause serious problems in control applications as will be discussed in below.

When a practical application is of primary concern, a control system must possess some degree of robustness, that is, it must be insensitive to uncertainties, which are inevitable in real applications. Variable structure systems theory (VSS) offers a solution to robustness problem by means of employing discontinuous control actions in different regions of the state space of the system. Especially, if these discontinuities are introduced deliberately system trajectories can be forced towards an asymptotically stable manifold in the state space of the system and they may keep staying on the manifold after a finite reaching time even in the presence of uncertainties [2]. Inspired from the motion of the trajectories on the manifold, this type of systems is referred to sliding mode control (SMC) systems in the related literature. The novel properties of SMC systems may provide a useful framework for training of intelligent structures because they allow finite convergence periods in first order systems and it is possible to utilize simplified models provided that the necessary control gain is available.

Soft computing architectures with their extensive flexibility stipulated by large number adjustable parameters are very adequate for control applications which possess time varying characteristics and uncertainties. In the relevant literature, there exist several approaches, which utilize soft computing architectures for different purposes such as direct inverse controllers [3], adaptive controllers [4] and system identification models [5]. Although, satisfactory results have been attained in these studies, in general, they are based on empirical results, that is, there is no stability guarantee and in some cases even it is possible to find a set of conditions which drives the system into unstable regimes [6]. Moreover, these approaches require a discrete-time model of the system in hand. Thus, in applications to continuous-time systems, it is necessary to employ discretization methods which leads to inexact representations due to unavoidable approximations. These undesirable properties of the available methods can be accounted to fact that they rely on gradient based learning strategies, which have parametric convergence problems as discussed before. To somehow eliminate these difficulties Ramirez, et al. [7] proposed a method, which incorporates SMC strategies into training of intelligent architectures. In this

work, they suggested a parameters update mechanism for adaptive linear element (ADALINE) networks which ensures the occurrence of the zero learning-error level at the output of the network in finite time and applied this strategy to inverse dynamic identification of a Kapitsa pendulum. Yu et al. [8] further developed results in [7] by introducing adaptive bound dynamics. A different approach, which also exploits SMC strategy, is introduced by Parma et al. [9] for training of feed forward neural networks (FNN) with nonlinear activation functions at the output layer neurons.

Recently, the idea of incorporating continuous time robust learning mechanisms for control of nonlinear systems has been proposed by Efe [10]. In his work, the learning algorithm in [7] is used to train different flexible structures having linear activation functions at their output layers. As the error measure, which is necessary to train and evaluate performance of the intelligent controller, sliding line at the output of the plant is utilized. In this thesis, the control methodology given in [10] is investigated for systems modeled by first order and time varying nonlinear ordinary differential equations. Although, the plants, which are chosen as test beds, are not first order ones, they are divided into first order sub-systems by matching an appropriate key state with each input variable. Then each sub system is controlled separately by assuming the couplings between them as disturbances.

This thesis is organized as follows. In the second section, the basic concepts of fuzzy logic and artificial neural networks are introduced within the context of the intelligent architectures utilized throughout the work. A general SMC design technique for a class of nonlinear systems is discussed in the third section. The fourth section explains the philosophy of the SMC based learning algorithm given in [7] and how it can be incorporated into control of first order nonlinear dynamical systems. A method to prevent parameter drift problem, which occurs in above mentioned control scheme, is also introduced in this section. In the fifth section, test plants are introduced and performance of different architectures are investigated based on the simulation results. The last section is the conclusions section where the results are discussed.

2. SOFT COMPUTING

Technological revolutions of the last century have brought upon us new problems that we have not met or considered before. In this regard, as coverage and complexity of new applications increase, the classical methods get far away from providing a suitable framework. Precise nature of classical mathematics is not adequate for real world problems not only because it may not deal with ambiguity but also solutions it provides may turn out to be too complicated for practical realization. Moreover, large-scale systems with a large number of components require higher-level techniques involving some degree of intelligence. One may think that classical artificial intelligence (AI), which uses predicate calculus and manipulates on symbols to reason and draw conclusions, may offer a convenient high-level approach, but, AI based schemes have serious limitations in dealing with some applications like computer vision, speech recognition, handwriting recognition, image understanding, multimedia database search, motion planning, common-sense reasoning, management of uncertainty and other fields which relate to machine intelligence [11].

In the face of difficulties mentioned above researchers are turned away from conventional techniques with the hope to find more suitable methods. Soft computing methodologies, first introduced in the early second half of the 20^{th} century, are an outcome of the efforts in this direction. What makes soft computing so appealing, opposite to hard computing, is its ability to exploit tolerance to uncertainty, imprecision and partial truth to achieve tractability, low cost solutions and robustness [11]. These properties of the soft computing can be accounted to the fact that it is made up by different constituents which are complementary rather than being competitive. This makes it a synergic methodology which allows the designers to exploit all available knowledge regarding to problem in hand. The main constituents of soft computing are artificial neural networks (ANN), fuzzy logic (FL) and genetic algorithms (GA). Artificial neural networks are biologically inspired intelligent structures which can adapt to changes in the environment and learn from the available data samples. They are used in a variety of areas including machine vision, pattern recognition, automatic control systems and speech processing. Soft computing borrows its ability to deal with imprecision and uncertainty from fuzzy logic. Fuzzy logic

is an approach which provides a systematic calculus to deal with incomplete and imprecise information linguistically. At this juncture, it may be appropriate to emphasize the difference between fuzzy logic and conventional AI. Conventional AI can express the information in symbolic form while excluding numeric approaches. This makes it unsuitable for low-level intelligent applications like speech processing which highly relies on mathematical calculations. On the other hand, while it provides knowledge representation in the linguistic domain, fuzzy logic also performs numerical computations by means of linguistic labels stipulated by membership functions. With this property, fuzzy logic provides an interface between mathematical knowledge and symbolic representation. The third tool of soft computing is genetic algorithms. Genetic algorithm has origins from the evolution of living systems. It tries to mimic the evaluation in the nature by utilizing basic rules of molecular biology. By this property, while genetic algorithm searches in different directions not to stuck at local minimas, it may not able to reach global optimal solutions. However, when the search space of the problem is too large for exhaustive search algorithms and it is hard to obtain a heuristic knowledge about the system in hand, it is necessary to employ more appropriate techniques which can produce results in reasonable time scales. At that point, genetic algorithm with its systematic random search methods, offers solutions in acceptable time intervals although resulting solutions may be sub-optimal ones.

Today, with the availability of high-speed computers, it is possible to realize soft computing components which require extensive computations to find rules and regularity in data. Each component has different application areas related with its capabilities. Capabilities of different soft computing structures along with those of control theory and artificial intelligent is given in Table 2.1 [12]. As can be seen from the table, although, none of these schemes is exhaustive, if they can be used together, they may cover a wide range of spectrum. In the light of this fact, recently, there have been many attempts to unify intelligent structures in favor of constructing more powerful methodologies. One of the consequences in this direction is neuro-fuzzy systems. As its name implies, a neuro-fuzzy system is a unification of both fuzzy logic and neural network approaches. It can utilize adaptive network representation to use well established optimization techniques of neural networks while it can incorporate the expert knowledge to reduce training time of an intelligent structure. Successful applications of neuro-fuzzy systems have proven the

complementary nature of the constituents of soft computing and encouraged researchers for further unification of soft computing framework. In this regard, we are beginning to see fuzzy-genetic, neuro-genetic and neuro-fuzzy-genetic systems. It seems that, in the future, soft computing will have a great impact on the ways in which intelligent systems are designed and build [11].

Table 2.1. Comparison of capabilities of different methodologies

	Mathematical Model	**Learning Data**	**Operator Knowledge**	**Real Time**	**Knowledge Representation**	**Non-linearity**	**Optimization**
Control Theory	Good or suitable	Unsuitable	Needs other methods	Good or suitable	Unsuitable	Unsuitable	Unsuitable
Neural Network	Unsuitable	Good or suitable	Unsuitable	Good or suitable	Unsuitable	Good or suitable	Fair
Fuzzy Logic	Fair	Unsuitable	Good or suitable	Good or suitable	Needs other methods	Good or suitable	Unsuitable
Artificial Intelligence	Needs other methods	Unsuitable	Good or suitable	Unsuitable	Good or suitable	Needs other methods	Unsuitable
Genetic Algorithms	Unsuitable	Good or suitable	Unsuitable	Needs other methods	Unsuitable	Good or suitable	Good or suitable

In the rest of this section, neural networks and fuzzy logic will be introduced and four different soft computing architectures will be investigated.

2.1. Artificial Neural Networks

Although, being very effective, intelligence in natural form is also so complicated that we are still far away from implementing efficient systems which can mimic most simple tasks that a human can perform in its daily life. What lies behind the capabilities of a human is its nervous system. Although, a neuron, the building block of nervous system, is in the order of six magnitude slower than the silicon circuits used in computers, its massively parallel structure makes the nervous system much more efficient than the so called intelligent machines that we can build today.

One of the first attempts to explain the complex internal representation of nervous system was the pioneering work of McCulloch and Pits [13] in which they proposed a mathematical model for a neuron. Since then, there has been a great interest in area of artificial neural networks. Especially with the development of error back propagation algorithm for training of feedforward networks, most of the capabilities of ANNs are understood better. Among these capabilities, learning constitutes the most important one. A NN can utilize input samples with their corresponding outputs to store characteristics of data in form of weights connecting network layers, which are composed of several neurons. While this form of internal representation allows it to adapt the changes in data by means of updating synaptic weights, its redundant structure also makes ANNs fault tolerant. Moreover, after a learning phase, a ANN can generalize the information given in the form of data samples by generating reasonable outputs to inputs other than those included in the training set. This property is especially very useful for functional interpolation and statistical pattern recognition applications.

The Figure 2.1 depicts the basic model for a neuron. Here the inputs of a neuron, including the bias input which can be chosen as 1, are multiplied with synaptic weights and then passed through a summing junction to calculate the soma potential of the neuron. At the last step, an activation function is utilized to attain the neuron output. This procedure can be summarized by the following equations for a neuron with n external inputs and a bias input.

$$v = \sum_{i=0}^{n} w_i x_i \tag{2.1}$$

$$y = \varphi(v) \tag{2.2}$$

where w_i s are the weights corresponding to inputs x_i s. Here the bias input and its weight are represented as x_0 and w_0 respectively.

In the following subsections two types of ANNs, which will be used throughout the thesis, are investigated.

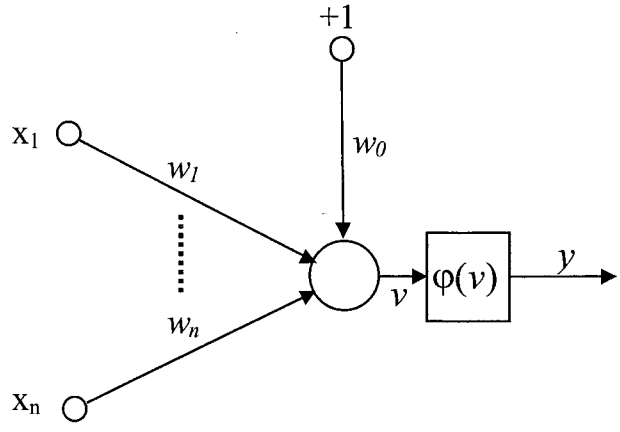

Figure 2.1. Model of a neuron

2.1.1. Adaptive Linear Element

Adaptive Linear Element (ADALINE) with its simple structure is one of the most popular soft computing architectures. It has many application areas ranging from adaptive noise cancellation to adaptive inverse control because it can be realized easily both in hardware and software. This structure is shown in Figure 2.2.

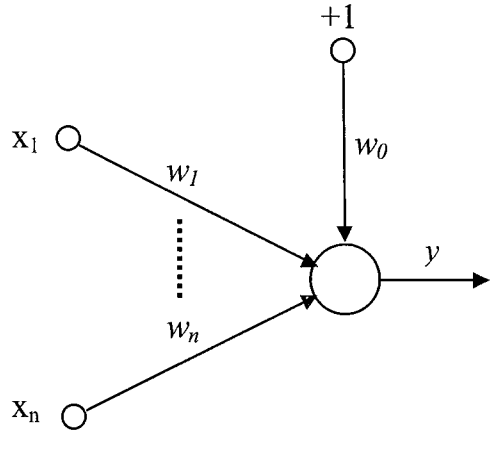

Figure 2.2. ADALINE network

As depicted in the figure, this structure has n+1 inputs and one output. The first input signal is used for bias and chosen as +1. The others are external input signals. Each input is multiplied with the weight associated with it. Then all of the weighted input signals are

summed to attain overall output of the architecture. In mathematical terms an ADALINE can be represented as follows.

$$y = \underline{w}^T \underline{x} \tag{2.3}$$

where

$$\underline{w} = [w_0 \; w_1 \; w_2 \; \ldots \; w_n]^T \tag{2.4}$$

$$\underline{x} = [1 \; x_1 \; x_2 \; \ldots \; x_n]^T \tag{2.5}$$

In these equations, $\underline{x}$ represents the augmented input vector, which is composed of external inputs and the bias input. The weight vector $\underline{w}$ is composed of the weights associated with input vector $\underline{x}$.

2.1.2. Gaussian Radial Basis Function Neural Networks

Radial Basis Function Neural Networks (RBFNN) have originated from exact interpolation theory which requires exact mapping of every data point in a training set to the corresponding target output. That is why learning in RBFNNs can be boiled down to a curve fitting problem which deals with finding a curve in multidimensional space that provides a best fit to a given data set. A RBFNN has a three-layered simple structure which eases its mathematical analysis. Input layer provides a connection between the network and its environment. The second layer, hidden layer, performs a nonlinear mapping between the input space and the high dimensional hidden space. The last layer, output layer, produces an output depending on the signal levels at the outputs of the hidden layer neurons. In general, activation function of the output layer is chosen as a linear function for the sake of simplicity. Structure of a RBFNN with n inputs and single output is shown in Figure 2.3.

In this structure, there are r hidden neurons providing a nonlinear mapping between the input space and the hidden space. Potential of each hidden neuron is determined by the

distance between the input vector and the prototype vector of the neuron. Output of a neuron is obtained by passing the potential of the neuron through a nonlinear activation function. The activation function may take different forms such as Gaussian, thin-plate spline, cubic and linear functions [14]. However, Gaussian is the most widely used activation function not only because it allows localization of the input space but also it has useful analytical properties. RBFNN having Gaussian activation function is referred to Gaussian Radial Basis Function Neural Network (GRBFNN) in the related literature. Mathematical representation of a GRBFNN can be given as follows.

$$y = \underline{w}^T \underline{\varphi}(\underline{x}) \tag{2.6}$$

where

$$\underline{w} = [w_1 \; w_2 \; \ldots \; w_r]^T \tag{2.7}$$

$$\underline{\varphi} = [\varphi_1(\underline{x}) \; \varphi_2(\underline{x}) \; \ldots \; \varphi_r(\underline{x})]^T \tag{2.8}$$

$$\underline{x} = [x_1 \; x_2 \; \ldots \; x_n]^T \tag{2.9}$$

$$\varphi_j(\underline{x}) = \prod_{i=1}^{n} \exp\left\{ -\frac{(x_i - \mu_{ji})^2}{\sigma_{ji}} \right\} \tag{2.10}$$

In the equations given above each element of $\underline{\varphi}(\underline{x})$ represents output of a hidden layer neuron. μ_{ji}s give the center of the each Gaussian function and σ_{ji}s gives a measure of the with of Gaussian functions. Input and weight vectors are represented as $\underline{x}$ and $\underline{w}$ respectively.

2.2. Fuzzy Logic

Fuzzy logic, the theory of imprecision, is radically different than the classical tools of mathematics which rely on precision and exact knowledge. That is why the concept of

fuzzy logic encountered sharp criticisms from the academic community when first introduced by Zadeh in 1965. However, later on, with its successful applications, capabilities of fuzzy logic are understood better. Especially, from a control point of view, fuzzy inference systems offer an efficient methodology to a system designer. They can incorporate expert knowledge to implement systems which can mimic decisions of a human operator. Moreover, this kind of knowledge may also be utilized to reduce convergence time in adaptive applications. In this work, two different fuzzy inference systems will be utilized. Thus, in the following subsections these inference systems are investigated after a short introduction about the basic concepts of fuzzy logic.

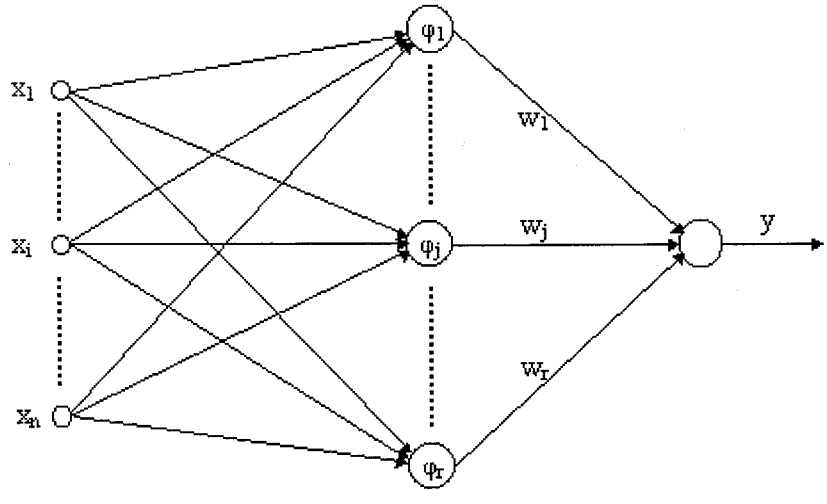

Figure 2.3. Radial basis function network

2.2.1. Fuzzy Sets

Let U be a universe of discourse which includes all objects related to problem in hand. A crisp set, A, on U is defined by the members it has and can be represented by assigning 1 or 0 values to each object in U to determine weather the object is a member of A or not respectively. Here, the assignment rule is called the characteristic function of A. A set of temperatures grater than 26 degrees can be given as an example to a crisp set and expressed as

$$H = \{x \mid x > 26\} \tag{2.11}$$

in closed form. The characteristic function of H is given in Figure 2.4. As can be seen from the figure, there is a sharp transition in membership grade at 26 degrees.

Although, crisp sets constitute a useful tool for many application areas, they are not appropriate to deal with qualitative information. For example, if one applies the (2.11) to express hot weather, it can be said 26.0001 degrees is hot while 26 degrees is not. It is obvious that this results conflicts with the way we express the hot weather in our daily life.

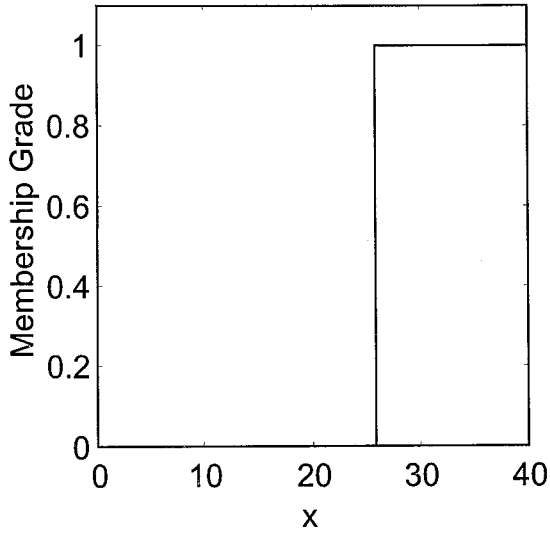

Figure 2.4. Characteristic function of crisp set H

Fuzzy sets theory offers systematic methodologies to alleviate the problem mentioned above. A fuzzy set can be defined by expending characteristic function concept of crisp sets by assigning values to objects from [0,1] interval instead of only 0 and 1. The assigned value determines the degree of membership of an object with 1 is the highest degree while 0 is the lowest. For a fuzzy set A, this scheme can be formulated as

$$A = \{(x, \mu_A(x)) \mid x \in U\} \tag{2.12}$$

where $\mu_A(x)$ is the membership function which determines to what degree the element x belongs to A. In the light of this definition, now it is possible to obtain a more meaningful explanation of the above mentioned hot weather concept with the help of the membership

function given in Figure 2.5. As can be seen from this figure, the membership degree of set H increases gradually as temperature increases.

2.2.2. Basic Operations on Fuzzy Sets

In this subsection, three basic operations of classical set theory, complement, union and intersection, will be extended to fuzzy sets by the help of membership function representation introduced in the previous subsection.

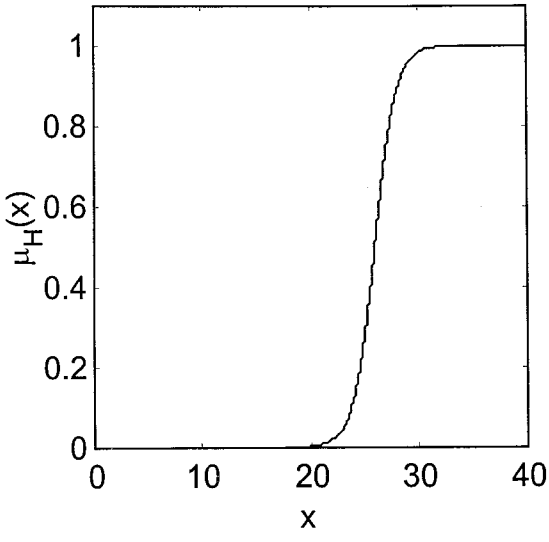

Figure 2.5. Membership function of fuzzy set H

In classical set theory, complement of a set is defined as a set which contains all elements in U that does not belong to A. Although, this definition is meaningful for crisp sets, it cannot be directly applied to fuzzy sets. In this regard, the following definition can be used to define fuzzy complement.

$$\overline{A} = \{(x, 1-\mu_A(x)) \mid x \in U\} \tag{2.13}$$

As can be seen from (2.13) this definition still holds for crisp sets because if x is an element of A, membership grade of x to set $\overline{A}$ will be 0 while it will be 1 if x is not a member. Application of (2.13) to a fuzzy set A is demonstrated in Figure 2.6 in graphical form. In fact, (2.13) is not the unique definition of fuzzy complement. In general form,

fuzzy complement is defined as a continuous function $N:[0,1]\rightarrow[0,1]$ which meets the following axiomatic requirements:

$$\begin{aligned} &N(0)=1 \text{ and } N(1)=0 \text{ (boundary)} \\ &N(a)\geq N(b) \text{ if } a\leq b \text{ (monotonicity)} \end{aligned} \tag{2.14}$$

Another optional constraint that may be imposed on a fuzzy complement is involution which can be expressed as

$$N(N(a))=a \tag{2.15}$$

This property guarantees that double complement of a fuzzy set is still itself. There exist several proposed complements satisfying both (2.14) and (2.15) in the literature.

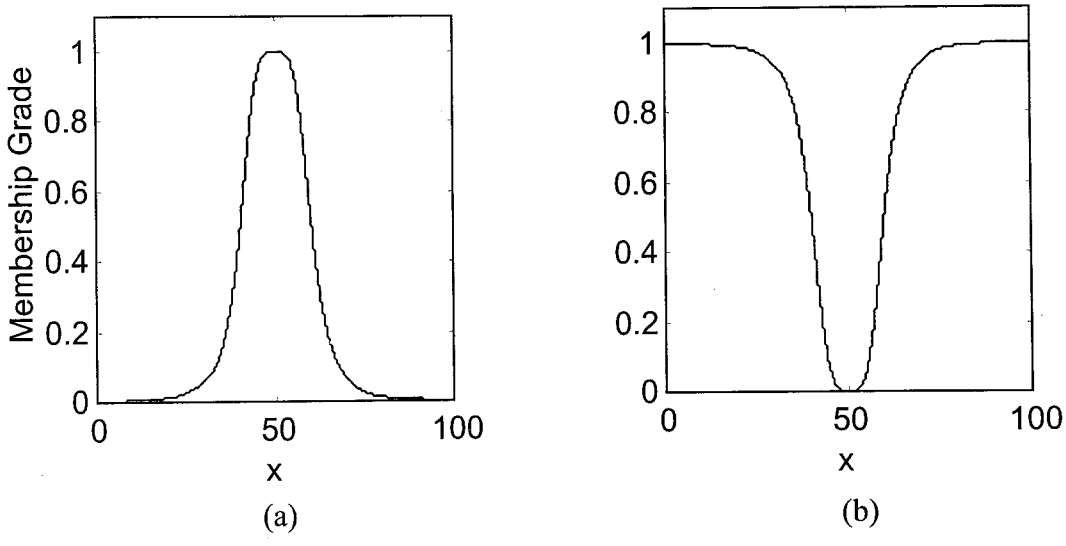

Figure 2.6. (a) Membership function of fuzzy set A; (b) complement of A

Union or disjunction is another operation defined for crisp sets. If A and B represents two fuzzy sets with corresponding membership functions μ_A and μ_B respectively, an intuitive counterpart of union for fuzzy sets can be defined as

$$C=\{(x,\max(\mu_A(x),\mu_B(x))\,|\,x\in U)\} \tag{2.16}$$

The graphical representation of this operation is shown in Figure 2.7. Although (2.16) is appropriate to describe union, it is not the unique definition. In the literature, There exists several disjunction operators which obey the following set of rules.

$$\begin{aligned} &S(1,1)=1, S(0,a)=S(a,0)=a \\ &S(a,b)\le S(c,d) \text{ if } a\le c \text{ and } b\le d \\ &S(a,b)=S(b,a) \\ &S(a,S(b,c))=S(S(a,b),c) \end{aligned} \tag{2.17}$$

where S is disjunction function ($S:[0,1]\times[0,1]\rightarrow[0,1]$).

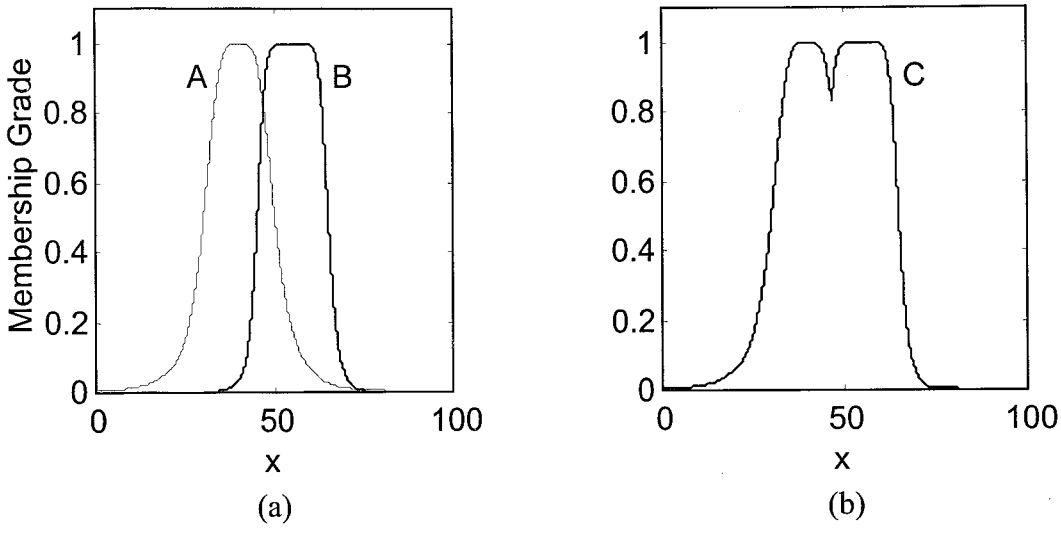

Figure 2.7. (a) Membership functions of sets A and B; (b) result of union operation

Intersection, also known as conjunction, is the last basic operator of fuzzy sets and can be defined as

$$C=\{(x,\min(\mu_A(x),\mu_B(x))\,|\,x\in U)\} \tag{2.18}$$

This operation is shown in Figure 2.8.

Another alternative to the definition of conjunction is product operator which will be used in the thesis and can be given as

$$C = \{(x, \mu_A(x) \times \mu_B(x) \mid x \in U)\} \tag{2.19}$$

It is also possible to define conjunction operators other than the ones given in (2.18) and (2.19) as long as they satisfy set of conditions in (2.20).

$$\begin{aligned} &T(0,0) = 0, T(a,1) = T(1,a) = a \\ &T(a,b) \le T(c,d) \text{ if } a \le c \text{ and } b \le d \\ &T(a,b) = T(b,a) \\ &T(a,T(b,c)) = T(T(a,b),c) \end{aligned} \tag{2.20}$$

where T is disjunction function ($T : [0,1] \times [0,1] \rightarrow [0,1]$).

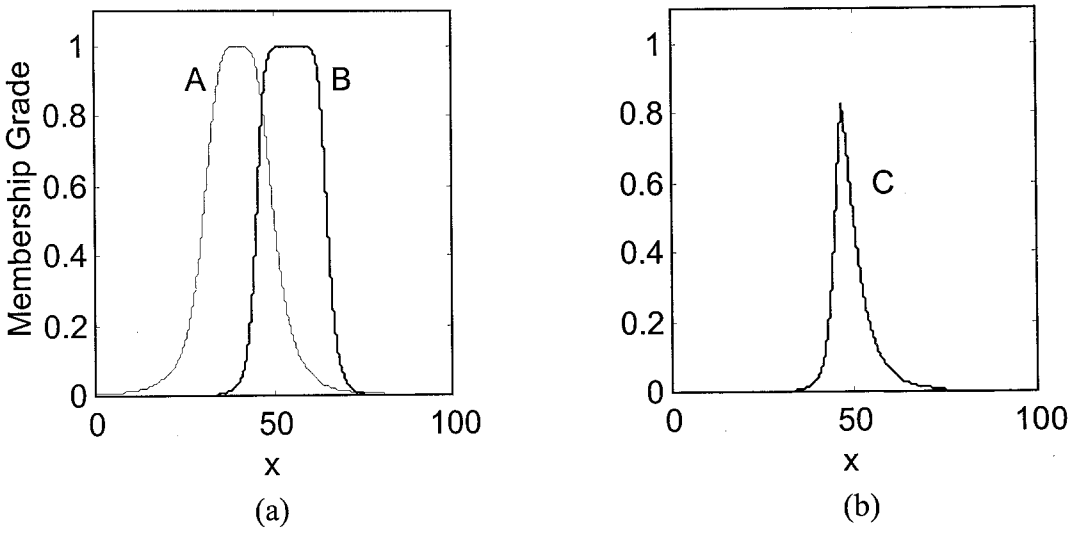

Figure 2.8. (a) Membership functions of sets A and B; (b) result of intersection operation

2.2.3. Linguistic Variables

As discussed in subsection 2.2.1 fuzzy sets provide a useful framework to make qualitative judgments about a quantity by means of mathematical computations. By this property, fuzzy logic allows us to make a smooth transactions from symbolic domain to mathematical domain. For instance, while a human makes judgments about weather by adjectives like hot, cold, warm, computers use numbers like 45° to characterize it. In fact, here both human and computer give a representative value of a variable, but with this form, there is not a clear correspondence between two domains. Fuzzy sets can be utilized to

provide an interface between human thoughts and mathematical domain. We can assign a fuzzy set to each linguistic value and then use membership functions to represent each set in mathematical from. By this way, it is possible for computers to make manipulations based on linguistic values of the quantity. Here the quantity taking fuzzy sets as values is called linguistic variable. Zadeh gives a more formal definition of the linguistic variable concept:

A linguistic variable is characterized by a quintuple $(x, T(x), X, G, M)$ in which x is the name of the variable; $T(x)$ is the term set of x— that is, the set of its linguistic values or linguistic terms; X is universe of discourse; G is a syntactic rule which generates the terms in $T(x)$; and M is a semantic rule which associates with each linguistic value A its meaning $M(A)$, where $M(A)$ denotes a fuzzy set in X [1].

2.2.4. Fuzzy Inference Systems

Today, fuzzy inference systems have been successfully applied in many disciplines ranging from pattern recognition to automatic control systems. There are several factors behind the success of fuzzy inference systems. Especially, when control applications are taken into account two of them are dominating. First, a fuzzy inference system can realize nonlinear mappings which obey to a rough description in linguistic form. Second, a fuzzy inference system can exploit the expert knowledge given in form of fuzzy if-then rules to realize controllers and reduce training time in fuzzy adaptive applications.

The main constituent of fuzzy inference systems is fuzzy if-then rules. Fuzzy if-then rules have the form of

$$\text{if } x_1 \text{ is } A_1 \text{ and/or } x_2 \text{ is } A_2 \text{ and/or } \ldots x_n \text{ is } A_n \text{ then } y \text{ is } C$$

Here, the part "x_1 is A_1 and/or x_2 is A_2 and/or ... x_n is A_n" is called premise and "y is C" is called consequence. A premise in general is a compound statement which is made up by conjunction, disjunction connectors and statements in form of "x_i is A_i". In this representation A_i' s are the values of linguistic variables each of which is defined in the universe of its corresponding input.

At that point, it will be helpful to explain the main philosophy of fuzzy if-then rules. A fuzzy if-then rule, in fact, is a fuzzy relation which inherits mathematical information related with its linguistic representation. Thus, compositional rule of inference can be readily applied to fuzzy rules to make deductions which are similar to ones produced by a human expert. The following procedure, where minimum operator is used for conjunction, is a consequence of compositional rule of inference when applied to fuzzy if-then rules. In this procedure, as a first step, input values, fuzzy or crisp, and linguistic values in the premise part of the rule corresponding to each input are passed through *and* operation to determine the matching degree of the input. Then, matching degrees of all inputs are combined using connectors in the premise to produce the firing strength of the rule. At the last step, the output of the rule is evaluated by taking the minimum of the firing strength, which is a fuzzy singleton, and consequence of the rule. This procedure is shown in Figure 2.9 graphically, where minimum operation is used as conjunction.

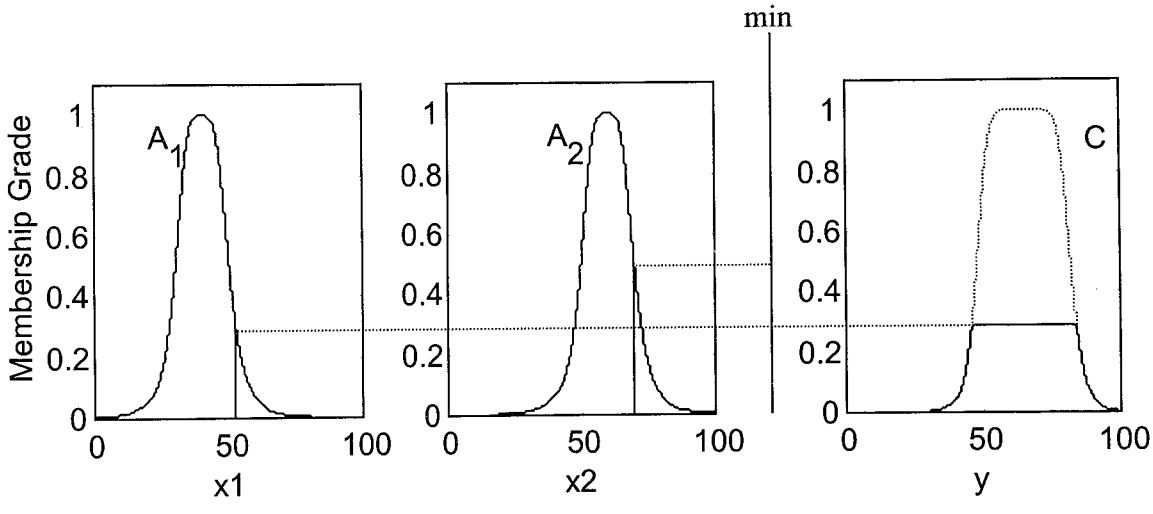

Figure 2.9. Fuzzy reasoning procedure

After introducing fuzzy if-then rules, now it is possible give a description of fuzzy inference systems. A fuzzy inference system consists of several rules which are formed by utilizing the rough knowledge about the problem to attain a representative structure. Each rule in the system produces an output that is more active than the other rules in a partition of the input space which is determined by the premise part of the rule. The outputs of all rules are, then, combined by an appropriate aggregation mechanism to attain the overall

output of the system which is a fuzzy set in general. Although, results in form of fuzzy sets may be adequate for many applications, they are not useful in automatic control problems. Thus, a last stage, which is called defuzzification stage, is necessary to attain a crisp result which inherits the most important characteristics of the reasoning procedure. Centroid of area, bisector of area, mean of maximum and the largest of maximum are the most basic methods in this regard. While the overall picture of a fuzzy inference system is given in Figure 2.10, a schematic representation of the inference procedure is depicted in Figure 2.11. Here maximum of rule consequences is chosen as aggregation method.

In the rest of this section, two fuzzy inference systems, which are utilized in this work, will be investigated in detail.

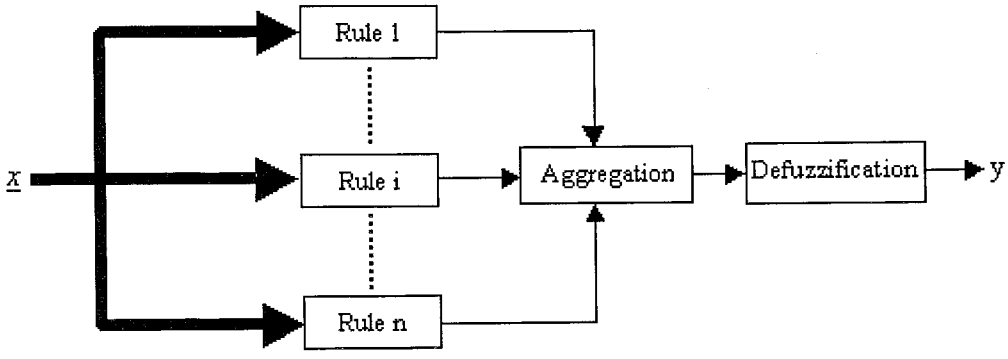

Figure 2.10. Block diagram for a fuzzy inference system

2.2.5. Standard Fuzzy System

Several fuzzy inference systems, which are inspired by the reasoning mechanism explained in subsection 2.2.4, exist in the literature. The difference between them lies in the conjunction and disjunction operators, aggregation methods and the defuzzification strategies they use [1]. SFS, which is one of these variants, has the same structure with the inference system described in the previous subsection except that it restricts the rule consequences to singletons in favor of reducing computational complexity and producing more suitable mappings. Rules of this structure can be expressed as

$$\text{if } x_1 \text{ is } A_{i1} \text{ and } x_2 \text{ is } A_{i2} \text{ and } \ldots x_n \text{ is } A_{1n} \text{ then } f_i = w_i$$

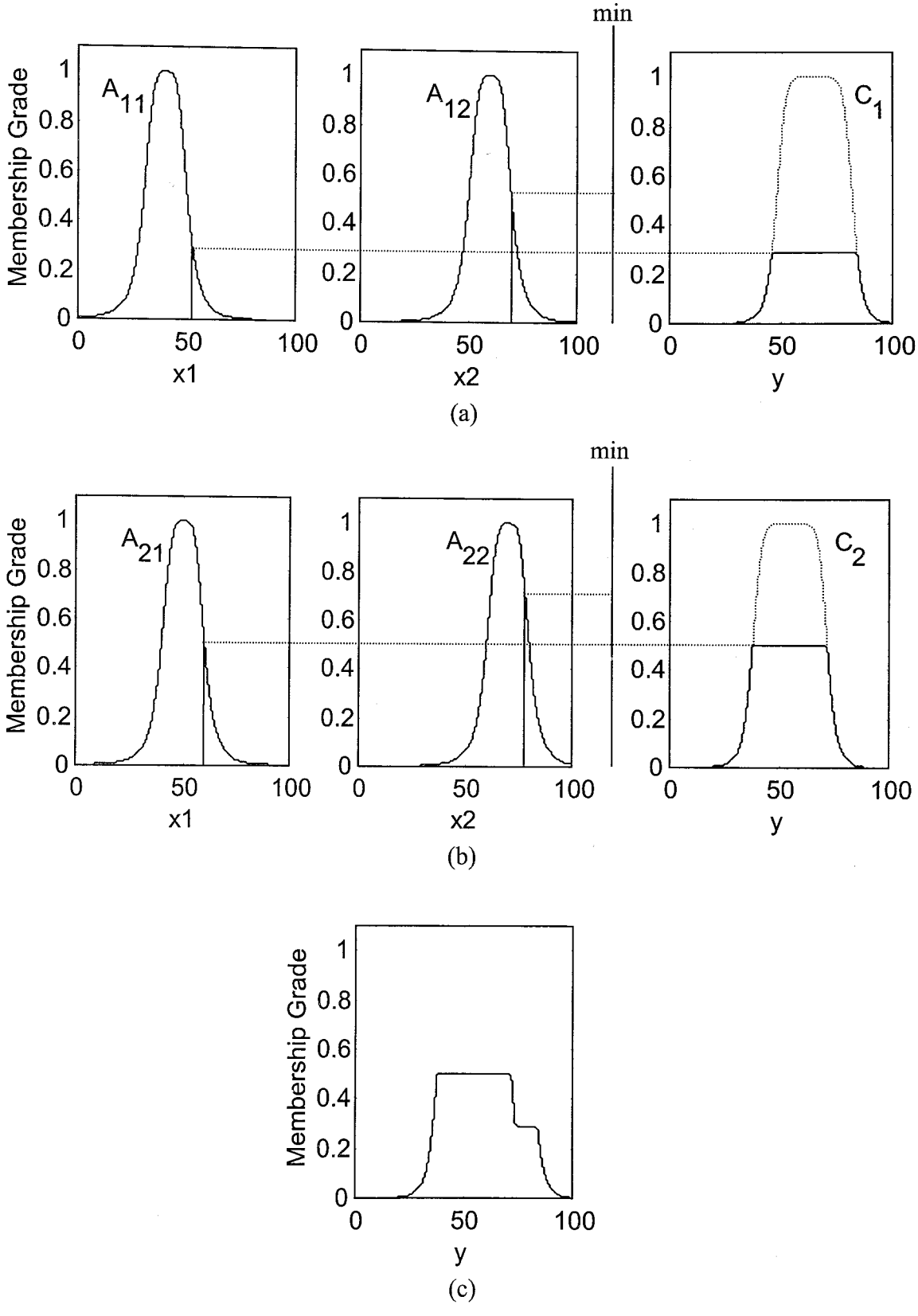

Figure 2.11. Fuzzy inference procedure; (a) rule1; (b) rule 2; (c) aggregated output

where w_i is a crisp number. Data flow of this structure is shown in Figure 2.12. In the premise part of this structure multiplication is used as conjunction operator. For aggregation and defuzzification stages, weighted average of the rule consequences is used. The set of equations, which describes a SFS with bell shaped membership functions, is given below.

$$u_i(\underline{x}) = \prod_{j=1}^{n} \frac{1}{1 + \left| \frac{x_j - \mu_{ij}}{\sigma_{ij}} \right|^{2b_{ij}}} \tag{2.21}$$

$$u_i^n(\underline{x}) = \frac{u_i(\underline{x})}{\sum_{j=1}^{r} u_j(\underline{x})} \tag{2.22}$$

$$y = \underline{w}^T \underline{u}^n(\underline{x}) \tag{2.23}$$

where

$$\underline{w} = [w_1, w_2, \ldots, w_r]^T \tag{2.24}$$

$$\underline{u}^n(\underline{x}) = [u_1^n(\underline{x}), u_2^n(\underline{x}), \ldots u_r^n(\underline{x})]^T \tag{2.25}$$

2.2.6. Adaptive Neuro-Fuzzy Inference System

So far, we have seen that both fuzzy inference systems and neural networks are appropriate tools for realizing input-output mappings, but they differ in the way they deal with the information. ANNs can produce input-output mappings through a training procedure which tries to minimize an error measure between the samples and network output. In other words, ANNs are capable of realizing input-output mappings which best fits to the available data samples given in numerical form. One drawback of ANNs is that their internal representation of data in form of weights connecting large number of neurons has no intuitive meaning except some structures like GRBFNNs. On contrary, while fuzzy inference systems are capable of realizing mappings based on rough observations about the system given in linguistic form, they suffer from the lack of an optimization procedure for

fine tuning. One natural way to eliminate drawbacks of both systems is to combine them under one structure. One of the architectures proposed in this direction is Adaptive Neuro-fuzzy Inference System (ANFIS). Rule base of an ANFIS structure with first order Sugeno model may be given as

$$\text{if } x_1 \text{ is } A_{i1} \text{ and } x_2 \text{ and } A_{i2} \text{ ... then } f_i = w_{i1}x_1 + w_{i2}x_2 + \ldots + w_{in}x_n + w_{i(n+1)}$$

Here the consequence of each rule is a linear function of inputs. This property makes ANFIS very suitable for realization of functional mappings. The adaptive network representation of ANFIS structure is shown in Figure 2.13. As depicted by the figure product operator is used in premise part of the rules, while aggregation and defuzzification stages are realized by calculating weighted average of rule consequences. This structure with bell shaped membership functions can be expressed as follows in mathematical terms.

$$u_i(\underline{x}) = \prod_{j=1}^{n} \frac{1}{1 + \left| \frac{x_j - \mu_{ij}}{\sigma_{ij}} \right|^{2b_{ij}}} \tag{2.26}$$

$$u_i^n(\underline{x}) = \frac{u_i(\underline{x})}{\sum_{j=1}^{r} u_j(\underline{x})} \tag{2.27}$$

$$f_i(\underline{x}) = \underline{w}_i^T \begin{bmatrix} \underline{x} \\ 1 \end{bmatrix} \tag{2.28}$$

$$y = f^T(\underline{x}) u^n(\underline{x}) \tag{2.29}$$

where

$$\underline{x} = [x_1, x_2, \ldots, x_n]^T \tag{2.30}$$

$$\underline{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}, w_{i(n+1)}]^T \tag{2.31}$$

$$\underline{f}(\underline{x}) = [f_1(\underline{x}), f_2(\underline{x}), \ldots f_r(\underline{x})]^T \tag{2.32}$$

$$\underline{u}^n(\underline{x}) = [u_1^n(\underline{x}), u_2^n(\underline{x}), \ldots u_r^n(\underline{x})]^T \tag{2.33}$$

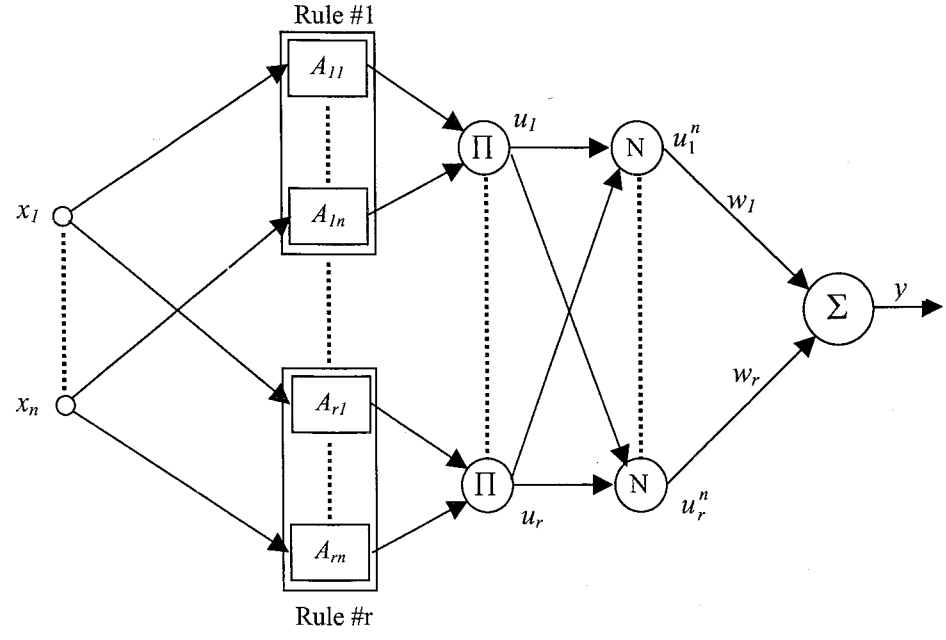

Figure 2.12. Standard Fuzzy System

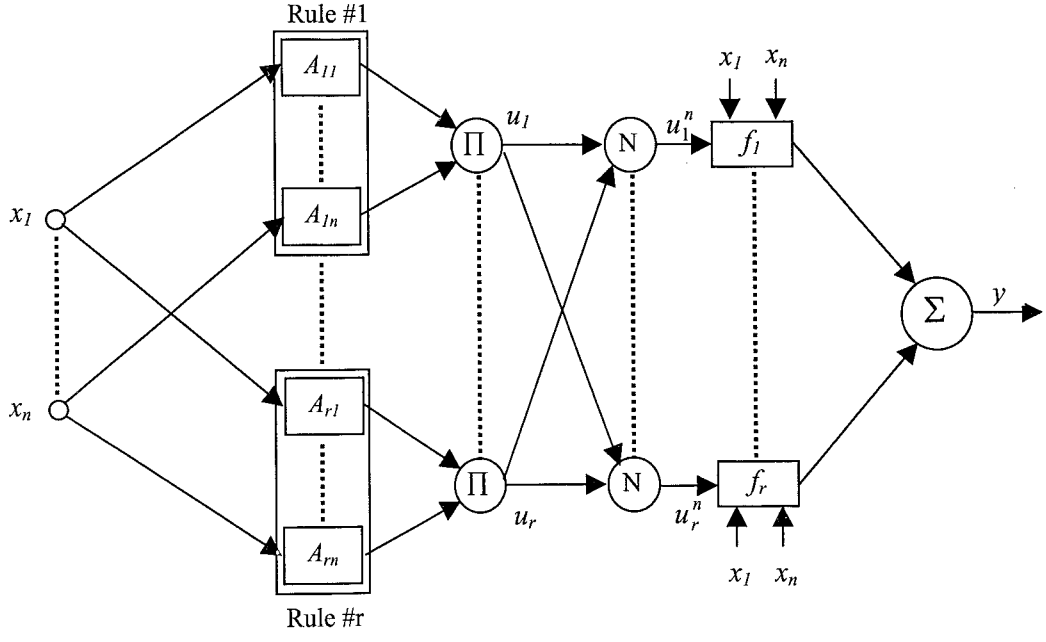

Figure 2.13. Adaptive Neuro-fuzzy Inference System

3. VARIABLE STRUCTURE SYSTEMS THEORY AND SLIDING MODE CONTROL

Variable structure systems theory, which is first introduced in 1950s, utilizes a discontinuous control law that switches between different control actions depending on the state of the system. Although, when applied separately, these control actions may be insufficient to provide satisfactory performance, if they are combined with an appropriate switching law, desired control objectives may be imposed to closed loop system. This kind of strategy offers a wide range of possibilities to control system designers. Especially, if the switching law is chosen properly, trajectories of the system may be forced towards a manifold in the state space of the system so that they reach to the manifold in a finite time interval. This situation is shown in Figure 3.1. The period until the trajectories hit to the sliding manifold is referred to reaching time in the related literature. Once the system is trapped to the manifold, a sliding motion starts, during which trajectories are governed by the equations of the manifold, which can be defined by the designer to impose desired control objectives to the system. Control systems utilizing this kind of strategy are called sliding mode control systems (SMC). SMC systems possess many attractive properties. First, they are insensitive to unstructured uncertainties because motion in sliding regime is governed by the characteristics of the sliding manifold which is independent of the system model. Moreover, because the dimension of sliding manifold is lower than that of state space of the system, the order of the differential equations governing the motion of the system is reduced while it is in the sliding regime [2].

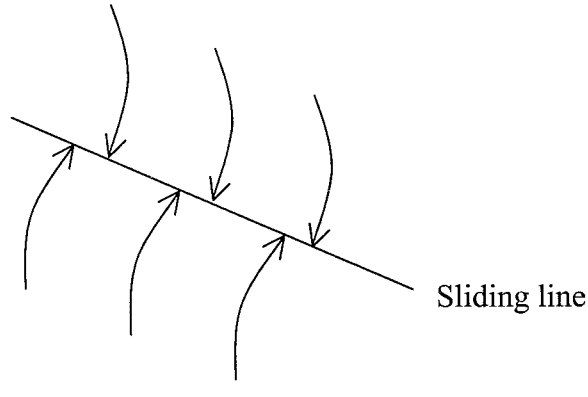

Figure 3.1. Trajectories of a SMC system

3.1. SMC System Design

This section is devoted to SMC design for a class of multi-input, multi,-output systems described by the following equations.

$$x_i^{(m_i)} = f_i(\underline{x}) + \sum_{j=1}^{n} b_{ij} u_j \qquad i = 1,...,n \tag{3.1}$$

For the sake of analytical tractability (3.1) can be rewritten as

$$\dot{\underline{x}} = \underline{f}(\underline{x}) + B\underline{u} \tag{3.2}$$

where

$$\underline{x} = [x_1, \dot{x}_1, ..., x_1^{m_1-1}, ..., x_n, \dot{x}_n, ..., x_n^{m_n-1}]^T \tag{3.3}$$

$$\underline{u} = [u_1, u_2, ..., u_n] \tag{3.4}$$

If the tracking error vector between the actual state of the system and desired state is expressed as

$$\underline{e} = \underline{x} - \underline{x}_d \tag{3.5}$$

one can define a (n×1) dimensional sliding surface vector of form

$$\underline{s}_p(\underline{e}) = G\underline{e} \tag{3.6}$$

in the error space of the system. With an appropriate choice of the gain matrix, G, sliding surface equations may take the following form which ensures the asymptotic stability of the system while it is in sliding regime.

$$s_i = (\frac{d}{dt} + \lambda_i)^{m_i-1} e_i \tag{3.7}$$

If the following function is chosen as Lyapunov candidate

$$\frac{1}{2}\underline{s}_p^T(\underline{e})\underline{s}_p(\underline{e}) \tag{3.8}$$

negative definiteness of the time derivative of (3.8) ensures the occurrence of sliding mode in error space of the system. This kind of condition can be obtained by solving following equations.

$$\underline{s}_p^T(\underline{e})\dot{\underline{s}}_p(\underline{e}) = -\underline{s}_p^T(\underline{e})K\,\mathrm{sgn}(\underline{e}) \tag{3.9}$$

The resulting control law can be expressed by the following equations as sum of two components namely, equivalent control ($\underline{u}_{eq}$), corrective control ($\underline{u}_c$).

$$\underline{u} = \underline{u}_{eq} + \underline{u}_c \tag{3.10}$$

where

$$\underline{u}_{eq} = -(GB)^{-1}[G\underline{f}(\underline{x}) - G\dot{\underline{x}}] \tag{3.11}$$

$$\underline{u}_c = -(GB)^{-1}K\,\mathrm{sgn}(\underline{s}_p) \tag{3.12}$$

Although, it provides a useful framework, the procedure described above suffers from some disadvantages. First of all, corrective term requires a discontinuous switching function to establish the sliding motion. This kind of strategy leads to chattering in which a very high frequency control signal is applied to the system. Chattering is undesirable in practice because high frequency input signals may result in unforeseen instabilities by exciting unmodeled high frequency dynamics of the system. Furthermore, the corrective term may produce an unnecessarily high control signal to account for uncertainties. Calculation of the equivalent term constitutes another difficulty because it requires a complete knowledge of the plant model. There are several methods proposed in the literature to alleviate theses difficulties. Among them the most important one is the boundary layer approach. In this method, discontinuous control is replaced by a linear control law in the vicinity of the sliding surface to eliminate high frequency switching

behavior [15]. Another modification to eliminate chattering, which will be utilized in this work, is the use of a smooth function to approximate sign term in the corrective control. Sigmoid, linear and arctangent functions are the most frequently used ones in this regard [16].

4. A CONTINUOUS-TIME ADAPTATION SCHEME FOR INTELLIGENT CONTROLERS

In this section, a parameter adaptation scheme for intelligent control systems is investigated. In this regard, first a SMC based learning algorithm is introduced for training of flexible structures. Then, it is discussed that how intelligent architectures utilizing continuous-time robust learning mechanisms can be incorporated into control of nonlinear dynamical systems.

4.1. Stabilizing Learning Dynamics by Means of SMC System Theory

In addition to reasoning capabilities, an intelligent system must also be able to learn from past experiences by storing the knowledge acquired from interactions with its environment. As long as it is consistent within the system, internal representation of knowledge may take different forms depending on the paradigm which is utilized. In case of soft computing methodologies, data is represented as adjustable parameters of appropriate input-output mappings. Thus, learning in soft computing methodologies can be boiled down to the problem of finding optimal parameter set which best represents the available data. Unfortunately, finding optimal parameter set is not an easy task in general. Least mean square error minimization algorithms may result in nonlinear equations which cannot be solved analytically. Therefore, it may be necessary to employ slow iterative optimization methods for training of intelligent structures.

The method of error back propagation, which is first proposed for multi-layer networks, is one of the most widely used training algorithms for intelligent structures. In each step of the algorithm, the error information at the output of the network is propagated backwards through the structure to obtain gradient information related with each adjustable parameter. Then, the negative gradient direction is used to determine the new value of the parameter vector. Another method, Levenberg-Marquardt method, utilizes additional information by incorporating Hessian matrix of the error cost function into learning process. While this strategy leads to faster convergence in terms of number of iteration steps, it increases the computational complexity of the algorithm. Both error

backpropogation and Levenberg-Marquardt methods are extensively used for training of different soft computing architectures. But these techniques suffer from some disadvantages. First, because they are discrete-time methods, some difficulties may arise in applications to continuous time systems. Moreover, they cannot guarantee the stability of the learning dynamics. The presence of noise and uncertainties in real applications may decrease stability further.

In the iterative learning methods, the parameter vector of the flexible structure at a time instant depends on its past values. Thus, learning process of a flexible structure can be modeled as a dynamic system in which adjustable parameter vector represents the state of the system. In the light of this fact, it may be possible to incorporate the tools of control system theory into training of intelligence architectures. Especially, SMC approach may provide a useful framework to alleviate instability problems mentioned above. The following parameter training method utilizes this idea to robustify learning dynamics in ADALINE networks.

Consider the flexible structure depicted in Figure 2.2. The learning error-level at the output of the structure is defined as

$$s_c = y - y_d \tag{4.1}$$

where y_d is the desired output sequence. If the parameter vector ($\underline{w}$), input vector ($\underline{x}$), the time derivative of the input vector ($\dot{\underline{x}}$) and the time derivative of the desired output ($\dot{y}_d$) satisfies the following inequalities

$$\|\underline{w}\| \le N_w \tag{4.2}$$

$$\|\underline{x}\| \le N_x \tag{4.3}$$

$$\|\dot{\underline{x}}\| \le N_{\dot{x}} \tag{4.4}$$

$$|\dot{y}_d| \le N_{\dot{y}_d} \tag{4.5}$$

the parameter update mechanism, described as

$$\dot{\underline{w}} = -\frac{\underline{x}}{\underline{x}^T \underline{x}} K \operatorname{sgn}(s_c) \tag{4.6}$$

enforces the flexible structure to sliding surface (s_c=0) so that it reaches to zero learning error-level in finite time which is estimated as

$$t_h \leq \frac{|s_c(0)|}{K - (N_w N_{\dot{x}} + N_{\dot{y}_d})} \tag{4.7}$$

To ensure the occurrence of sliding regime K must satisfy the following criterion.

$$K > N_w N_{\dot{x}} + N_{\dot{y}_d} \tag{4.8}$$

For mathematical proof the reader is referred to [7].

The adaptation mechanism described above provides an efficient tool for learning in ADALINE networks. This fact is demonstrated by the following example in which ADALINE structure with three inputs is utilized. It is required to track the trajectory described as

$$y_d(t) = 0.4 \sin(10t) \cos(20t) \tag{4.9}$$

while the system is subject to the following constant input signal.

$$\underline{x} = [x_0 \;\; x_1 \;\; x_2] = [1 \; 2 \; 1.5] \tag{4.10}$$

To smooth out chattering while system is in the sliding regime, sgn term in (4.6) is replaced with the following approximating function.

$$\operatorname{sgn}(s) = \frac{s}{|s| + 0.005} \tag{4.11}$$

Results of computer simulations are depicted in Figure 4.1 for K=20 and, the schematic representation of the intelligence structure together with update mechanism is shown in Figure 4.2. As can be seen from the simulation results, the output signal converges to desired trajectory very fast. Furthermore, the parameters of the structure remain bounded.

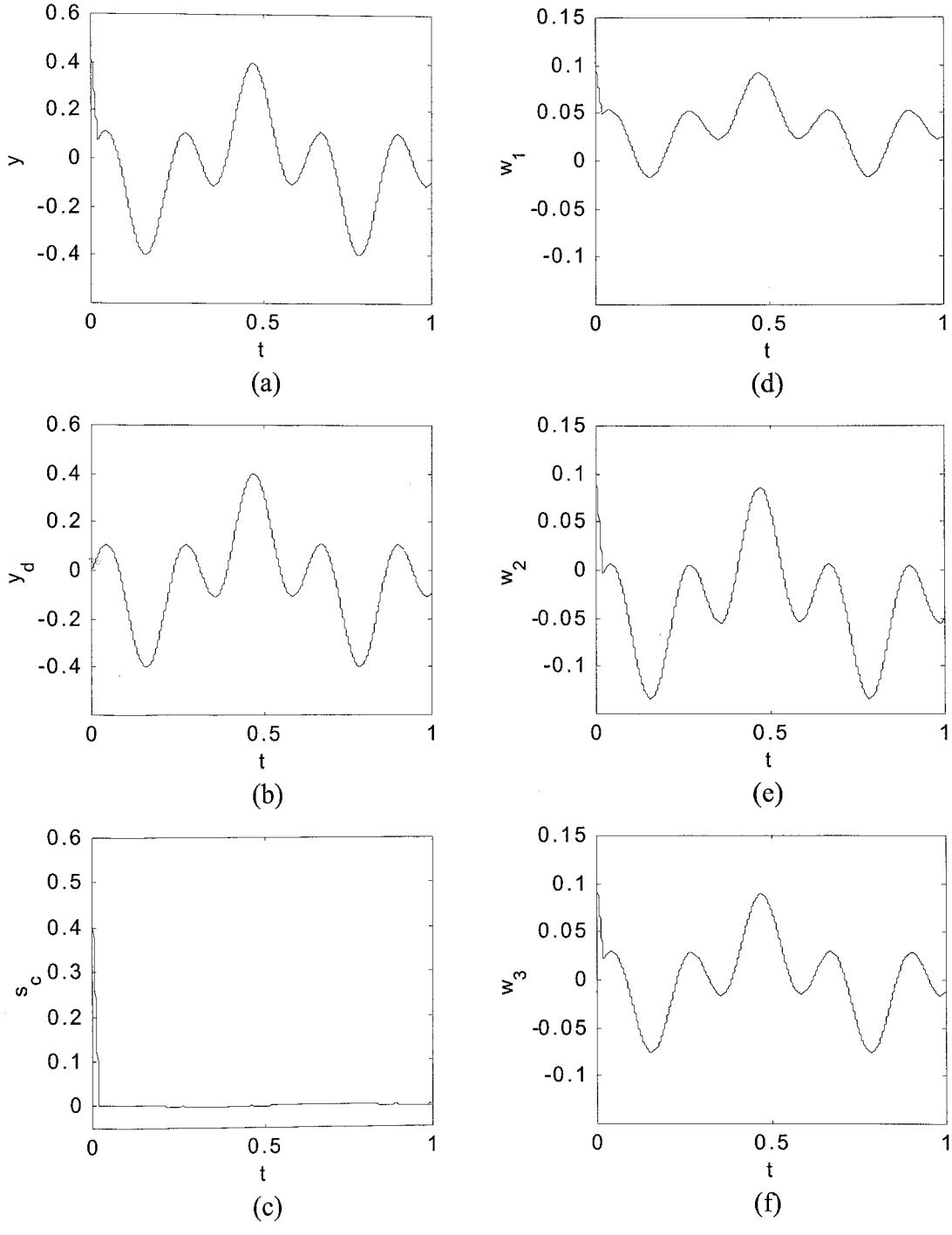

Figure 4.1. Simulation results for training of ADALINE network

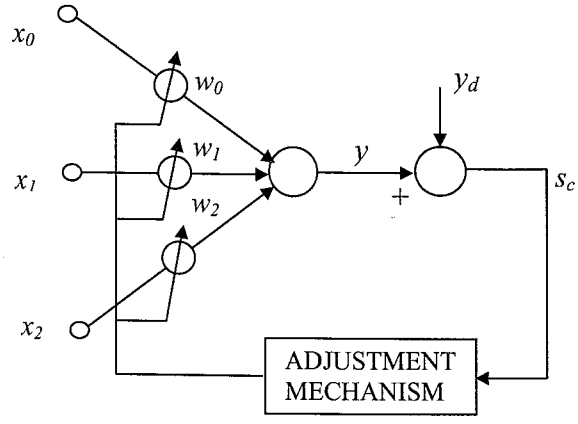

Figure 4.2. Schematic representation of SMC based parameter adjustment mechanism for an ADALINE network

In fact, this scheme can be extended to any intelligent architecture having linear parameters with respect to its output. In below, the adaptation mechanism is formulated for the other architectures discussed in Section 2.

GRBNN:

$$\dot{\underline{w}} = -\frac{\underline{\varphi}}{\underline{\varphi}^T \underline{\varphi}} K \operatorname{sgn}(s_c) \tag{4.12}$$

SFS:

$$\dot{\underline{w}} = -\frac{\underline{u}^n}{(\underline{u}^n)^T \underline{u}^n} K \operatorname{sgn}(s_c) \tag{4.13}$$

ANFIS:

$$\dot{\underline{w}}_i = -\frac{[\underline{x}^T \; 1] u_i^n}{(\underline{x}^T \underline{x} + 1)(u^n)^T u^n} K \operatorname{sgn}(s_c) \tag{4.14}$$

where

$$\underline{w}_i = [w_{i1} \; w_{i2} \ldots w_{in} \; w_{i(n+1)}] \tag{4.15}$$

4.2. Application of SMC Based Learning Algorithms to Control of Nonlinear Systems

As discussed before, soft computing architectures can learn the rules and regularities in a system by means of training methods. Especially, when the control of partially known systems is of primary concern, this learning ability may provide a useful tool to obtain satisfactory performance in the presence of uncertainties. Figure 4.3 depicts one of the most frequently employed intelligent control schemes for control of partially known systems. This scheme includes a forward identification model, which is a soft computing architecture and utilizes several past system inputs and outputs provided by tapped delay lines represented by bold arrows in the figure, to attain a measure of error at the output of the controller. This error measure is, then, utilized by the training mechanism to adapt the parameters of the intelligent controller in order to improve the performance of the closed loop dynamics of the system. Despite of being very useful, this control mechanism has some drawbacks. First, requirement of a forward identification model increases the computational burden of the control algorithm and restricts its applications only to discrete-time systems. Furthermore, there is not a well developed theory for stability of this control structure.

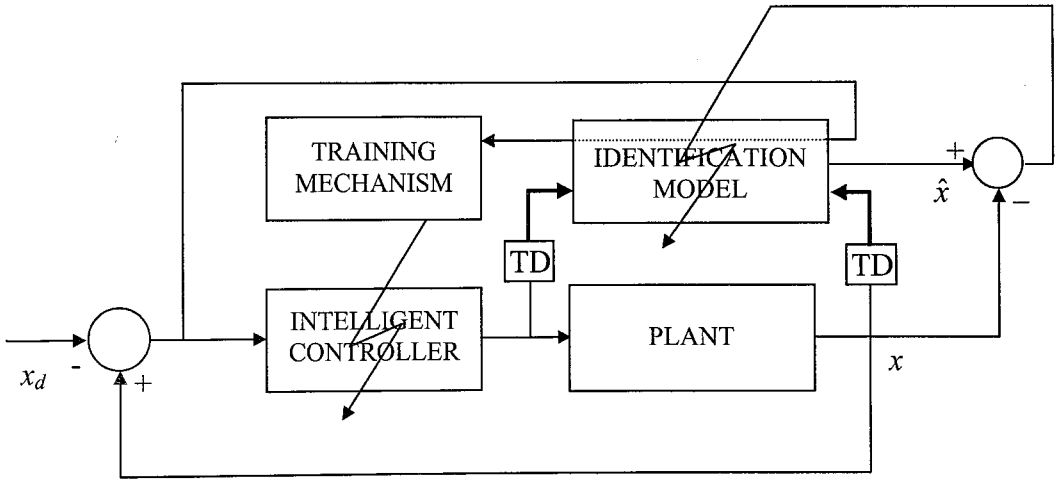

Figure 4.3. Conventional intelligent control scheme

To somehow alleviate the problems associated with the conventional intelligent control schemes, Efe [10] has proposed an alternative approach which incorporates continuous-time SMC based learning algorithms into training of intelligent controllers.

This approach eliminates the need for a forward identification model, thus, reduces the computational complexity of the parameter update mechanism. Moreover, the design procedure given in [10] is based on the assumption that the system in hand can be described by a single-input model in canonical form. However, in practice not all plants can be modeled in canonical form and they may have multiple inputs. This difficulty can be alleviated by dividing the plant model into first order subsystems having couplings in between, and then each sub system can be controlled by a separate controller.

The overall block diagram of the control loop for a first order system is shown in Figure 4.4. As depicted in the figure, the discrepancy (e) between the desired (x_d) and actual (x) states is defined as the sliding surface at the output of the plant (s_p). The functional relation between s_p and the learning error-level of the intelligent controller (s_c) is chosen as $s_c = \Psi(s_p) = s_p$. The reason behind this choice is that it is the simplest relation which satisfies the conditions given in [10]. In this scheme, training mechanism utilizes s_c to update parameters of the controller in order to attain optimum performance. It is possible to use different flexible structures as intelligent controller depending on the system requirements. In this thesis, four different architectures are utilized, thus, the rest of this section is devoted to analysis of the control scheme for these structures.

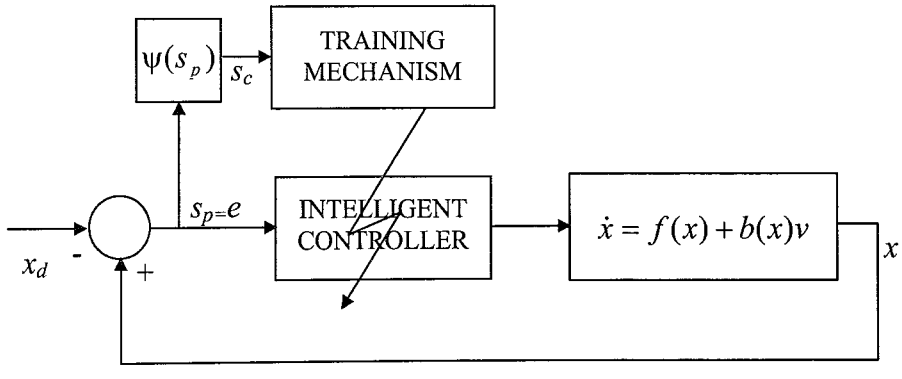

Figure 4.4. Continuous-time adaptation scheme for intelligent controllers

Analysis of ADALINE controller is helpful to understand the basic behavior of the parameter update mechanism since it possesses the fundamental characteristics which is common to all controllers. As discussed in 2.1.1 an ADALINE network provides a linear

multi-input single-output mapping, which is described by (2.3)-(2.5), between the input signals, including bias, and output signal. If one utilizes this flexible structure in the control loop shown in Figure 4.4. input-output relation of the structure can be written as

$$v = w_1 e + w_0 \tag{4.16}$$

As can be seen from (4.16) this mapping includes a proportional control term and a bias term.

The equations of the parameter update mechanism for this structure, which can be easily derived from (4.6), can be expressed as

$$\dot{w}_1 = -\frac{e}{1+e^2} K \operatorname{sgn}(e) = -\frac{|e|}{1+e^2} K \tag{4.17}$$

$$\dot{w}_0 = -\frac{1}{1+e^2} K \operatorname{sgn}(e) \tag{4.18}$$

A brief analysis of these parameter adaptation equations reveals the behavior of the control mechanism. As (4.17) suggests the weight associated with the proportional control term will decrease as long as the error is not equal to zero. Although, this strategy may be adequate to increase stability because the system is first order, it is unpractical since small deviations from the zero error-level, which are unavoidable in real applications due to the noise and uncertainties, will lead to the weight to evolve unboundedly. The other term given in (4.18) tries to eliminate the steady state error by employing discontinuous switching to update the bias term. In other words, the bias term is adjusted so that the controller can produce the nominal input value corresponding to desired state in order to attain zero error at the output of the plant in steady state. The overall behavior of the control mechanism is summarized in Figure 4.5 in graphical form. As shown in the figure, the slope of the line decreases as long as the error is not equal to zero. The intersection of the line with e=0 axis gives the bias term.

As in the ADALINE case, it is also possible to give a quantitative analysis about behavior of the update mechanism for GRBFNN and SFS controllers. Because the analysis

for both structures follows the same reasoning, only GRBFNN will be investigated here. For GRBFNN the equations describing input-output behavior of the controller and time evolution of its weights can be given as

$$v = \underline{w}^T \underline{\varphi}(e) \tag{4.19}$$

$$\underline{\varphi}(e) = [\varphi_1(e), \varphi_2(e), ..., \varphi_r(e)]^T \tag{4.20}$$

$$\dot{w}_i = -R_{Gi}(e) K \operatorname{sgn}(e) \tag{4.21}$$

$$R_{Gi}(e) = \frac{\varphi_i(e)}{\underline{\varphi}(e)^T \underline{\varphi}(e)} \tag{4.22}$$

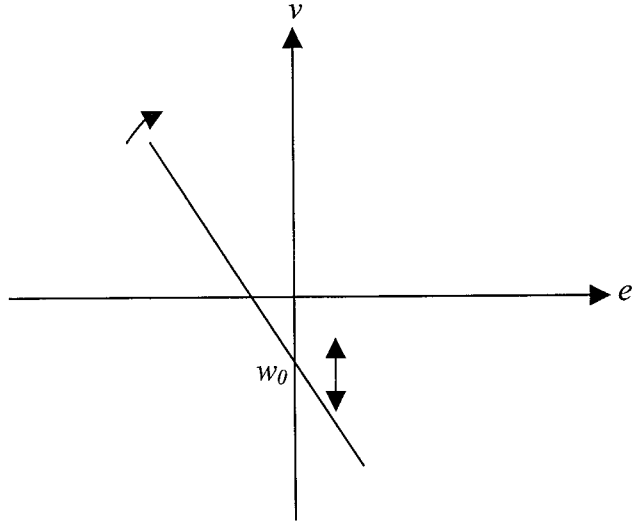

Figure 4.5. Input-output relation of ADALINE controller

From (4.21), one can easily see that time derivative of the weight associated with i^{th} neuron (w_i) is proportional with the term R_{Gi} which determines the rate of change in the weight. This term includes the activation function of the i^{th} neuron (φ_i), which is a Gaussian function, as multiplicative factor, and thus, it converges to zero rapidly as the distance between the controller input (e) and center of the activation function increases. Based on this reasoning, it can be said that input-output relation of the controller will be updated only in a neighborhood of the e because update rate for the neurons, whose center

of activation function is not close to e, will be almost zero. This strategy is shown in Figure 4.6 in graphical form for both $e<0$ and $e>0$. As can be seen form the figure, while error is less than zero part of the curve in some neighborhood of input will move upwards. On the other hand, if error is grater than zero the curve will go downwards around the input. In fact, this behavior is similar with ADALINE case in the local sense, that is, only some partition of the curve will be updated instead of whole curve.

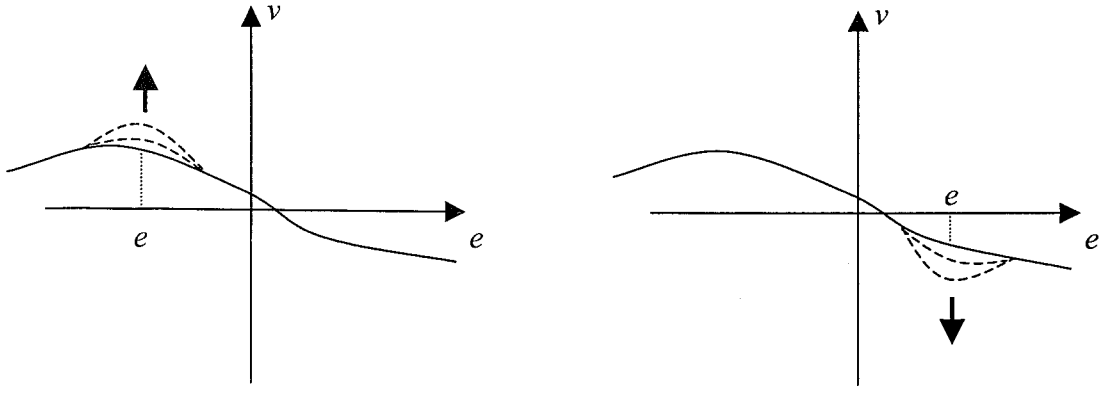

Figure 4.6. Time evolution of input-output curve of GRBFNN controller; (a) $e<0$; (b) $e>0$

The last architecture investigated in this thesis is Adaptive Neuro-fuzzy Inference system, which has good approximation capabilities due to extensive degree of freedom it possesses. Input-output relation of this structure can be given as

$$v = \underline{f}^T(e)\underline{u}^n(e) \tag{4.23}$$

$$\underline{f}(e) = \left[w_{11}e + w_{12}, w_{21}e + w_{22}, \ldots, w_{r1}e + w_{r2}\right]^T \tag{4.24}$$

$$\underline{u}^n(e) = \left[u_1^n(e), u_2^n(e), \ldots, u_r^n(e)\right] \tag{4.25}$$

and the equations governing time evolution of parameters are expressed as

$$\dot{w}_{i1} = -R_{Ai}(e)\frac{e}{e^2+1}K\,\mathrm{sgn}(e) = -R_{Ai}(e)\frac{|e|}{e^2+1}K \tag{4.26}$$

$$\dot{w}_{i2} = -R_{Ai}(e)\frac{1}{e^2+1}K\,\mathrm{sgn}(e) \tag{4.27}$$

$$R_{Ai}(e) = \frac{u_i^n(e)}{(\underline{u}^n(e))^T\underline{u}^n(e)} \tag{4.28}$$

Apparently from (4.26) and (4.27), parameter update equations for each linear function corresponding to different rules are same as ADALINE structure except that they are multiplied with term R_{Ai}. This term effects update rate of the parameters, and similar to case of GRBFNN it decreases as distance between the input (e) and center of membership function corresponding to i^{th} rule increases. Thus, again it can be said that the input-output curve of the controller will be updated locally depending on the sign of the error, and it will move upwards if error is negative and downwards if error is positive. This reasoning further confirms the similarity between different control architectures.

Although, the control strategy introduced in this section provides a good mechanism for control of nonlinear systems described by first order dynamical equations in canonical form, it suffers from problem of unbounded evolution of adjustable parameters. For ADALINE structure, deviations from zero error-level substantially decreases the network weight associated with error input (w_1), thus, increases the sensitivity of the controller with respect to its input (e). To get rid of this parameter drift problem, in the applications investigated in section 5, the update term given in (4.17) is turned off if the absolute value of error is less than a certain value, which can be determined by the system designer. This strategy ensures availability of necessary proportional control gain by increasing w_1 until steady state and prevents the parameter to increase unboundedly.

As in the ADALINE case, unbounded evolution controller parameters constitute difficulties in other structures. Based on the observations for GRBFNN, it can be said that if small deviations from zero error-level occurs at the output of the plant, the weight associated with the neuron, whose center of activation function is to the left of origin, increases substantially. Similarly, if the center of activation function of a neuron is to the right of the origin, its weight will decrease as long as the error is close but not equal to zero. The natural result of this behavior is that the slope of the curve describing input-output behavior of the controller will decrease in a local neighborhood of the origin. It is

needless to say that this behavior is quite similar with that of ADALINE controller in the local sense.

It is possible to give an intuitive explanation of above mentioned observations for GRBFNN based controller by investigating the update equation (4.21). Consider that the error at the output plant fulfils the following conditions

$$|e(t)| < \varepsilon, \quad \text{for } \forall t > t_h \tag{4.29}$$

$$\lim_{t\to\infty} \int_0^t |e(\tau)|\, d\tau = +\infty \tag{4.30}$$

where ε is a constant which is close enough to zero. Here, the first assumption requires the system to be stable, and the second one states that the error cannot converge to zero because of noise and uncertainties. Under these circumstances, it is easy to show that the long term behavior of the weights w_i and w_{i+1}, which are the weights associated with neurons whose center of activation functions are the closest ones to the e=0 line from left and right respectively, can be given as

$$\lim_{t\to\infty} w_i(t) = +\infty \tag{4.31}$$

$$\lim_{t\to\infty} w_{i+1}(t) = -\infty \tag{4.32}$$

In the following, the above given statement will be proven by contradiction.

To ease the proof of the statement mentioned above, graphical interpretation of the rate terms for both neurons (R_{Gi}, R_{Gi+1}) is given in Figure 4.7 on the same graph. As can be seen from the figure, it is possible to use below given linear approximation of both terms at e=0 due to the constraint imposed by (4.29).

$$R_{Gi}(e) = m_i e + n_i \tag{4.33}$$

$$R_{Gi+1}(e) = m_{i+1} e + n_{i+1} \tag{4.34}$$

where

$$m_i < 0,\ m_{i+1} > 0 \text{ and } n_i, n_{i+1} > 0 \tag{4.35}$$

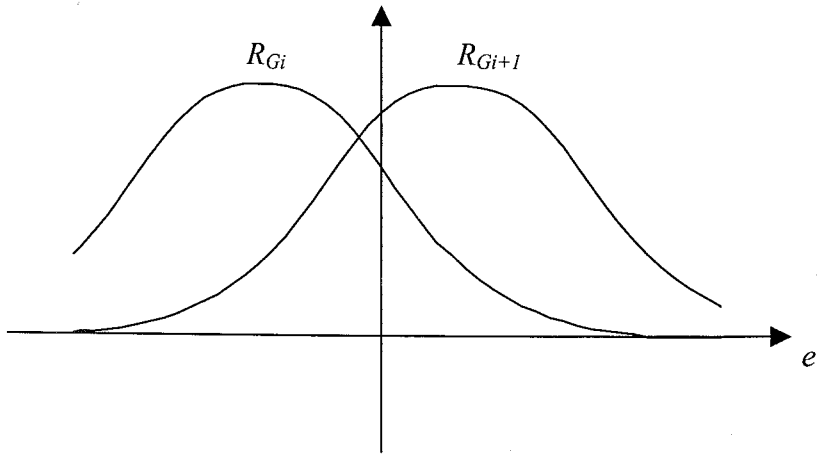

Figure 4.7. Graphical representation of two update rates corresponding i^{th} and $(i+1)^{th}$ neurons.

Let's we assume that the limit of weight w_{i+1} is bounded from below, that is,

$$\lim_{t\to\infty} w_{i+1}(t) > M > -\infty \tag{4.36}$$

Under this assumption the following relations can be induced.

$$\lim_{t\to\infty} w_{i+1}(t) = \lim_{t\to\infty} -\int_0^t R_{Gi+1}(e) K \operatorname{sgn}(e)\, d\tau + w_{i+1}(0)$$

$$= \lim_{t\to\infty} -\int_0^t (m_{i+1} e + n_{i+1}) K \operatorname{sgn}(e)\, d\tau + w_{i+1}(0) \tag{4.37}$$

$$= -\lim_{t\to\infty} m_{i+1} K \int_0^t |e|\, d\tau - \lim_{t\to\infty} n_{i+1} K \int_0^t \operatorname{sgn}(e) d\tau + w_{i+1}(0) > M > -\infty$$

By rearranging the last equation, following result can be obtained.

$$-n_{i+1}K\lim_{t\to\infty}\int_0^t \mathrm{sgn}(e) > m_{i+1}K\lim_{t\to\infty}\int_0^t |e|\,d\tau - w_{i+1}(0) + M$$

$$\lim_{t\to\infty}\int_0^t \mathrm{sgn}(e) < -\frac{1}{n_{i+1}K}\left(m_{i+1}K\lim_{t\to\infty}\int_0^t |e|\,d\tau - w_{i+1}(0) + M\right) = -\infty \tag{4.38}$$

$$\lim_{t\to\infty}\int_0^t \mathrm{sgn}(e) = -\infty$$

In what follows, the long run behavior of the weight w_i is obtained by the help of the last equation given in (4.38) which is valid under assumption (4.36).

$$\begin{aligned}\lim_{t\to\infty} w_i(t) &= \lim_{t\to\infty} -\int_0^t R_{Gi}(e)\,K\,\mathrm{sgn}(e)\,d\tau + w_i(0)\\ &= \lim_{t\to\infty} -\int_0^t (m_i e + n_i)\,K\,\mathrm{sgn}(e)\,d\tau + w_i(0)\\ &= -\lim_{t\to\infty} m_i K\int_0^t |e|\,d\tau - \lim_{t\to\infty} n_i K\int_0^t \mathrm{sgn}(e)\,d\tau + w_i(0) = +\infty\end{aligned} \tag{4.39}$$

Recall from (4.19) that the output of the controller can be written as

$$v(e) = w_1\varphi_1(e) + \ldots + w_i\varphi_i(e) + w_{i+1}\varphi_{i+1}(e) + \ldots + w_r\varphi_r(e) \tag{4.40}$$

If it is taken into account that w_{i+1} is greater than M, and the weights other than w_i and w_{i+1} will not be updated because their update rate R_{Gj} will be almost zero, we can calculate the following limit.

$$\lim_{t\to\infty} v(e) = \lim_{t\to\infty}\{w_1\varphi_1(e) + \ldots + w_i\varphi_i(e) + w_{i+1}\varphi_{i+1}(e) + \ldots + w_r\varphi_r(e)\}$$

$$= \lim_{t\to\infty} w_i\varphi_i(e) + \lim_{t\to\infty}\{w_1\varphi_1(e) + \ldots + w_{i-1}\varphi_{i-1}(e) + w_{i+1}\varphi_{i+1}(e) + \ldots + w_r\varphi_r(e)\} \qquad (4.41)$$

$$> \lim_{t\to\infty} w_i\varphi_i(e) + M' = +\infty$$

where M' is the lower bound of the limit for the second term given in (4.41). It is obvious that this result contradicts with the stability assumption expressed in (4.29) because the control input will increase indefinitely and so the output of the plant. This reasoning shows that if the closed loop system is stable and deviations from zero error level is inevitable, the weight w_{i+1} converges to -∞. The proof of the statement given in (4.32) can also be shown following the same reasoning and will not be given here.

The main reason behind the unbounded parameter evolution problem of the GRBFNN controller is that because slope of the tangent of the term R_{Gi} is not equal to zero at e=0, the effect of positive (negative) deviations from zero error-level on derivative of the weight will be more dominant than that of negative (positive) deviations. On natural way to follow in order to alleviate this undesirable behavior is to set activation functions (φ) of fired neurons to +1 if the error at the output of the plant is less than a certain value. This modification makes the slope of rate term functions zero, and thus, prevents parameters to evolve indefinitely. The proposed method is successfully employed in all simulations given in section 5.

Explanations for ADALINE and GRBFNN controllers also hold true for other architectures. By comparing (4.12) and (4.13), it is possible to see the similarity between update equations of SFS and GRBFNN structures. Although, mathematical expressions of rate terms differ for both architectures, qualitative properties of curves describing them are quite similar. Therefore, without further investigation, it can be said that the qualitative behavior of SFS controller is the same as that of GRBFNN, and thus, technique described above can be readily applied to keep parameters of SFS bounded. To understand that the parameter update mechanism of ANFIS controller inherits behavior of both ADALINE and GRBFNN architectures, equations (4.26)-(4.28) must be analyzed. From these equations, it is easy to see that time derivative of the weights corresponding to first order terms

associated with each rule is less than zero as long as error is not equal zero. Thus, similar to ADALINE network, the weight of first order terms for each fired rule will substantially decrease with the small deviations from zero error-level. To get rid of this problem, same technique is utilized for ANFIS controller, that is, update terms of these parameters are turned off when the absolute value of error is less then a predetermined value. The similarity with GRBFNN, also SFS, lies in the second set of parameters, associated with zero order terms. Although, rate terms of these parameters differs from that of SFS with factor $1/(1+e^2)$, the characteristics of both update terms will be similar because the signs of their slopes at $e=0$ are same for both architectures. Given this, fact one can easily apply the method utilized in SFS to prevent unbounded evolution of this parameter set.

5. APPLICATIONS

The aim of this section is to evaluate performance of four control architectures, ADALINE, GRBFNN, SFS and ANFIS, utilizing the adaptation mechanism introduced in the previous section. For this purpose, three different nonlinear systems, a biochemical tank reactor, a cement mill circuit and a chaotic system, are chosen as test beds. The former ones are examples of large-scale industrial processes, which involve highly nonlinear characteristics and uncertainties, and the later one is the dynamic model of a phenomenon which occurs in a variety of systems changing from electronic circuits to chemical reactions. Based on the simulation results for each system, performances of all architectures are compared.

5.1. Bioreactor Process

The control of a bioreactor system is a challenging task because of complex characteristics of cell growth kinetics. This complexity stems from several factors. First, because of the scale of the tanks, in which biological reactions take place, and high viscosity and non-Newtonian nature of the medium in some situations, conditions within the container can differ from point to point. Moreover, since process involves several factors like pH of the medium, temperature, mechanical interactions and concentrations of several substances, clearly it is not reasonable to construct models characterizing all aspects of the biochemical reactions. Practically, one of the variables, which is the variable to be controlled, is assumed to be the key state of the system. But, generally, this kind of assumption results in models with structured and unstructured uncertainties. Thus, in the applications, where the objectives require precision and robustness, the use of classical control methods is not sufficient.

There are three common configurations for bioreactors: batch, fed-batch and continuous. When compared with the batch operation, in which growth dynamics may vary with time and environmental conditions might show nonlinear characteristics, continuous-stirred tank reactors (CSTR) with their ideal conditions allow straightforward mathematical analysis. Schematic representation of a CSTR is depicted in Figure 5.1. As the figure

suggests, there are two states variables, namely substrate concentration and cell concentration denoted by s and c respectively. While the substrate, which is necessary for growth of microorganisms, is added into the system by means of liquid influent, the reactor content, which includes the products, is removed from the reactor at rate equal to the inflow rate to keep the volume of the liquid in the system at a constant level, and this rate is called dilution rate denoted by D through which the system is controlled.

There exist several models characterizing the dynamics of CSTR. In general, the difference between them lies in the growth model they utilize. The following nonlinear differential equations describe a CSTR with Haldene growth model [17].

$$\dot{c} = \mu(s,c)c - Dc, \qquad c(0) = c_0 > 0 \tag{5.1}$$

$$\dot{s} = -\frac{\mu(s,c)c}{Y} + (S_F - s)D, \qquad s(0) = s_0 > 0 \tag{5.2}$$

$$\mu(s,c) = \frac{\mu_0 s}{K_S + s + \frac{s^2}{K_I}} \tag{5.3}$$

In above, μ represents the growth model of the system, S_F denotes the influent substrate concentration, Y denotes the yield coefficient and μ_0, K_S and K_I are the parameters of the growth model.

5.1.1. Control of the Bioreactor Process

The strategy introduced in section 4 is utilized to control the bioreactor system described by (5.1)-(5.2). For this purpose, substrate concentration is chosen as the state to be controlled. The target output is set to s=0.2 [g/l], and initial states of the plant are chosen as c_0=10 [g/l], s_0=10 [g/l]. Simulations are performed in MATLAB 5.3 environment and step size is chosen as 0.001 hours. During the simulations, it is assumed that the parameters of the growth model possesses time varying behavior described by the following equations while the yield coefficient and influent substrate concentration are kept at constant levels given by Y=0.5 [g cells/g substrate], S_F=200 [g/l].

$$\mu_0 = 0.35 + 0.15\cos(2\pi t/10)\ [1/h] \tag{5.4}$$

$$K_S = 0.105 + 0.095\sin(2\pi t/15 + 3\pi/2)[g/l] \tag{5.5}$$

$$K_I = 5 + 4.975\cos(2\pi t/25)[g/l] \tag{5.6}$$

In order to not to be in conflict with practical reality, the output measurement is subjected to Gaussian noise with zero mean and variance equal to 1.64e-6. To smooth out chattering, the sgn function is replaced by the following approximating function

$$\mathrm{sgn}(s_c) \approx \frac{s_c}{|s_c| + \delta} \tag{5.7}$$

where δ=0.005. Furthermore, the uncertainty bound in the parameter update algorithm is chosen as K=3 and the relevant mechanism which prevents unbounded parameter evolution for each controller is activated when absolute value of error is less then 0.1 for all of the structures studied in the following subsections. Lastly, output of the controller is passed through a saturation function because dilution rate cannot take negative values.

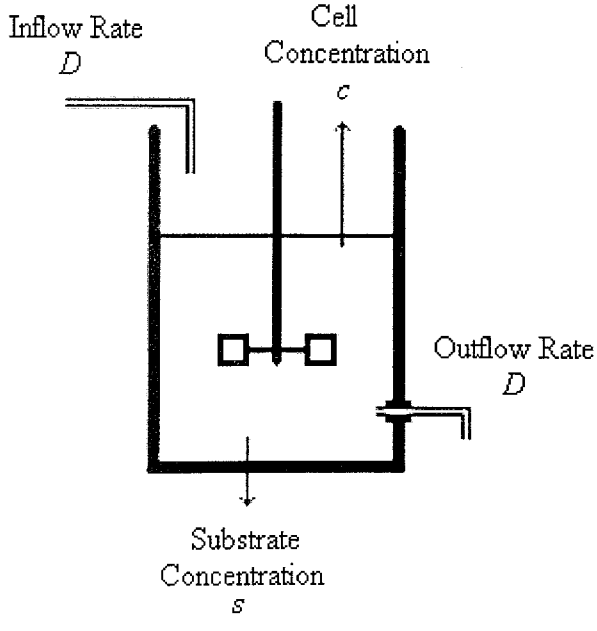

Figure 5.1. Schematic representation of the bioreactor system

5.1.2. Application with ADALINE Based Controller

In this subsection, the performance of ADALINE based controller is investigated. The simulation results are shown in the following figures. As Figure 5.2 depicts, the substrate rate converges to its desired value very rapidly. This is especially important because undesirable by-product formation may occur while the substrate concentration is above a limiting value. The cell concentration reaches its steady state value after 40 hours and the applied control input is acceptably smooth as can be seen from Figure 5.3. The time evolutions of controller parameters are shown in Figure 5.4. As the figure suggest, after a transient period parameters of the controller enters a steady state regime.

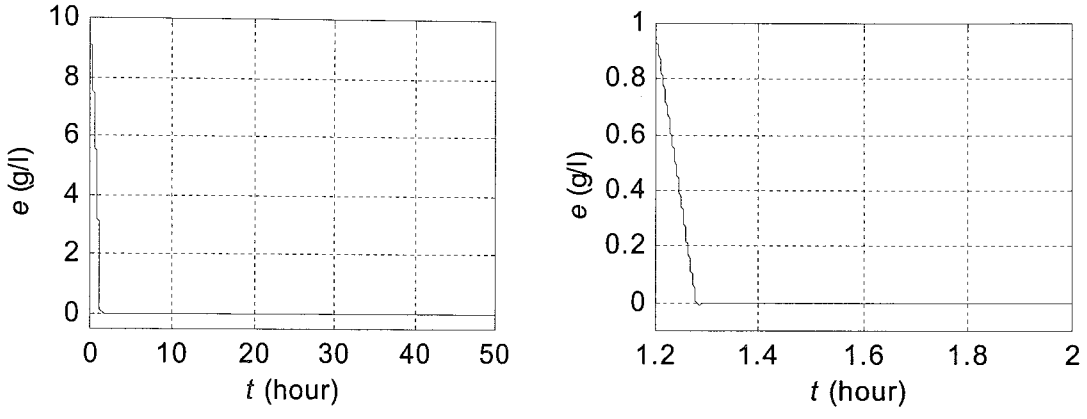

Figure 5.2. Substrate concentration error of the bioreactor system for ADALINE based controller

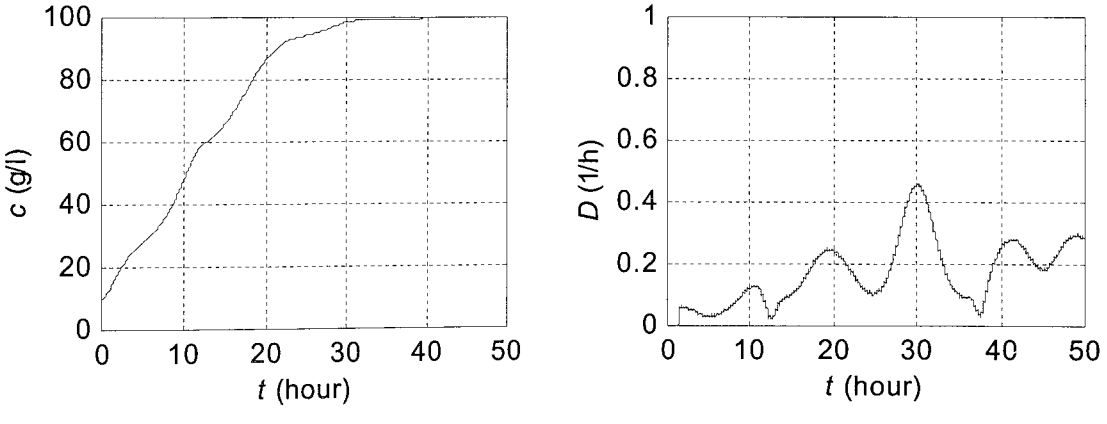

Figure 5.3. Cell concentration and dilution rate of the bioreactor system for ADALINE based controller

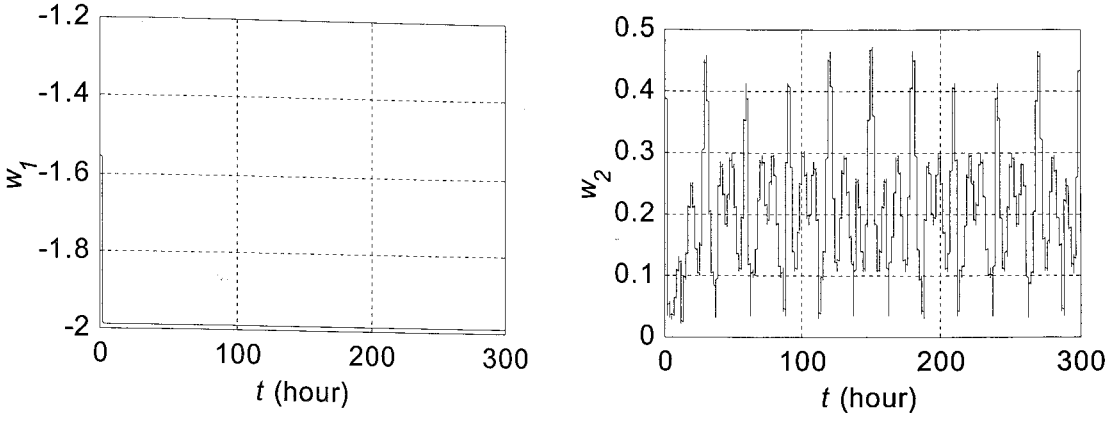

Figure 5.4. Time evolution of parameters for ADALINE based controller of the bioreactor system

5.1.3. Application with GRBNN Based Controller

Simulation results for GRBFNN based controller are shown below. The results given in Figure 5.5 stipulate that the transient phase for substrate concentration lasts acceptably short period of time. Based on the first graph given in Figure 5.6, it can be said that the settling time for cell concentration is about 40 hours, which is much longer than that of substrate concentration. Moreover, as shown in the same figure we cannot see chattering on applied control signal because of sign function smoothing in the vicinity of the decision boundary characterized by e=0. As Figure 5.7 suggests, activation functions of hidden layer neurons spans [-2,10] interval within which error signal remains. Furthermore, the parameters of GRBFNN controller, which are shown in Figure 5.8, remain bounded.

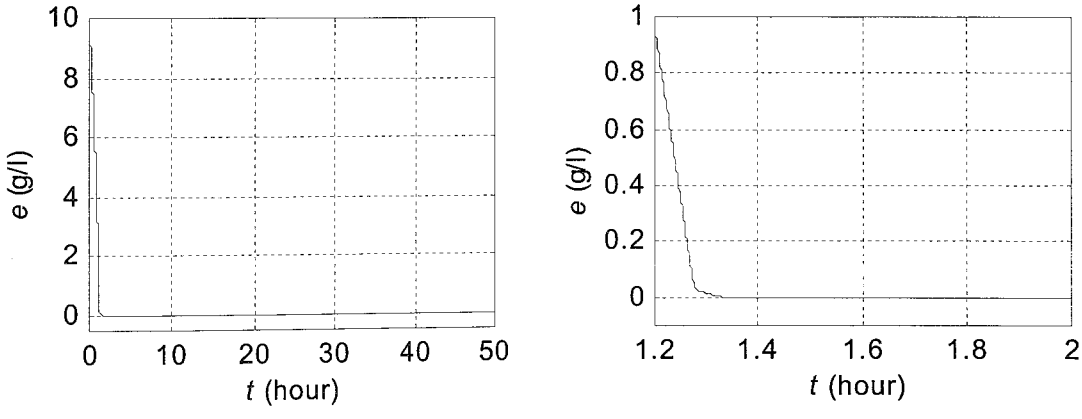

Figure 5.5. Substrate concentration error of the bioreactor system for GRBFNN based controller

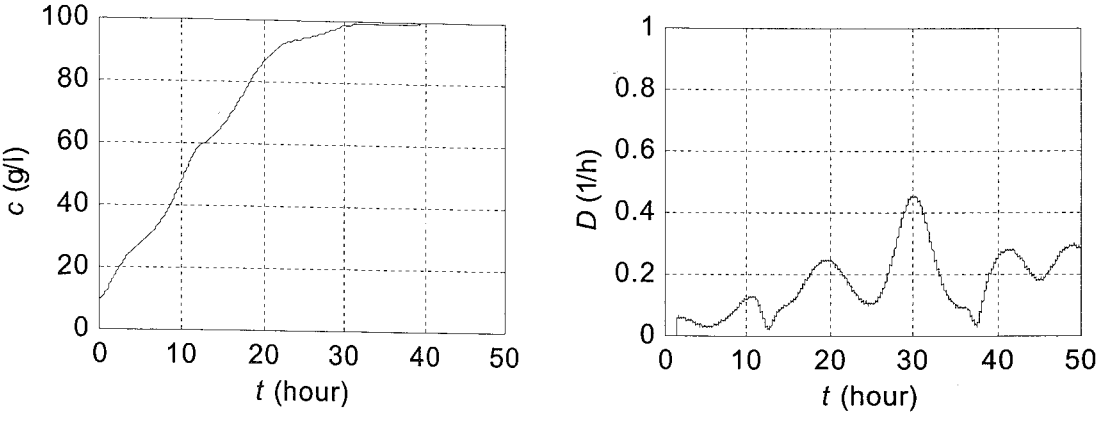

Figure 5.6. Cell concentration and dilution rate of the bioreactor system for GRNFNN based controller

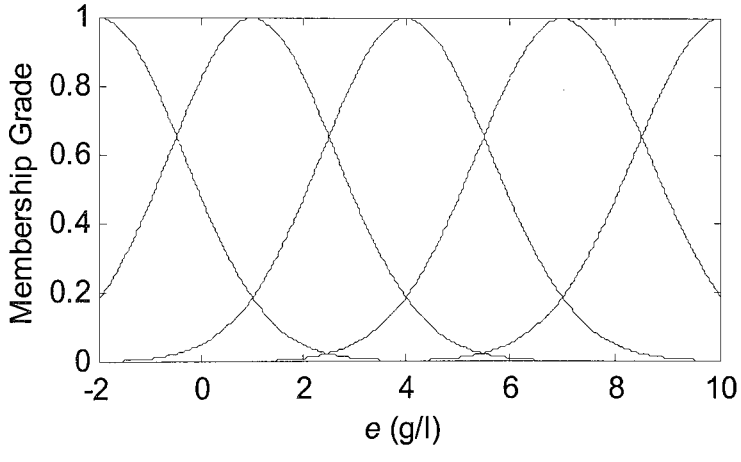

Figure 5.7. Membership functions for GRBFNN based controller of the bioreactor system

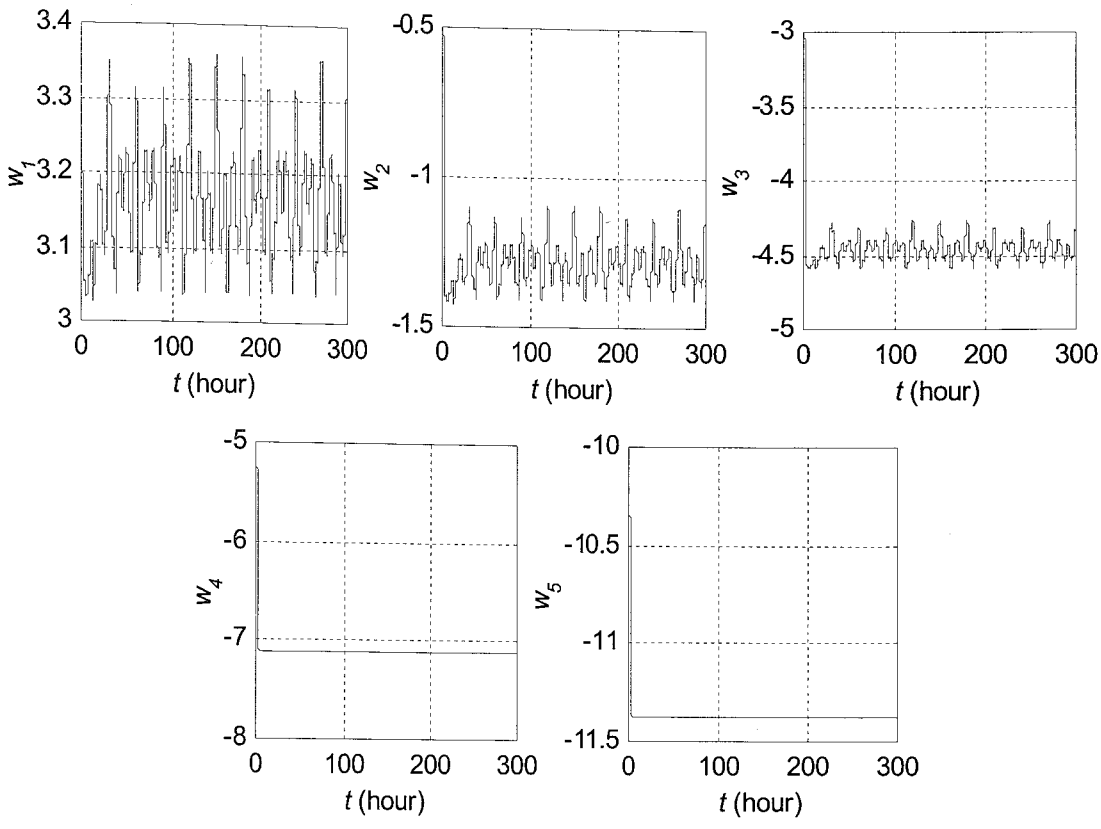

Figure 5.8. Time evolution of parameters for GRBFNN based controller of the bioreactor system

5.1.4. Application with SFS Based Controller

Another structure utilized in this work is Standard Fuzzy System, which is described by (2.21)-(2.25). Similar to the other controller structures, the settling time obtained with SFS based controller is about 1.3 hours for substrate concentration and is longer than 40 hours for cell concentration as can be seen from Figure 5.9 and Figure 5.10 respectively. Furthermore, by investigating Figure 5.10, it can be said that the produced control signal is practically applicable. The membership functions given in Figure 5.11 are utilized to partition the input (error) space on the SFS based controller. Time varying behavior of parameters for this structure is shown in Figure 5.12. As in the case of GBRFNN controller, the parameters of the SFS controller evolve bounded and no parameter drift occurs.

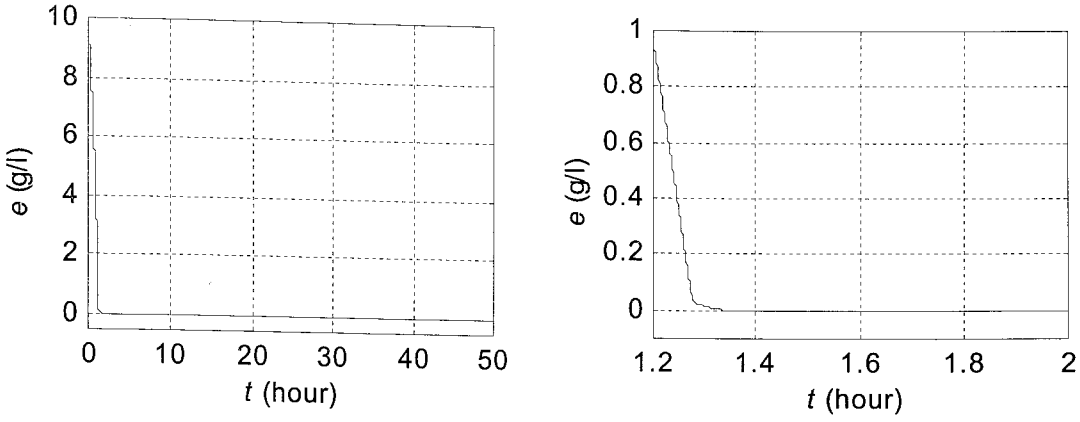

Figure 5.9. Substrate concentration error of the bioreactor system for SFS based controller

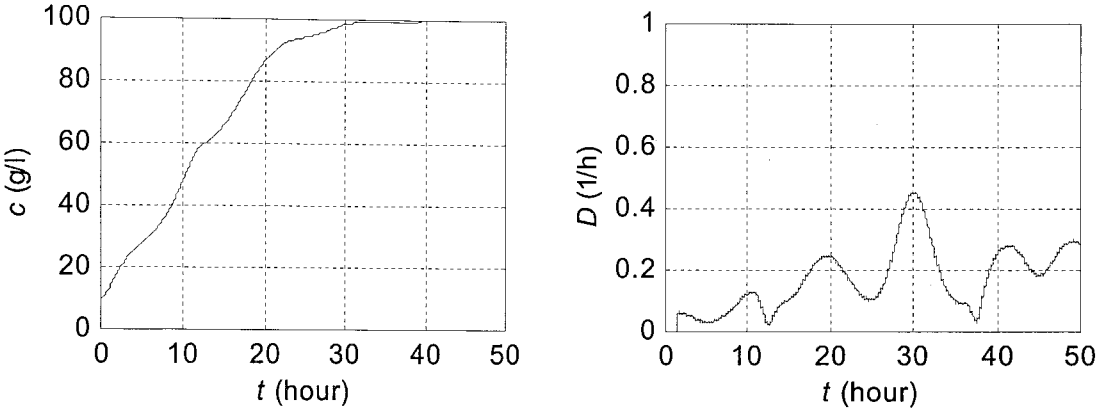

Figure 5.10. Cell concentration and dilution rate of the bioreactor system for SFS based controller

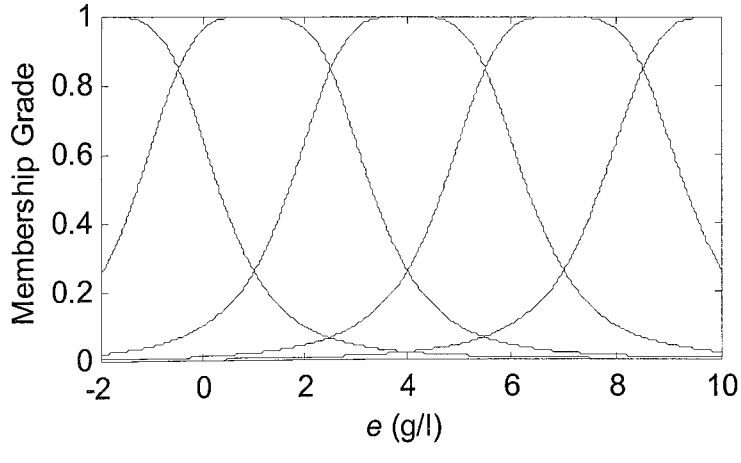

Figure 5.11. Membership functions for SFS based controller of the bioreactor system

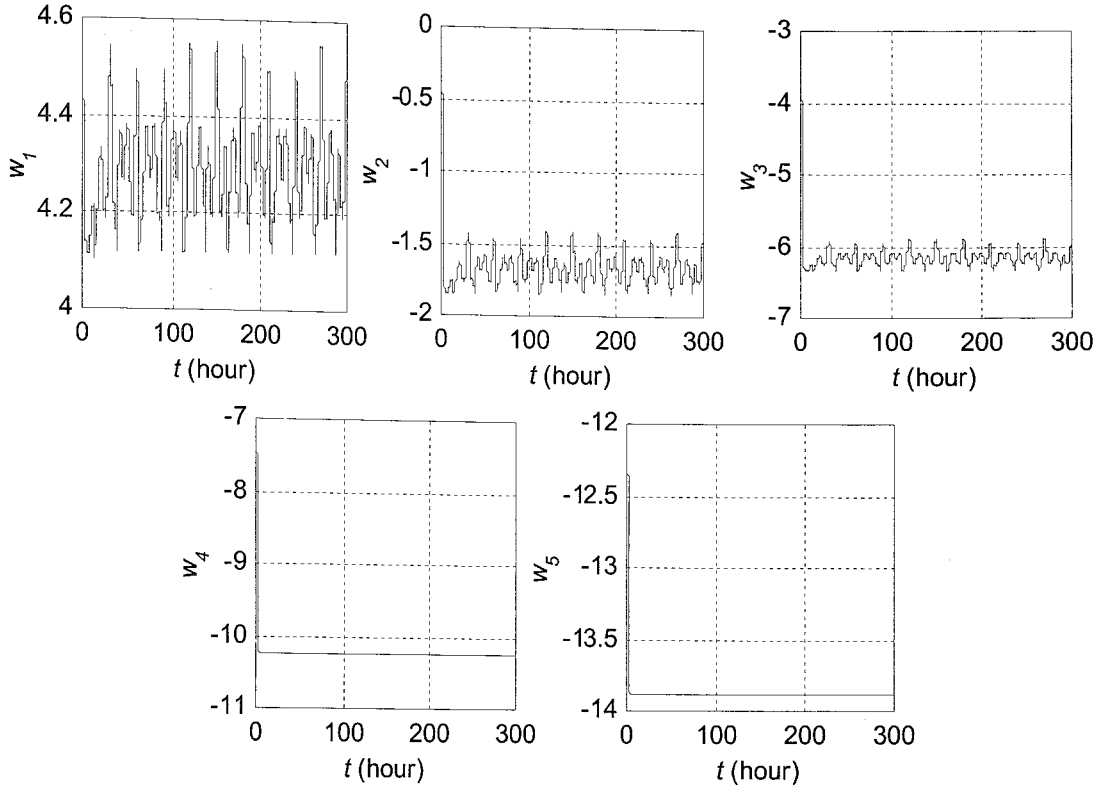

Figure 5.12. Time evolution of parameters for SFS based controller of the bioreactor system

5.1.5. Application with ANFIS Based Controller

One of the structures integrating the numeric and verbal power of intelligence is known as ANFIS and philosophically it is a suitable combination of neural and fuzzy representations of knowledge [1]. In this subsection, the performance of the ANFIS structure is elaborated. As can be seen from Figure 5.13 and Figure 5.14, in terms of the settling time metric, the response observed with ANFIS based controller is similar with the other controllers, that is, the settling time for substrate concentration is about 1.3 hours, which is an acceptable value for the process dynamics studied, and settling time for cell concentration is about 40 hours. Furthermore, the applied control signal is smooth as depicted in Figure 5.14. Because bell shaped membership functions are chosen to characterize the linguistic values, the partitioning of error space shown in Figure 5.15 is same as that of SFS controller. Lastly, the bounded evolution of controller parameters shown in Figure 5.16 confirms that the method proposed in section 4 is effective to prevent the parameter drift problem.

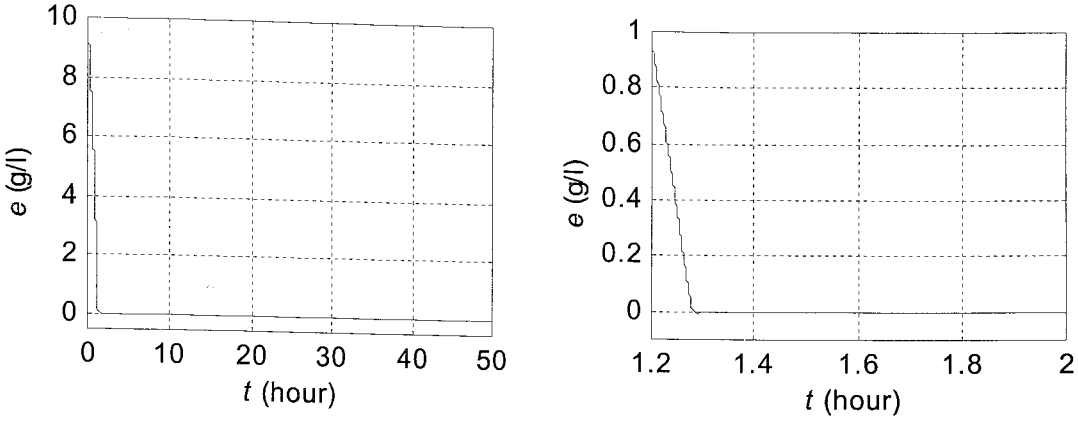

Figure 5.13. Substrate concentration error of the bioreactor system for ANFIS based controller

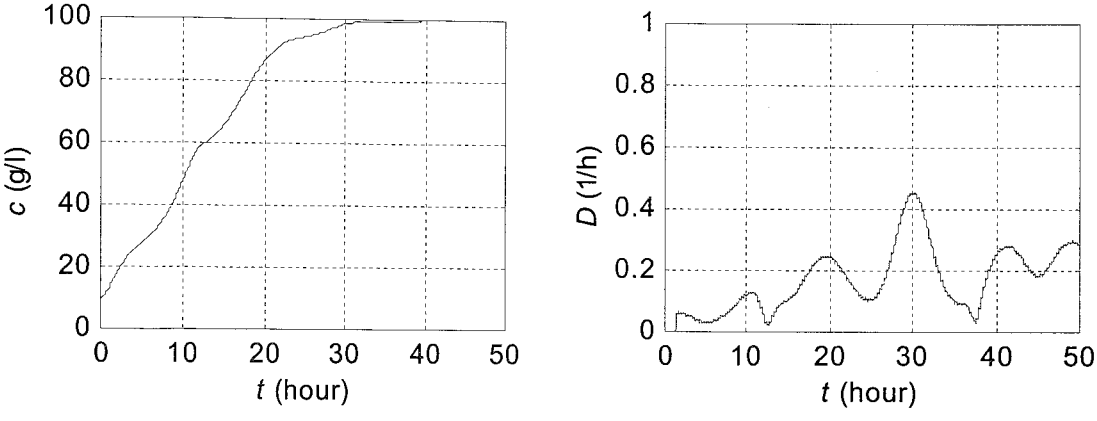

Figure 5.14. Cell concentration and dilution rate of the bioreactor system for ANFIS based controller

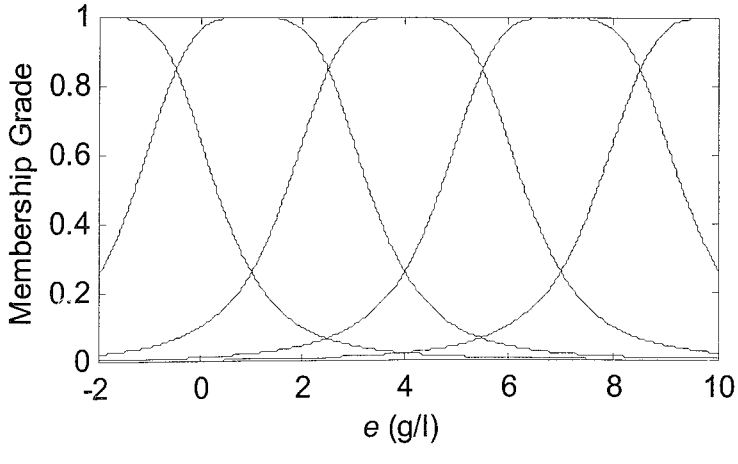

Figure 5.15. Membership functions for ANFIS based controller of the bioreactor system

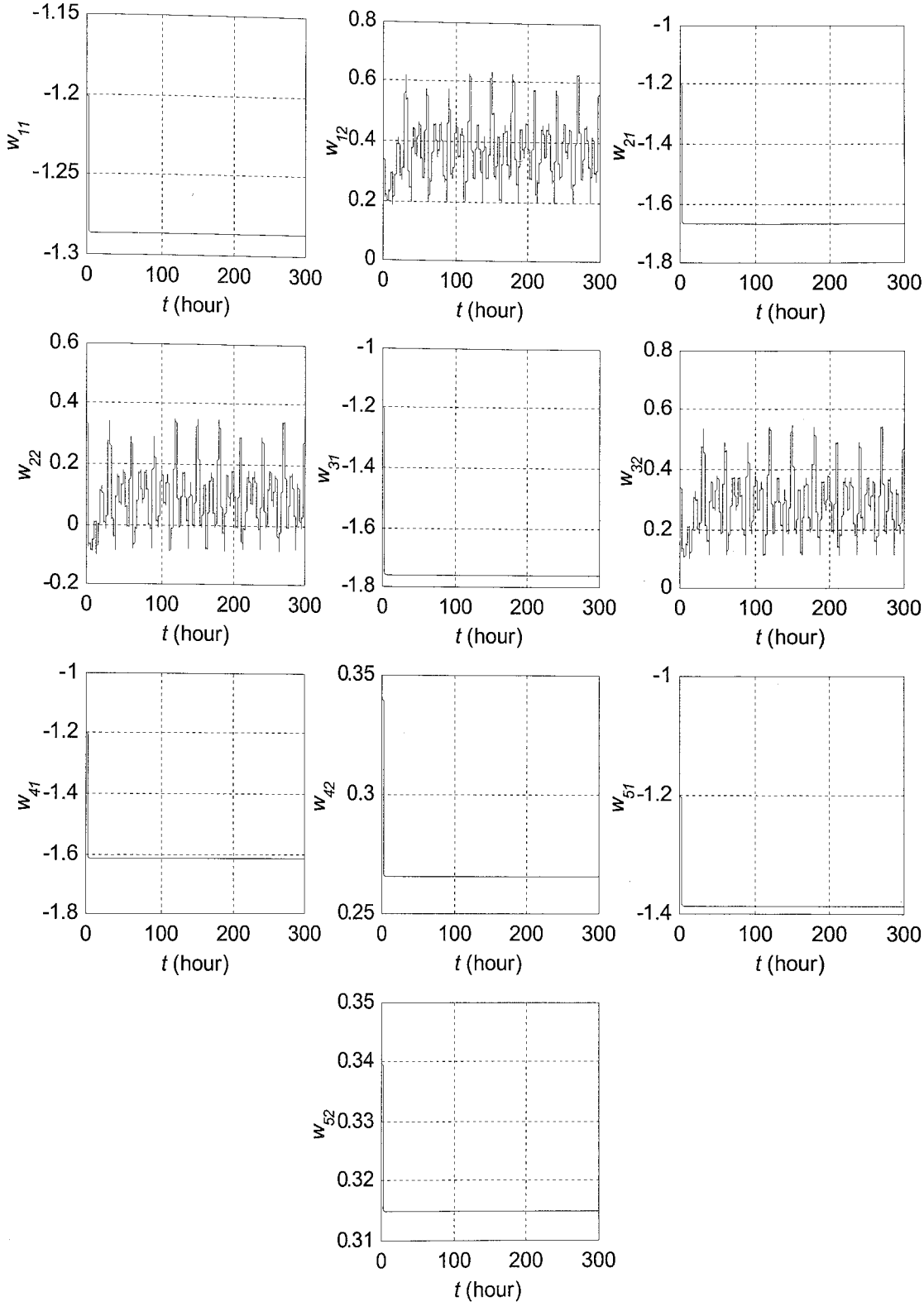

Figure 5.16. Time evolution of parameters for ANFIS controller of the bioreactor system

5.1.6. A Discussion on the Results

In the previous subsections, the parameter adaptation mechanism introduced in section 4 is utilized for control for bioreactor system described by (5.1)-(5.3). For this purpose the substrate concentration is chosen as the free variable and controlled through dilution rate. Based on simulation results, it can be said that the proposed adaptation mechanism is effective for control problem of the bioreactor system and there is not a significant difference between performances of different controllers. When the latter is taken into account it can be said that the ADALINE architecture will be the most suitable controller because it performs as good as the other controllers while the computational complexity for this structure is the lowest among the all architectures. For all of the structures studied, the error at the output of the plant converges to zero error-level very rapidly, and thus, undesirable byproduct formation, which occurs when the substrate concentration is above a limiting value, is prohibited. Moreover, while the system is in steady state, the error at the output of the plant remains almost zero in spite of time variation of plant parameters and presence of Gaussian noise on measurement of substrate concentration. This result states that the proposed control structure is robust against the uncertainties. Another advantage of the proposed method is that the measurement of cell concentration, which is not practical to realize, is not required. Therefore, the proposed control system does not necessitate estimation of the immeasurable state by use of an additional observer, which obviously increases cost and complexity of the system. Lastly, results for all controllers prove that the method proposed in section 4, which is used to prevent parameter drift problem, woks as expected. That is to say, the parameters of all controllers remains bounded.

5.2. Cement Mill Process

Production of portland cement is a highly complicated process, and composed of several stages [18]. Each stage of this process is shown in Figure 5.17. As can be seen from the figure, first, raw materials, the most commonly used ingredient is limestone coupled with much smaller quantities of clay and sand, are fed into the crusher, where they are broken into small pieces. Then, the stone from the crusher is blended with the other materials, including sand, mill scale or bauxite to achieve the right mix of calcium, silica,

aluminum and iron. The resulting material is ground to a fine powder in the raw mill. After a homogenization stage, the raw feed coming from the raw mill goes into the kiln, where it is heated up to 1450° C to produce the clinker which is later cooled and ground into a fine cement powder in the cement mill. In this work, the aim is to control the cement mill circuit which grinds clinker into fine cement powder.

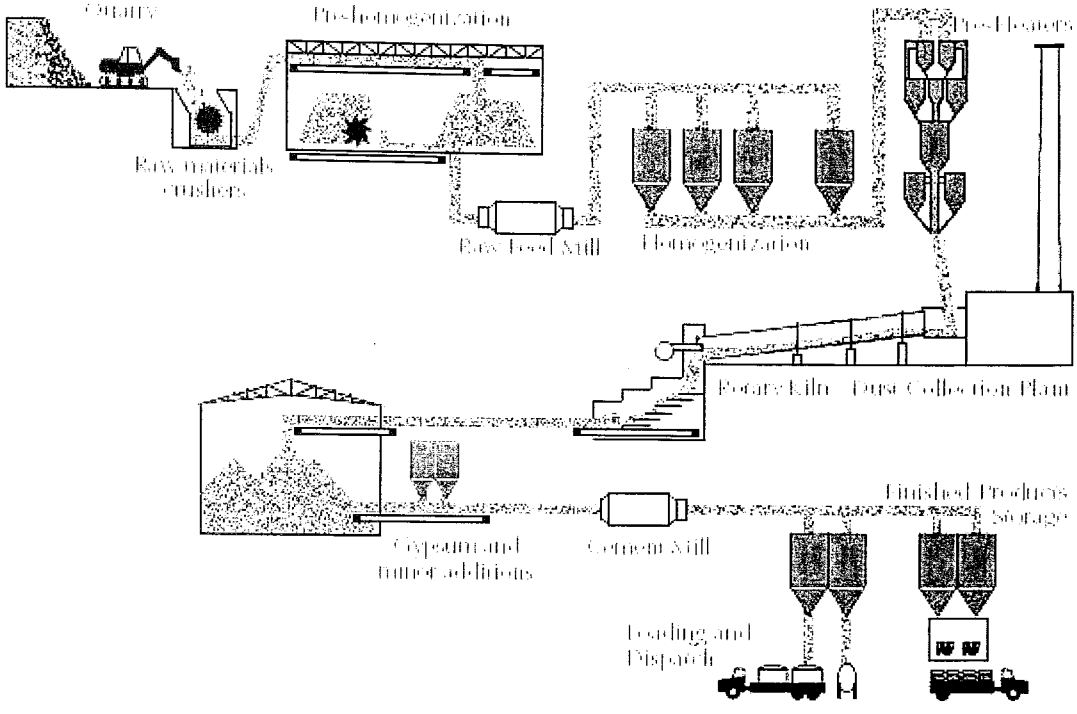

Figure 5.17. Production of portland cement

The schematic representation of a cement mill circuit is shown in Figure 5.19. As depicted in the figure, the raw material, clinker, is fed into the mill, where it is ground into fine powder with the rotational movements. Then, the resulting material is passed through the classifier and separated into two classes, product and tailing. While the product leaves the system, the rejected material, tailings, is fed back into the mill for further grinding. An experimentally verified dynamic nonlinear model for this process is proposed by Magni, et al. [19]. The differential equations describing the model are given as

$$T_f \dot{y}_f = -y_f + (1-\alpha(z,v,d))\varphi(z,d) \tag{5.8}$$

$$\dot{z} = -\varphi(z,d) + u + y_r \tag{5.9}$$

$$T_r \dot{y}_r = -y_r + \alpha(z,v,d)\varphi(z,d) \tag{5.10}$$

$$\alpha(z,v,d) = \frac{\varphi^m v^n}{K_\alpha + \varphi^m v^n} \tag{5.11}$$

$$\varphi(z,d) = \max\{0;(-dK_{\varphi 1}z^2 + K_{\varphi 2}z)\} \tag{5.12}$$

where y_f is the product flow rate (Tons/h), z is the load in the mill (Tons), y_r is the tailings flow rate (Tons/h), u is the feed flow rate (Tons/h), v is the classifier speed (r/min), and d represents the hardness of the material inside the mill with respect to its nominal value.

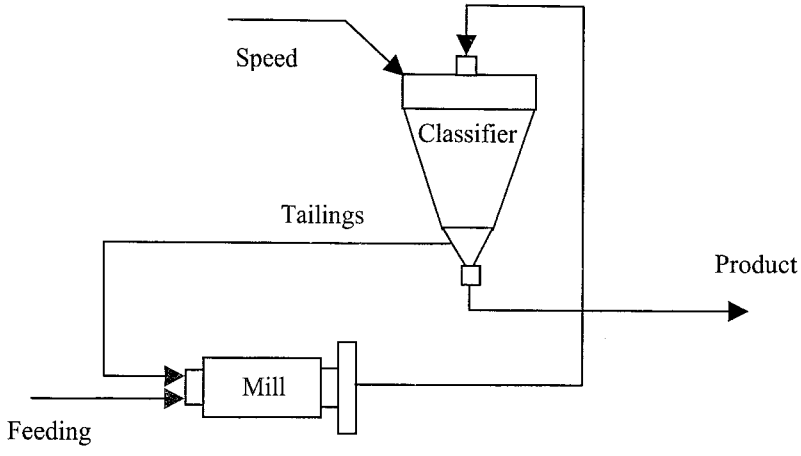

Figure 5.18. Cement mill circuit

5.2.1. Control of Cement Mill Process

This section is devoted to control problem of the cement mill circuit. As can be seen from the dynamical equations describing the process, this system has highly nonlinear characteristics with strong couplings between its states. Therefore, it constitutes a good candidate to evaluate performance of the control mechanism introduced in section 4. In what follows, it is discussed that how this control approach can be applied to the cement mill process.

At first glance, it will be helpful to decide on configuration of the control system. If one investigates (5.8)-(5.12), it can be seen that the system involves three state variables and two control inputs. Thus, two of the states must be chosen as free variables and controlled through system inputs while the remaining state must be the depended variable of the others. In this study, mill load and product flow rate are designated as free variables. Based on this choice, in order to achieve the control objective, u-z and v-y_f input-state pairs, each pair can be considered as a first order system, are controlled using separate controllers by regarding the couplings between the states as disturbances.

Unfortunately, the control mechanism introduced in section 4 cannot be adopted directly to control the first order subsystems defined above. There are some problems associated with application of the original control structure. First of all, although, feed flow rate, rate of the raw material fed into the mill, cannot take negative values, there is not such a constraint in the original algorithm. This inconsistency may lead to problems in applications, and hence, must be handled appropriately. In order to alleviate this difficulty, output of the associated controller is passed through a saturation function which produces zero for negative values of the control signal. By doing this, one can assure that a valid feed flow command signal is applied to actuators.

Another difficulty arises in the second subsystem having v-y_f pair as input and output respectively. This system could not be stabilized by constructing the original control loop shown in Figure 4.4. The reason behind this instability problem can be understood by investigating the behavior of the product flow rate with respect to classifier speed input as follows. The material leaving the mill, rate of which is denoted by φ, is separated into two parts, tailings with ratio α and end product with ratio 1-α, by the classifier. Therefore, the second term in the right hand side of equation (5.8) gives the rate of the material classified as end product. If one investigates (5.11), it can be seen that because α is monotonically increasing function of v for $v \geq 0$, the term in (5.8) decreases with increasing values of v. This result contradicts with assumption given in [10], which states that if the actual state is below the desired state, it is required to increase the input or vice versa.

In the light of the facts given above, a mapping is utilized between controller output and classifier speed command signal in order to make the overall system, including the

mapping, satisfy the conditions given in [10]. This mapping is shown in Figure 5.19 and can be formulated as follows.

$$v = \begin{cases} mx - mn, & x < n \\ 0, & x \geq n \end{cases} \tag{5.13}$$

where m=-1 and n=600. To be able to show effect of the mapping, graphs of $(1-\alpha)\varphi$ term for different constant values of φ are given in Figure 5.20 and 5.21 as a function of classifier speed and controller output respectively. As can be clearly seen from these figures, although, $(1-\alpha)\varphi$ is a decreasing function of v, it is an increasing function of controller output. Therefore, the conditions given in [10] are satisfied, and hence, the system can be stabilized. At that point, one may ask the reason behind the choice of the parameters (m, n) of the mapping given in 5.13. In fact, the values given above are not unique. As long as m and n are known constants, and sign of m is negative, it is possible to attain similar system response for different m and n values by adjusting parameters of the controller appropriately. For example, for higher values of m, the sensitivity of v with respect to controller output will decrease. On the other hand, if one increases the uncertainty bound term (K_v), controller will be more sensitive to error signal, that is, for negative (positive) values of error, output of the controller will increase (decrease) faster. Therefore, by an appropriate choice of K_v, one can keep sensitivity of the classifier speed command signal with respect to error approximately same for different values of m. In other words, the choice of m will not change dynamics of closed loop system considerably as long as uncertainty bound term set to a proper value. Furthermore, because changes in the value of n is simply corresponds to a shift in classifier speed, one can easily keep overall behavior of v same by adjusting initial parameters of the controller appropriately.

In the following subsections, simulation results for different intelligent architectures are investigated. Simulations are performed in MATLAB 5.3 environment and step size is chosen as 0.001 hours. For all architectures, initial states of the plant are set to zero and target set points for mill load and product flow rate are chosen as 55 tons and 100 tons/hour respectively. Moreover, to emphasize the robustness of the control system, the hardness parameter d is changed from 1 to 1.25 at t=10 hours and both state measurements are corrupted with Gaussian noise having zero mean and variance equal to 1.64e-6. To

eliminate chattering problem, sign functions utilized in the parameter adaptation algorithms of both controllers are replaced with the same approximation function given in subsection 5.1.1. Furthermore, in the adaptation algorithms, uncertainty bound parameter for the first subsystem (K_u) is chosen as 300, and that of second sub system (K_v) is set to 750. Moreover, each controller changes its operating mode as describe in section 4 when the absolute value of error is less than 0.2 in order to eliminate unbounded parameter evolution problem.

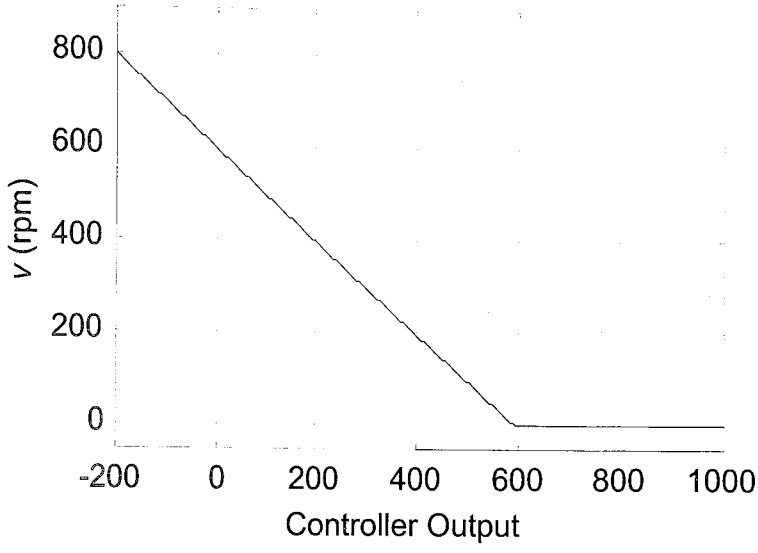

Figure 5.19. Graph of mapping utilized at the output of the controller

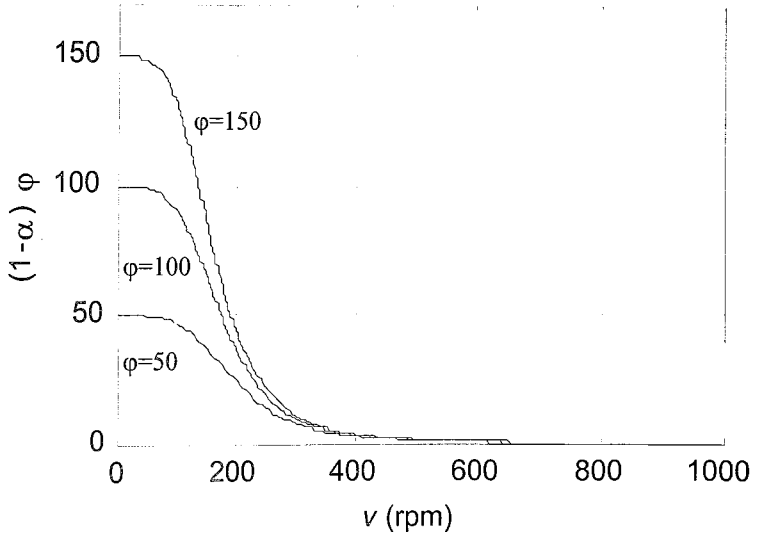

Figure 5.20. (1-α)φ term as a function of classifier speed

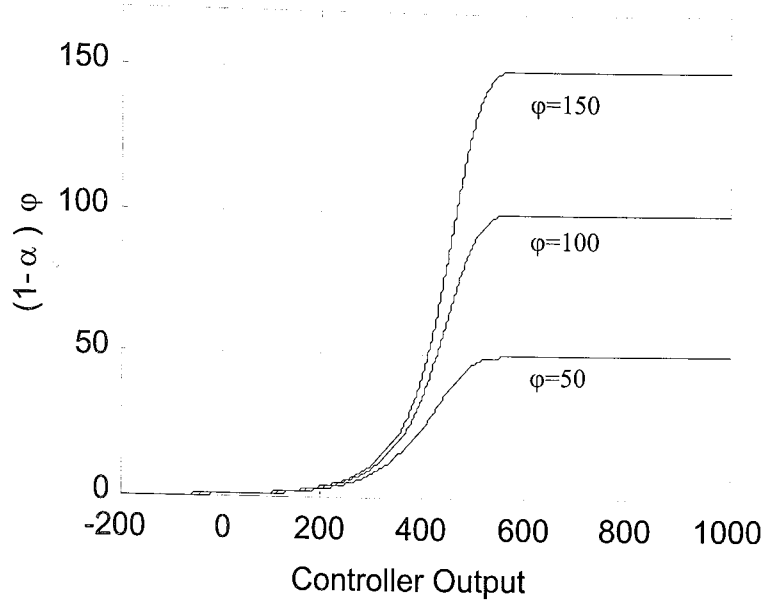

Figure 5.21. (1-α)φ term as a function of controller output

5.2.2. Application with ADALINE Based Controller

Performance of ADALINE based controller is investigated in this subsection. As can be seen from Figure 5.22, both errors of mill load and product flow rate converges to zero very rapidly. Furthermore, the remaining state of the system, tailing flows rate, decreases sharply form 470 tons/hour to 385 tons/hour at t=10 hours due to the change in the hardness parameter as shown in Figure 5.23. Inputs of the system, through which free states are controlled, are shown in Figure 5.24. As depicted in the figure, the feed flow rate increases up to 185 tons/hour very rapidly in transient phase, and it is very active in steady state due to disturbances. Moreover, both feed flow rate and classifier speed command signals change very fast at t=10 hours in order to compensate uncertainty in the hardness parameter. Although, time scale of the system is very large, the rapid transitions in inputs may not be followed closely by the actuators. Therefore, some difficulties may arise in the applications based on ADALINE architecture. Lastly, the time evolutions of controller parameters are shown in Figure 5.25 and Figure 5.26. As these figures suggests, parameters of both controllers evolves bounded.

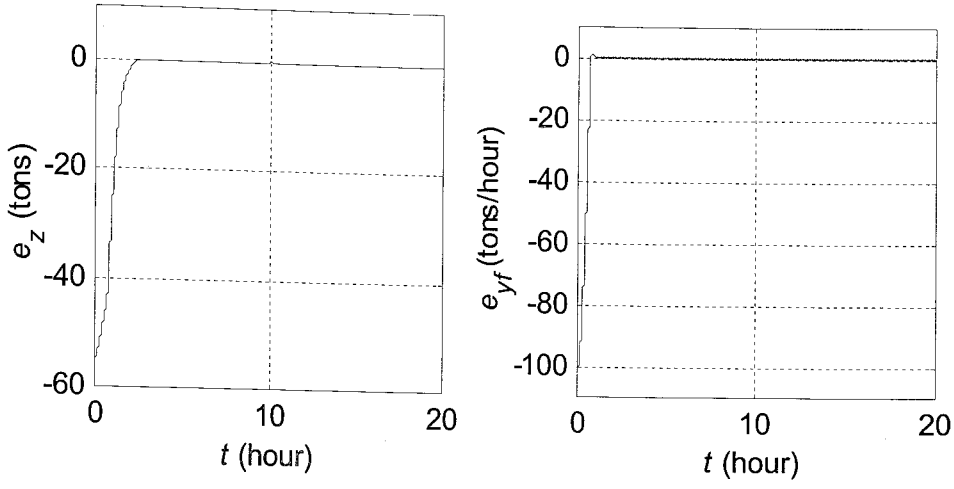

Figure 5.22. Mill load and product flow rate errors of the cement mill process for ADALINE based controller

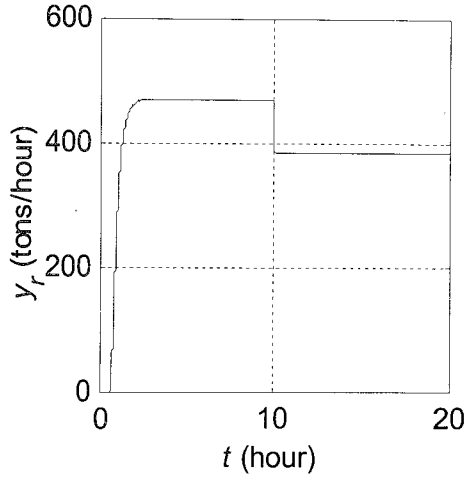

Figure 5.23. Time evolution of tailings flow rate of the cement mill process for ADALINE based controller

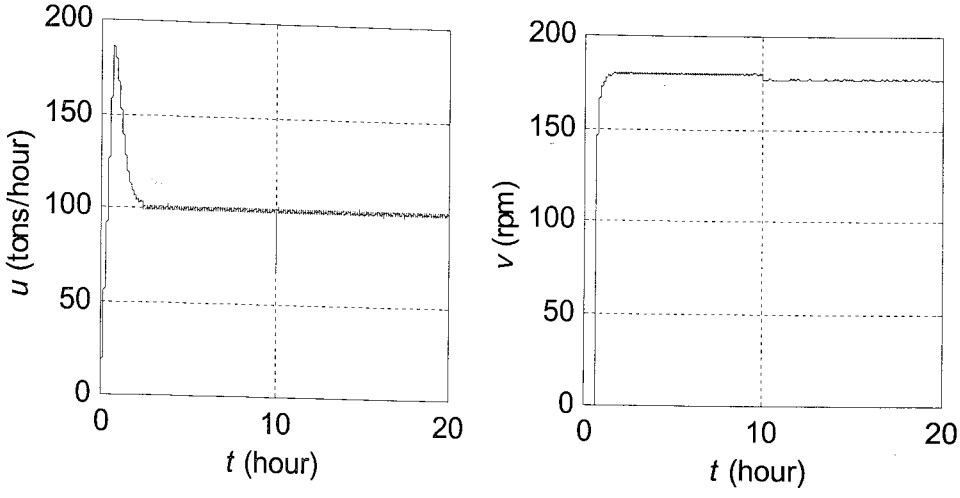

Figure 5.24. Time evolutions of feed flow rate and classifier speed of the cement mill process for ADALINE based controller

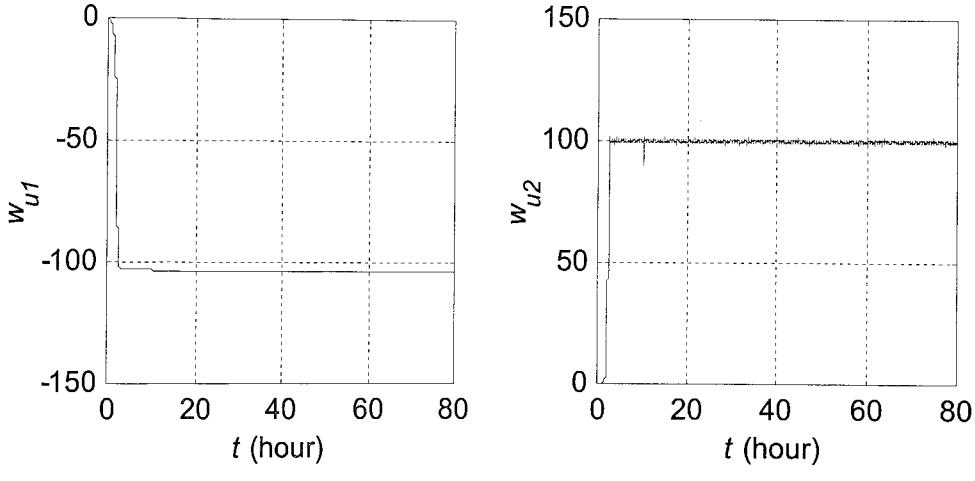

Figure 5.25. Time evolution of parameters for the first ADALINE based controller of the cement mill process

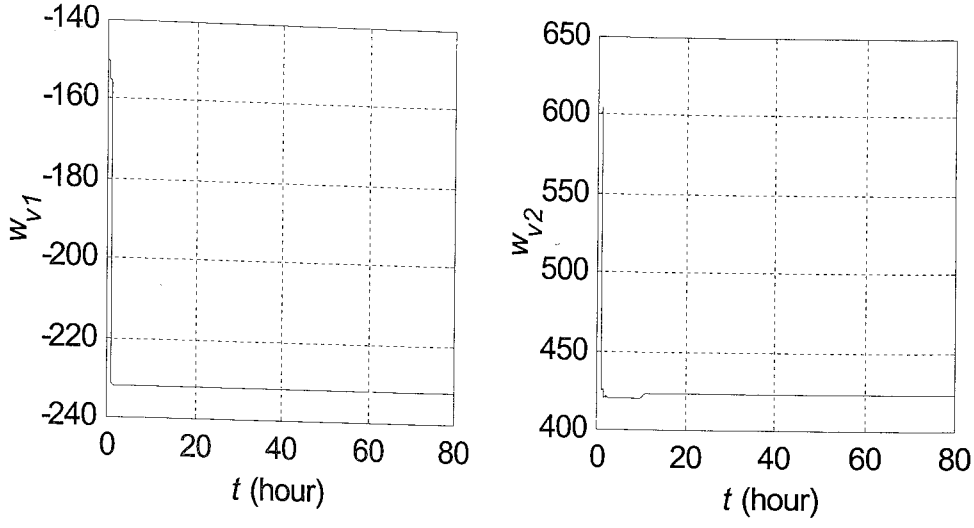

Figure 5.26. Time evolution of parameters for the second ADALINE controller of the cement mill process

5.2.3. Application with GRBFNN Based Controller

The simulation results of GRBFNN based controller are discussed in this subsection. The time evolutions of state errors given in Figure 5.27 show that the proposed method is successful in stabilizing the closed loop system. The errors of mill load and product flow rate converge to their steady state values in approximately 2.2 hours and 1 hour respectively. Moreover, state errors are not affected from the absurd change in the material hardness considerably, which proves robustness of the proposed method. To be able to compensate this parameter change, controllers produce fast input signals at t=10 hours as shown in Figure 5.29. Especially, the behavior of classifier speed at this time instant causes the tailings flow rate, time evolution of which is given in Figure 5.28, to decrease dramatically. Gaussian activation functions of controllers are shown in Figure 5.30. The ranges of activation functions are chosen so that both controllers work properly in the operating range of the system. Lastly, the time evolutions of parameters for first and second controllers are illustrated in Figure 5.31 and Figure 5.32 respectively. As these figures suggest, each parameter converges to a steady state value. In other words, method proposed in section 4 is successful in preventing parameter drift problem.

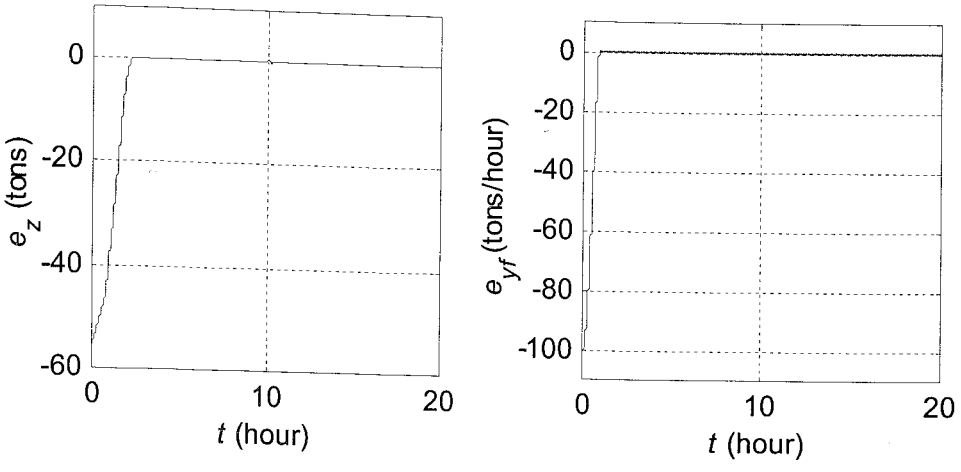

Figure 5.27. Mill load and product flow rate errors of the cement mill process for GRBFNN based controller

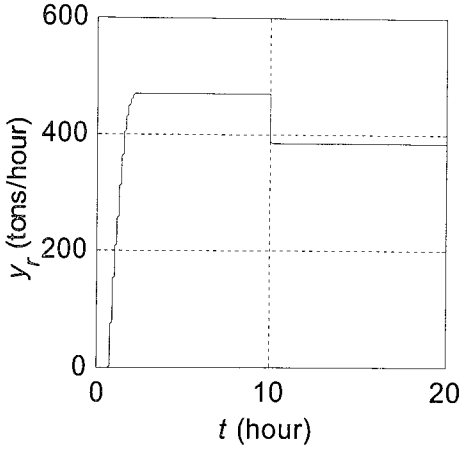

Figure 5.28. Time evolution of tailings flow rate of the cement mill process for GRBFNN based controller

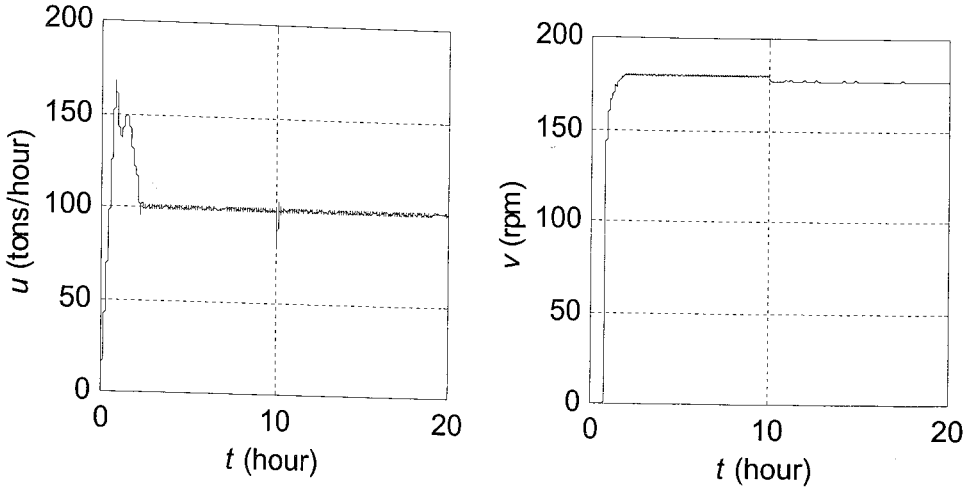

Figure 5.29. Time evolutions of feed flow rate and classifier speed of the cement mill process for GRBFNN based controller

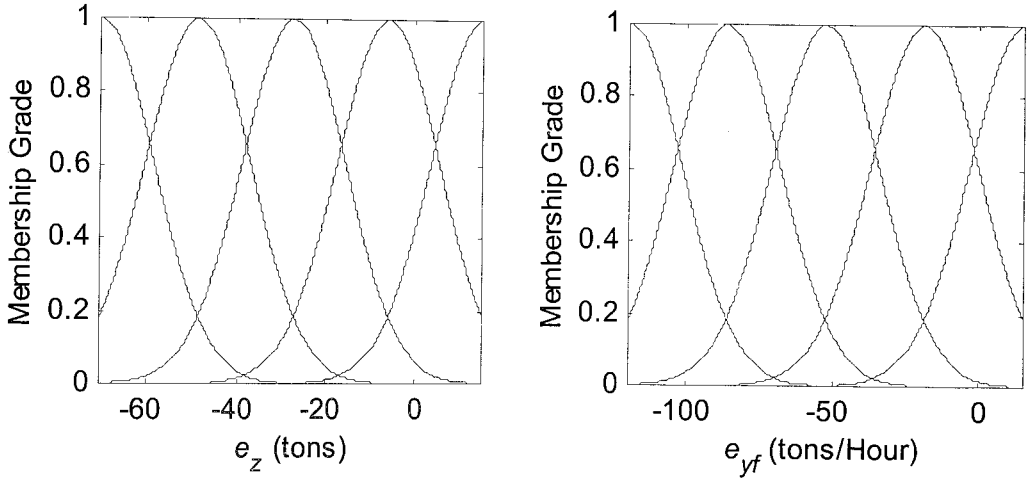

Figure 5.30. Neuron activation functions of GRBFNN architecture for the cement mill process

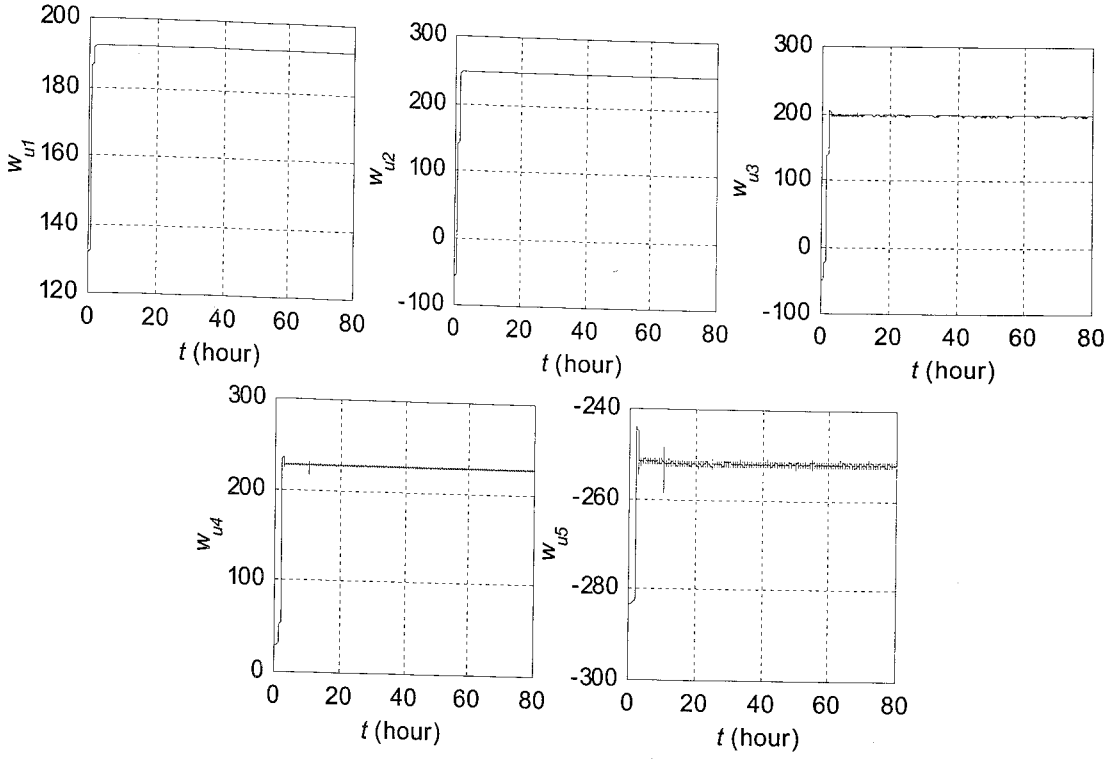

Figure 5.31. Time evolutions of parameters for the first GRBFNN based controller of the cement mill process

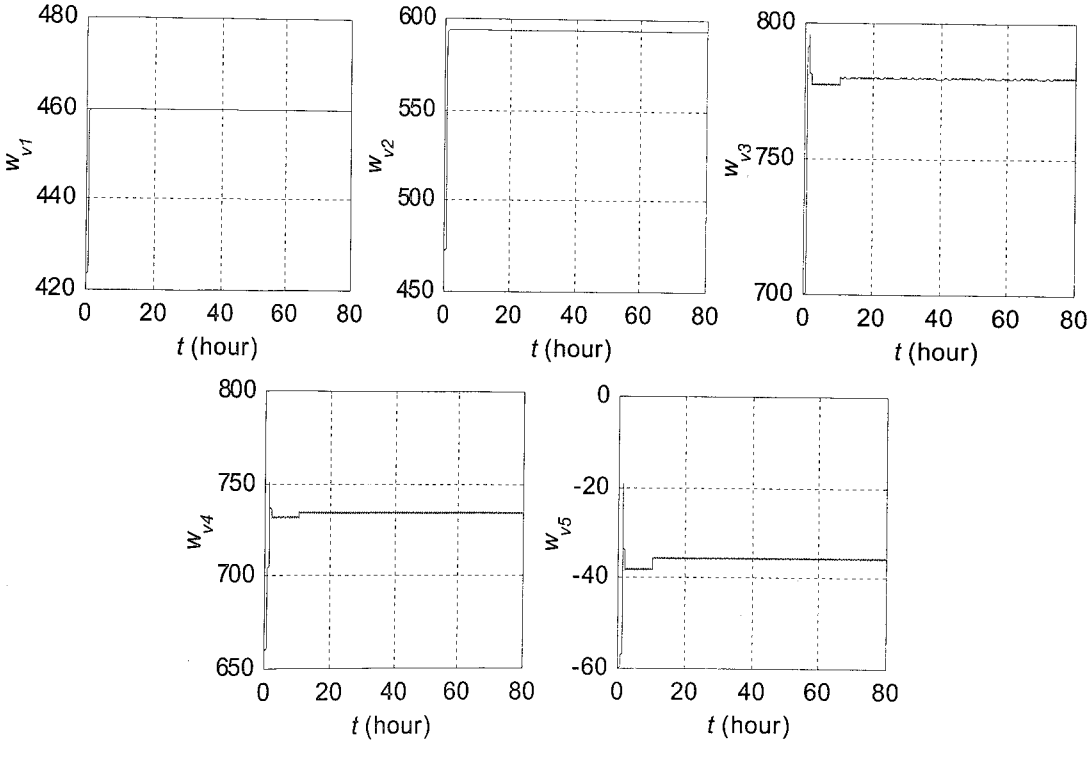

Figure 5.32. Time evolutions of parameters for the second GRBFNN based controller of the cement mill process

5.2.4. Application with SFS Based Controller

The aim of this subsection is to evaluate performance of SFS based controller. The state errors and time evolution of tailings flow rate for this intelligent architecture are given in Figure 5.33 and Figure 5.34 respectively. If one compares these results with that of the previously investigated structures, it can be seen that there is no significant difference between them in terms of both transient and steady state behavior. Moreover, as depicted in Figure 5.35, although, there is not a remarkable improvement in classifier speed input, the peak value of the feed flow rate is less than that of previously investigated architectures, and its steady state behavior is smooth in spite of presence of measurement noise. The partitioning of error space of both controllers is shown in Figure 5.36. Here, bell shaped membership functions are utilized in order to granulate the error values. Moreover, the centers of membership functions are chosen so that they cover the whole range of error signals. The time evolutions of system controller parameters are given in Figure 5.37 and Figure 5.38. As these figures suggest, all of the parameters remains bounded within the given time interval.

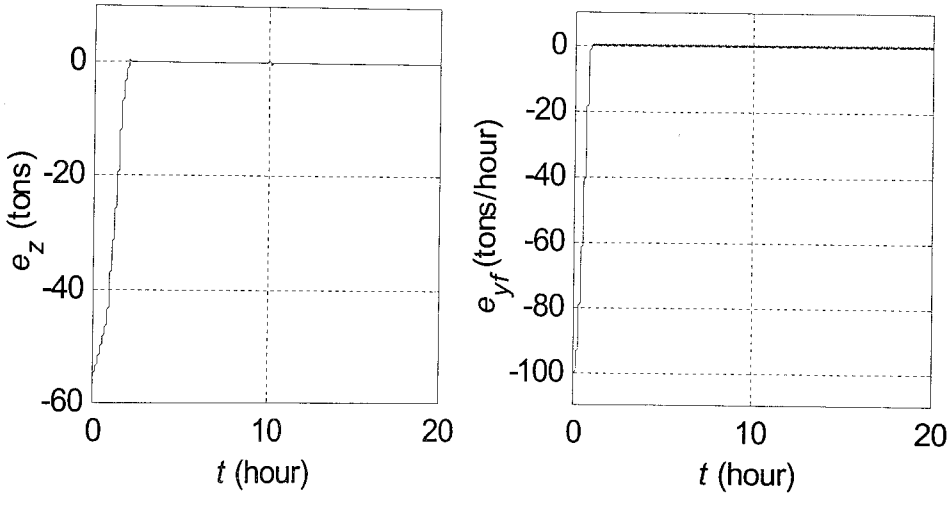

Figure 5.33. Mill load and product flow rate errors of the cement mill process for SFS based controller

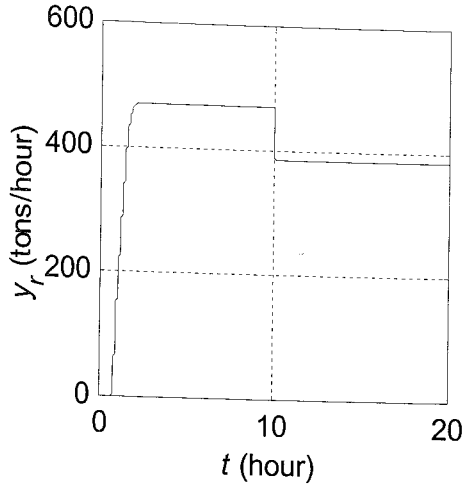

Figure 5.34. Time evolution of tailings flow rate of the cement mill process for SFS based controller

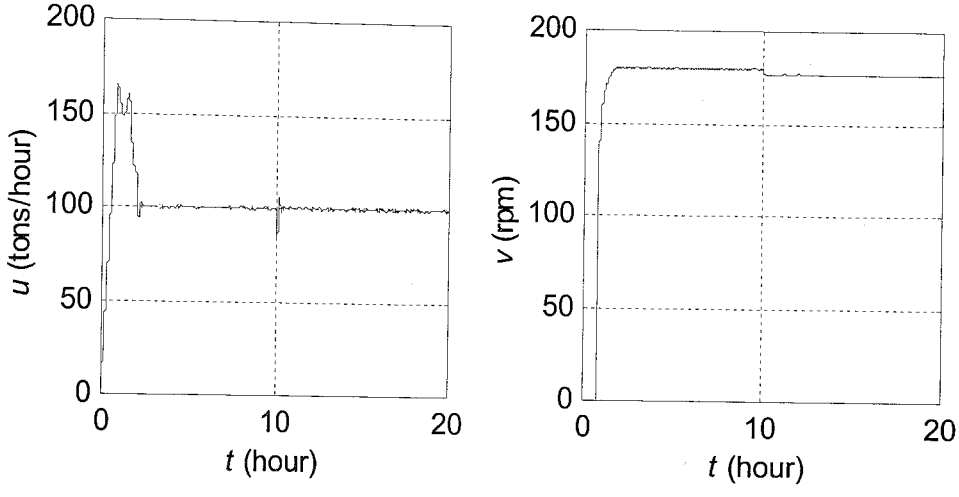

Figure 5.35. Time evolutions of feed flow rate and classifier speed of the cement mill process for SFS based controller

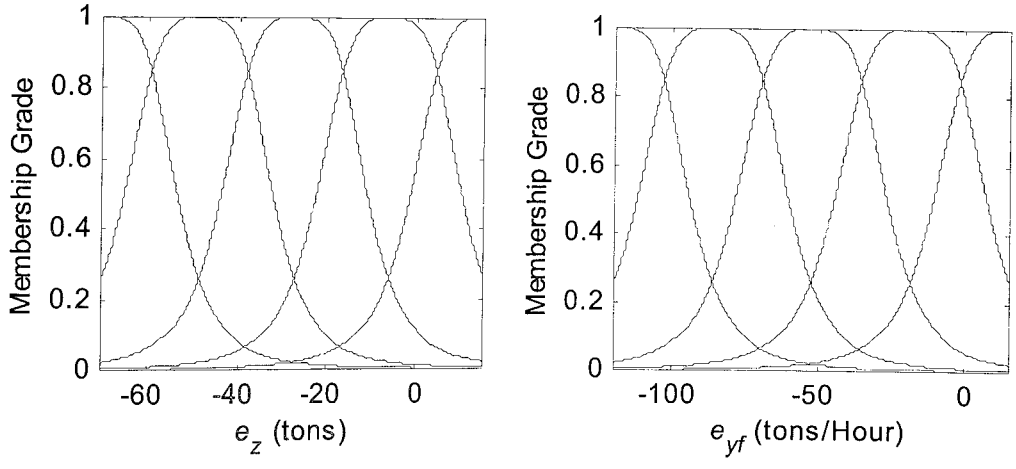

Figure 5.36. Membership functions of SFS controllers for the cement mill process

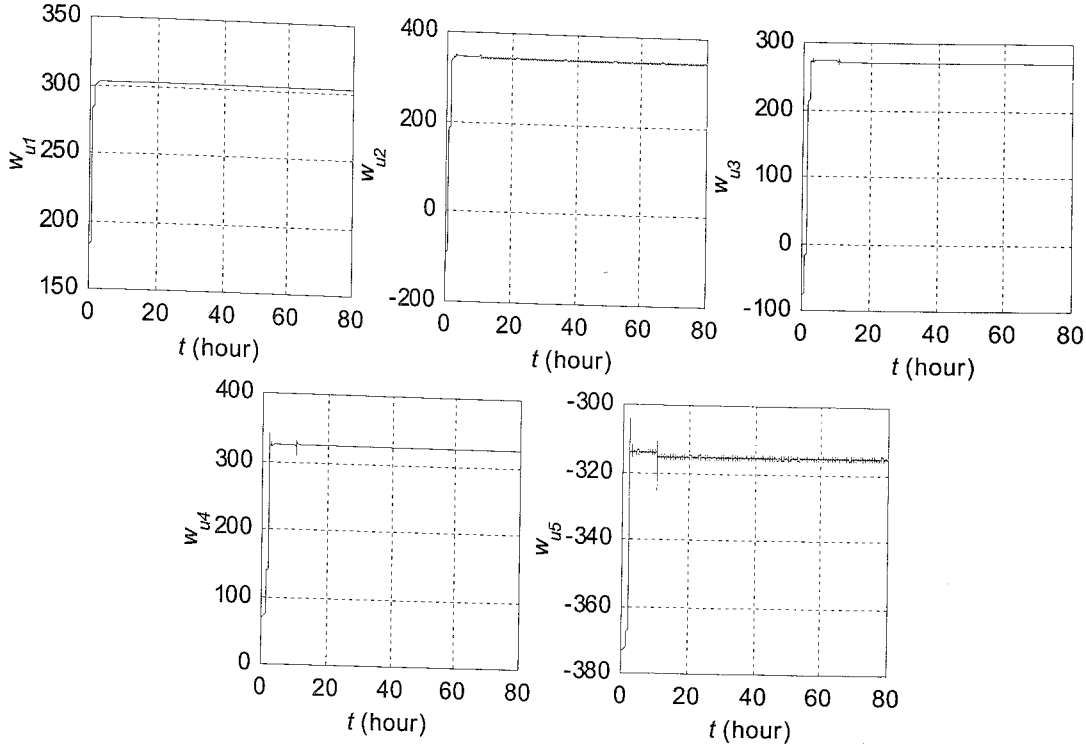

Figure 5.37. Time evolutions of parameters for the first SFS based controller of the cement mill process

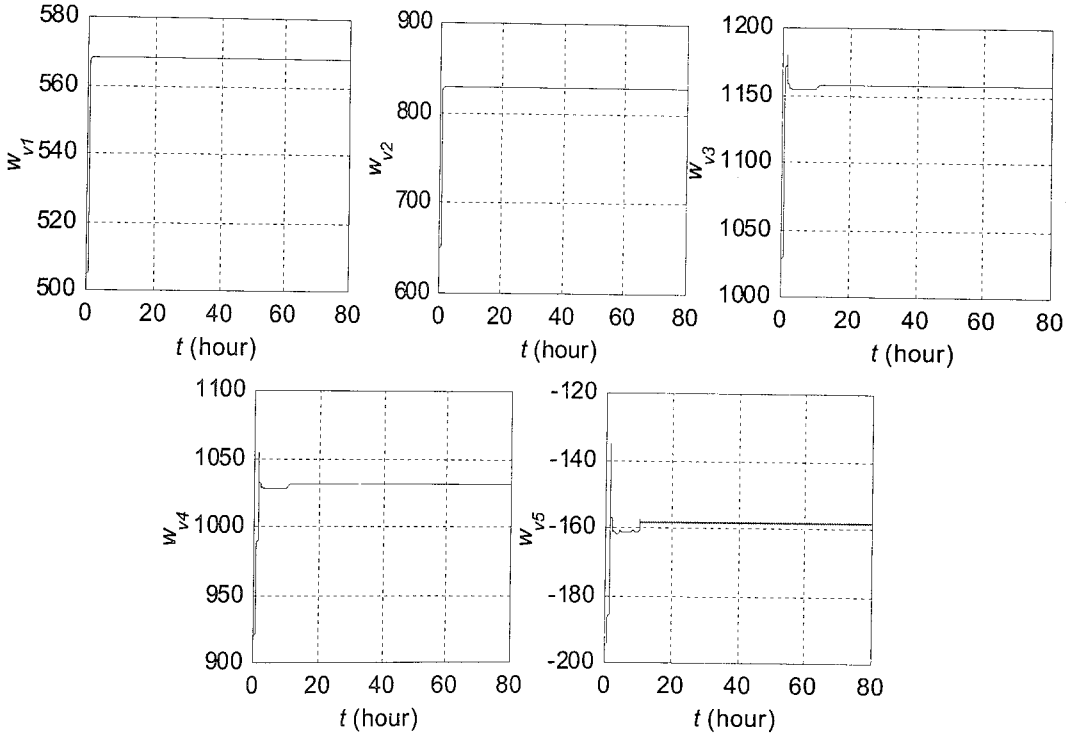

Figure 5.38. Time evolutions of parameters for the second SFS based controller of the cement mill process

5.2.5. Application with ANFIS Based Controller

ANFIS based controller is the last intelligent architecture utilized for control of the cement mill circuit. As can be seen from Figure 5.39, state errors for this controller become acceptably small after a sort transient phase and stays in the vicinity of zero-error level thereafter in spite of the parametric uncertainty, existence of measurement noise and couplings between system states. Although, change in the hardness parameter has not a considerable effect on mill load and product flow rate, tailings flow rate decreases approximately 85 tons/hour at t=10 hours as shown in Figure 5.40. The behaviors of feed flow rate and classifier speed inputs, which are demonstrated in Figure 5.41, shows that the peak value of feed flow rate is smaller than the maximum flow rate of ADALINE based controller. Moreover, the controller of the first subsystem applies a very large magnitude command signal in order to keep states of the system at their desired values when the hardness parameter changes. The membership functions of all rules along with the linear parameters of rule consequences are given in Figure 5.42, Figure 5.43 and Figure 5.44. As the later figures suggests no parameter drift occurs due to the modification in the parameter adaptation mechanism.

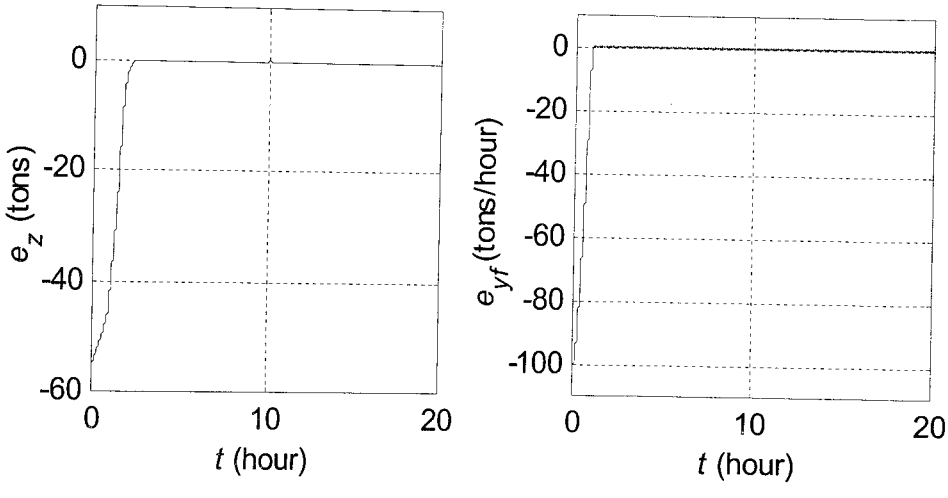

Figure 5.39. Mill load and product flow rate errors of the cement mill process for ANFIS based controller

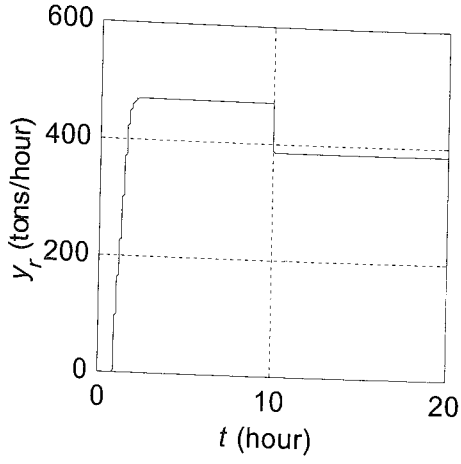

Figure 5.40. Time evolution of tailings flow rate of the cement mill process for ANFIS based controller

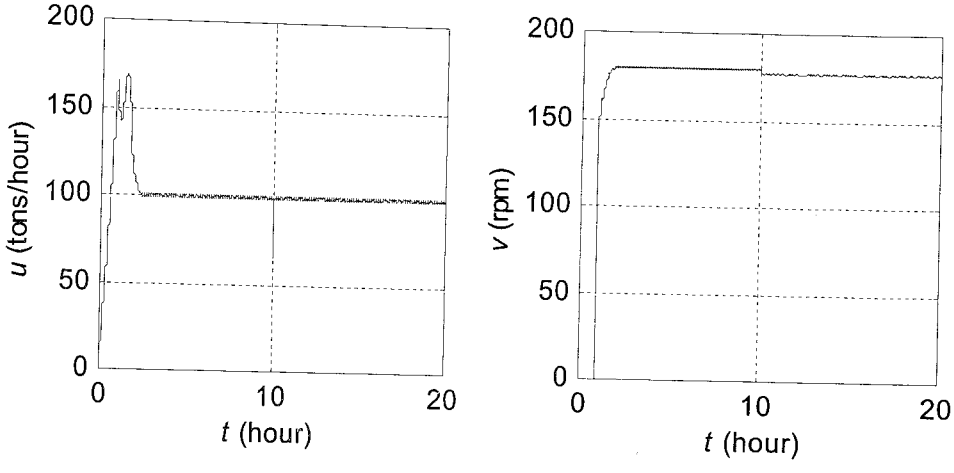

Figure 5.41. Time evolutions of feed flow rate and classifier speed of the cement mill process for ANFIS based controller

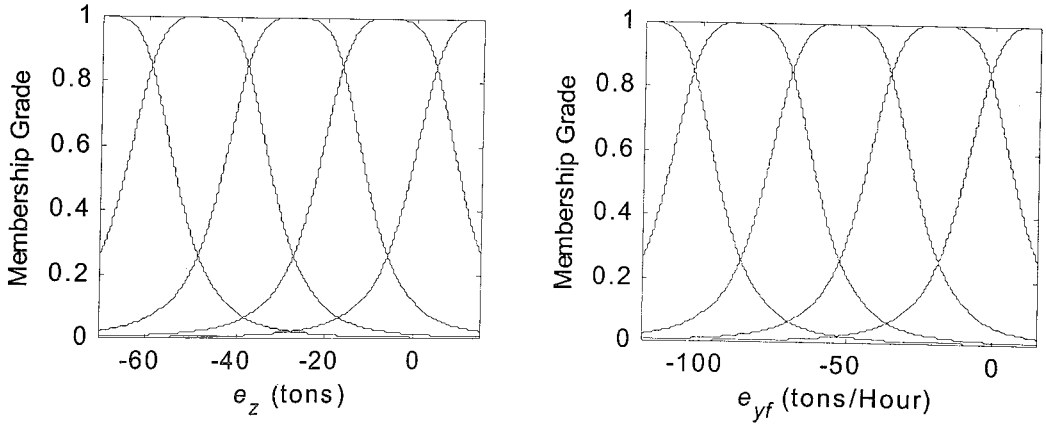

Figure 5.42. Membership functions of ANFIS controllers

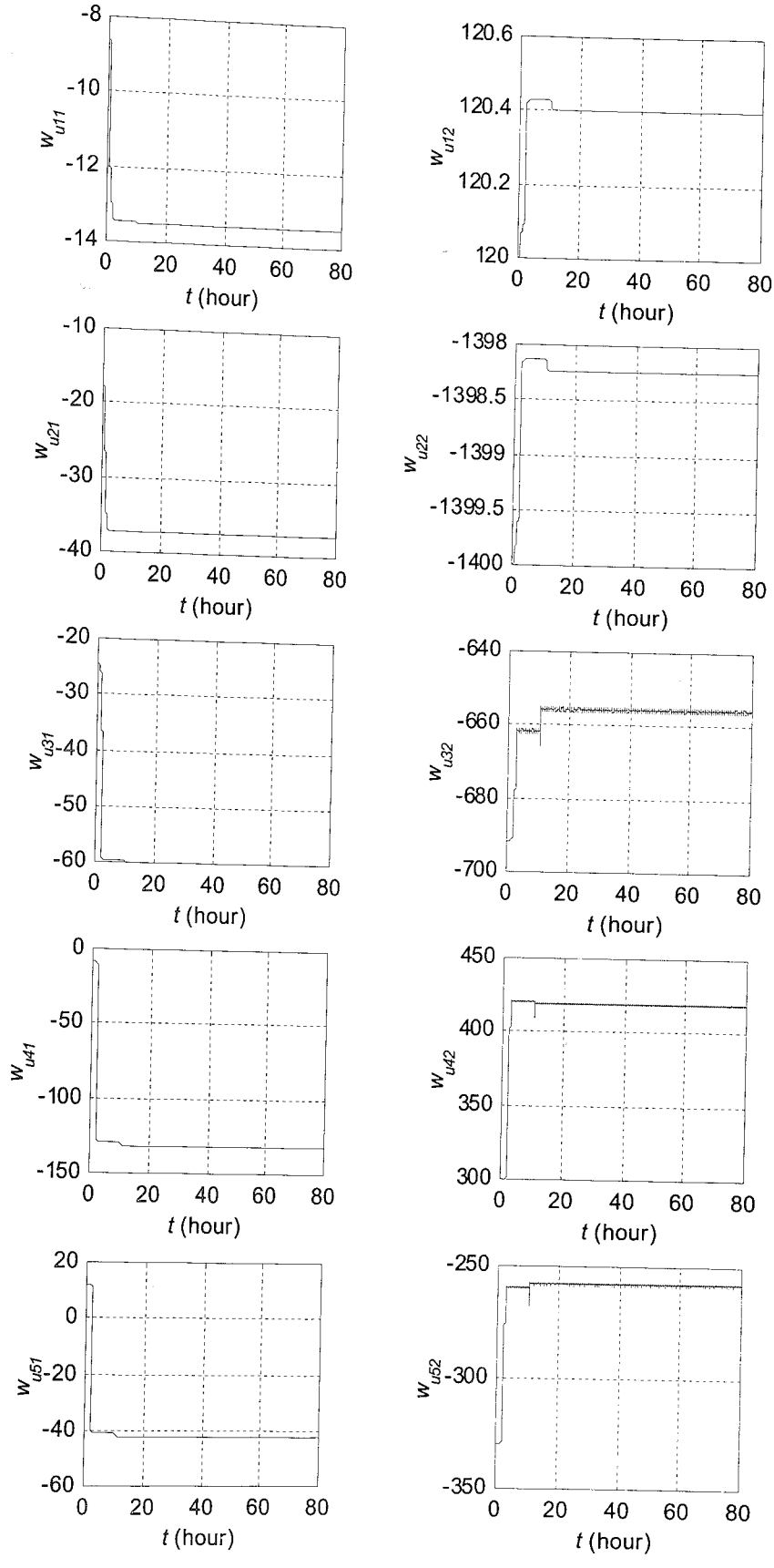

Figure 5.43. Time evolutions of parameters for the first ANFIS based controller of the cement mill process

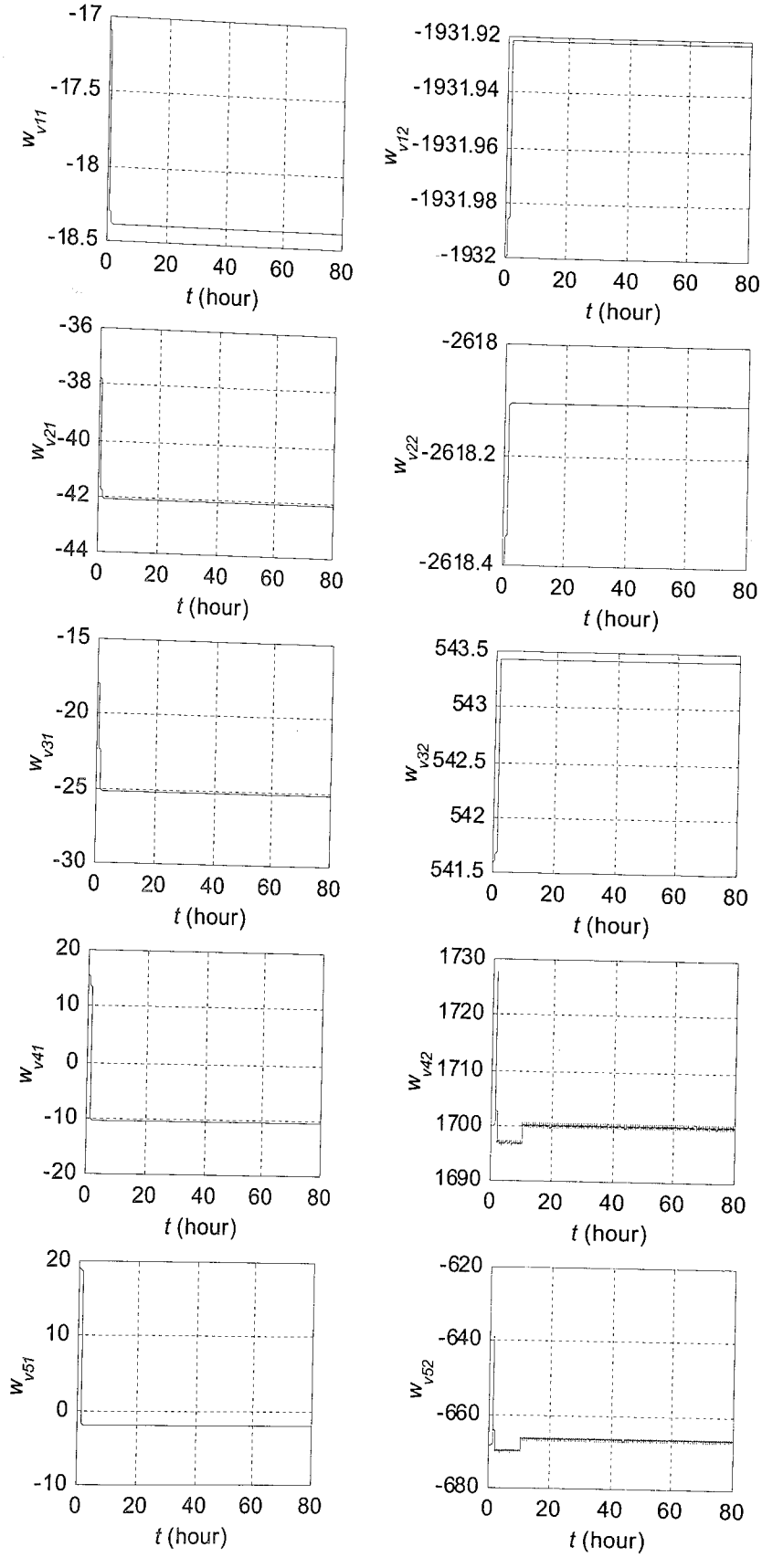

Figure 5.44. Time evolutions of parameters for the second ANFIS based controller of the cement mill process

5.2.6. A Discussion on the Results

Soft computing methodologies with their learning and adaptation abilities constitute a good candidate for control of large scale industrial processes, which are characterized by nonlinearities, modeling uncertainties and time varying parameter. Therefore, intelligent control of a cement mill circuit is investigated in this section. Four different intelligent architectures are utilized for this purpose, and a recently developed training algorithm, which incorporates SMC theory to robustify learning dynamics, is used to train each intelligent structure. Simulation results show that the proposed method is successful for control problem of the cement mill circuit, that is, the system states converge to their desired values in a short period of time and they preserve their values in spite of parametric uncertainties and measurement noise. The latter result shows that this new training algorithm robustifies closed loop dynamics of the system. Moreover, by investigating state errors of four architectures, one can see that there is not a significant difference between the time evolutions of states for different flexible structures. On the other hand, if one compares the implementation effort of all architectures, it can be seen that the time required for realization of the algorithm and to determine the optimal set may differ from one architecture to another. While it is easy to implement and find optimal parameters for ADALINE structure, the complexity of other structures increases as the number of parameters increases. Another issue addressed in this thesis is that the parameters of each intelligent architecture evolve unboundedly if the original learning algorithm is utilized to train the controller. A modification on the original algorithm is proposed in section 4 in order to get rid of this problem. The simulation results given here show that the proposed strategy works as expected, that is, controller parameters remains bounded.

5.3. Lorenz System

The last example investigated in this thesis is the Lorenz system which is discovered by Edward Lorenz while conducting a research on weather patterns. This remarkable system is described by the nonlinear dynamical equations given in (5.14)-(5.16) and shows an interesting behavior for certain values of its parameters, σ, r and b. For example, if one chooses the parameters set as (σ, r, b)=(10, 28, 8/3), which is the most widely investigated parameters set and original used by Lorenz, it can be seen that the state trajectories of the

system neither converge to an equilibrium nor goes to infinity. They simply follow a butterfly like trajectory in a limited volume in the state space. This phenomenon is known as chaos and has been an active area of research for several years.

$$\dot{x} = \sigma(y - x) \tag{5.14}$$

$$\dot{y} = rx - y - xz \tag{5.15}$$

$$\dot{z} = xy - bz \tag{5.16}$$

Chaotic behavior is undesirable in real systems because it has property of sensitivity to initial conditions. In other words, small deviations in initial states lead to completely different solution as time evolves. Moreover, chaotic systems have no stable equilibrium points or periodic orbits. Therefore, control of such systems is a not an easy problem [20]. In the following subsection the control algorithm proposed in section 4 is applied to the Lorenz system as an example of chaos control.

5.3.1. Control of the Lorenz System

Control problem of the Lorenz system is addressed in this subsection. As can be seen from equations (5.14)-(5.16), this system has three state variables and a parameter related with each state equation given above. Therefore, if one utilizes all of the parameters as control inputs there will be three first order subsystems to control. A natural choice for input-output pairs of each subsystem can be given as σ-x, r-y and b-z. Here, the control inputs are applied as parameter perturbations, not necessary to be small, and nominal values of the parameters are chosen as (σ_n, r_n, b_n)=(10, 28, 8/3).

If one carefully investigates the system state equations, it can be seen that a problem similar with the one discussed for cement mill circuit process exists for this system, that is, the control term may become a decreasing function of system inputs. To make this point clear, the dynamic equations given above are rewritten in the following form.

$$\dot{x} = \sigma_n (y - x) + \Delta\sigma (y - x) \tag{5.17}$$

$$\dot{y} = r_n x - y - xz + \Delta r x \tag{5.18}$$

$$\dot{z} = xy - b_n z - \Delta b z \tag{5.19}$$

where $\Delta\sigma$, Δr and Δb are the parametric perturbations. It is easy to see from these equations that if the coefficient of a perturbation input, which is a function of states, is negative, then the term related with this input will be a decreasing function of it. To get rid of this problem, applied perturbation inputs are chosen as follows.

$$\Delta\sigma = \text{sign}(y - x) p_1 \tag{5.20}$$

$$\Delta r = \text{sign}(x) p_2 \tag{5.21}$$

$$\Delta b = -\text{sign}(z) p_3 \tag{5.22}$$

where p_i's are outputs of the controllers associated with each subsystem. By substituting these equations into (5.17)-(5.19), one can easily show that the each control term is an increasing function of controller outputs.

In what follows, performance of the proposed control method is elaborated based on the simulation results. MATLAB 5.3 environment is used for simulations and the step size is chosen as 0.0005 hours. It is required to follow the trajectory, some restrictions are taken into account in this choice of the trajectory, expressed below while the initial states of the system are set as (x_0, y_0, z_0)=(-20, -20, 0).

$$x(t) = 10 \sin(\frac{2\pi}{1.2} t) + 13 \tag{5.23}$$

$$y(t) = 10 \sin(\frac{2\pi}{1.2} t) + 20 \tag{5.24}$$

$$z(t) = 10\cos(\frac{2\pi}{1.2}t) + 20 \tag{5.25}$$

The constraint that is imposed on the reference input is that control terms should not vanish while system states on the trajectory. In other words, the coefficients of perturbations in (5.17)-(5.19) should not be zero in order to not loose the control. As in the previously investigated systems, to test the robustness of the proposed mechanism, state measurements are subjected to Gaussian noise with zero mean and variance 1.64e-6. Moreover, instead of the discontinuous switching function utilized in the original adaptation algorithm, its smooth approximation given in (5.7) is used to eliminate the occurrence of high frequency input signals while the system in sliding regime. The uncertainty bounds for controllers corresponding subsystems having σ-x, r-y and b-z pairs as input-outputs are determined as K_σ=200, K_r=700 and K_b=700 respectively. Furthermore, each controller changes its operating mode to prevent unbounded evolution of parameters when the absolute value of error is less than 0.2.

5.3.2. Application with ADALINE Based Controller

Simulation results for ADALINE based controller are investigated in this subsection. Three-dimensional appearances of the reference trajectory and the actual trajectory of the system are shown in Figure 5.45 while time evolution of state errors are given in Figure 5.46. As these figures suggest, states of the system converges to the desired trajectory in a short period of time and follow it thereafter in spite of presence of measurement noise. This result shows that the ADALINE based controller successful in tracking control of the Lorenz system and robust against uncertainties. The transient phase of the control inputs produced by ADALINE architecture are given in Figure 5.47. As can be seen from the figure, control signals can take large values initially when compared with their long run behavior given in Figure 5.48. These long run graphs will not be given in the following subsections again because they are almost same for each control architecture. Lastly, time evolution of controller parameters are given in Figure 4.49, Figure 4.50 and Figure 4.51. As shown in these figures, parameters corresponding to bias terms make bounded oscillations while remaining ones converges to a limiting value. This observation shows

that the proposed method prevents the parameter drift problem which is inevitable in the original training algorithm.

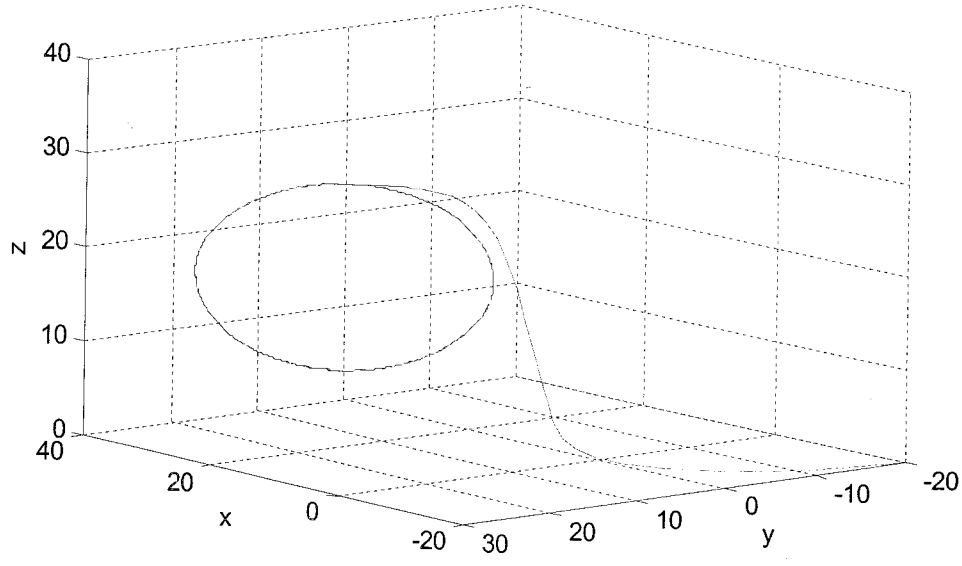

Figure 5.45. State and reference trajectories of the Lorenz system for ADALINE based controller

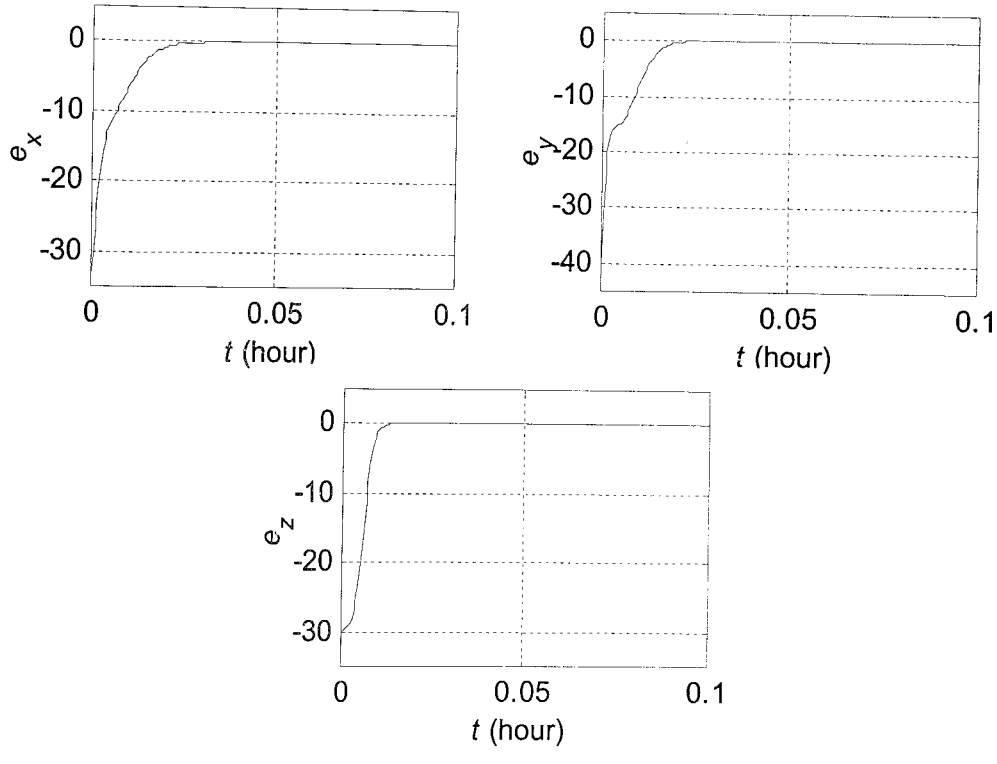

Figure 5.46. State tracking errors of the Lorenz system for ADALINE based controller

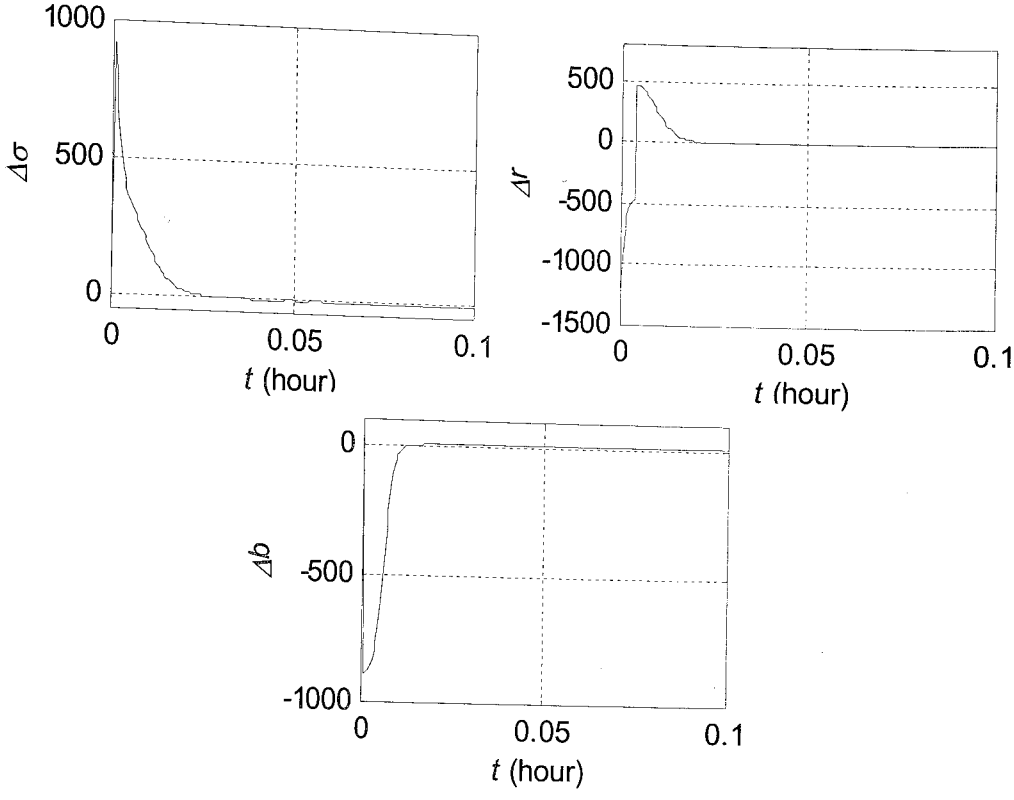

Figure 5.47. Transient phase of the control signals produced by ADALINE based controller to control the Lorenz system

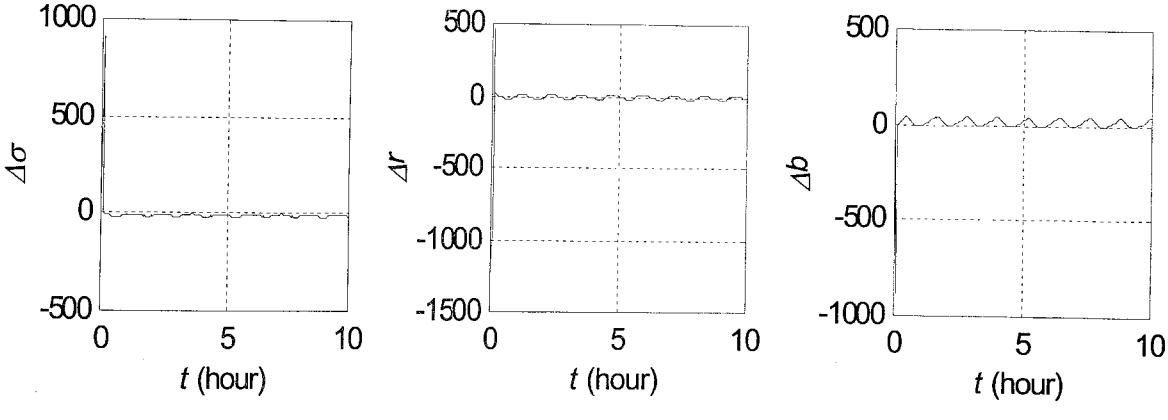

Figure 5.48. Control signal produced by ADALINE based controller in the long run

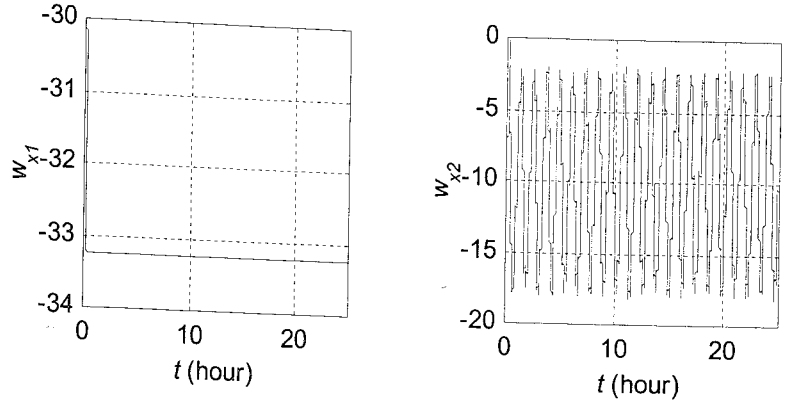

Figure 5.49. Time evolutions of parameters for the first ADALINE based controller of the Lorenz system

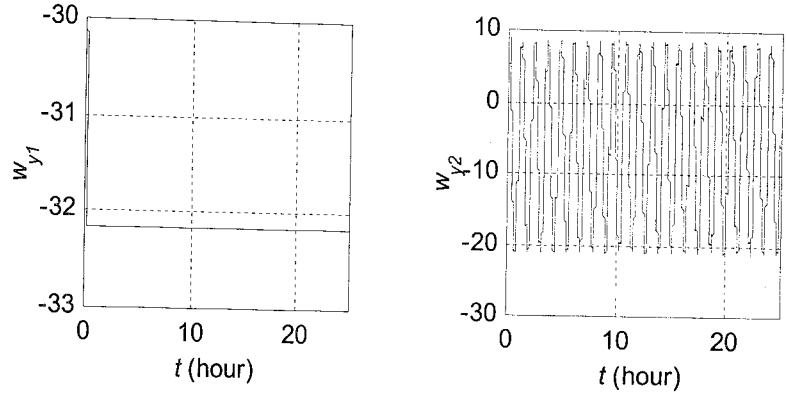

Figure 5.50. Time evolutions of parameters for the second ADALINE based controller of the Lorenz system

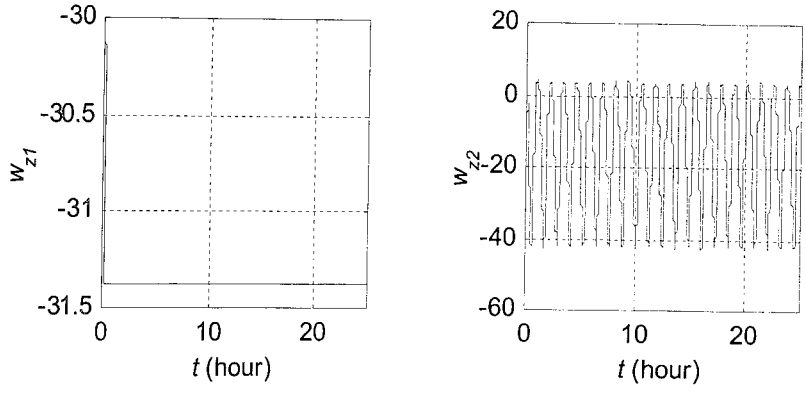

Figure 5.51. Time evolutions of parameters for the third ADALINE based controller of the Lorenz system

5.3.3. Application with GRBFNN Based Controller

The second architecture utilized for tracking control of the Lorenz system is Gaussian Radial Bases Function Neural Network. The state trajectory for this controller is shown in Figure 5.52. As depicted in the figure, starting from (x_0, y_0, z_0)=(-20, -20, 0), states of the system converges to the desired circular trajectory under the control law given in (4.19)-(4.22). By analyzing the time evolutions of state errors given in Figure 5.53, one can say that it takes approximately 018, 0.017 and 0.012 hours respectively for x, y, z components to converge to the reference trajectory. These values shorter than the required times for ADALINE based controller. On the other hand, if computational complexity is of primary concern, one should prefer ADALINE structure. Each GRBFNN utilized in this subsection can be described in terms of its activations functions and weights. While the former is shown in Figure 5.55, time evolutions of the weights are given in Figure 5.56-Figure 5.58. As these figures suggest, weights of the network remain bounded.

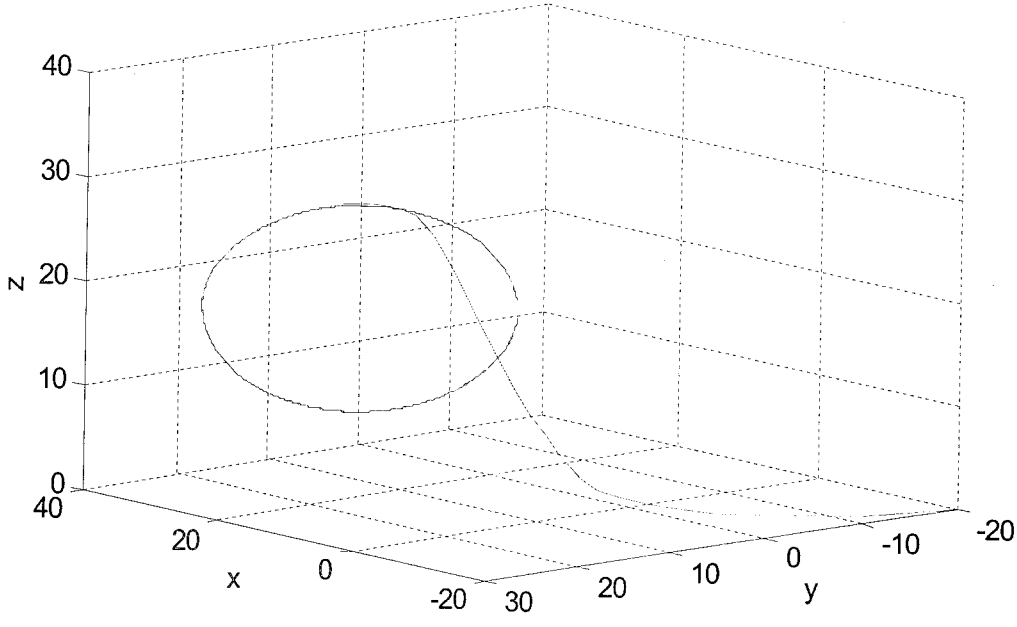

Figure 5.52. State and reference trajectories of the Lorenz system for GRBFNN controller

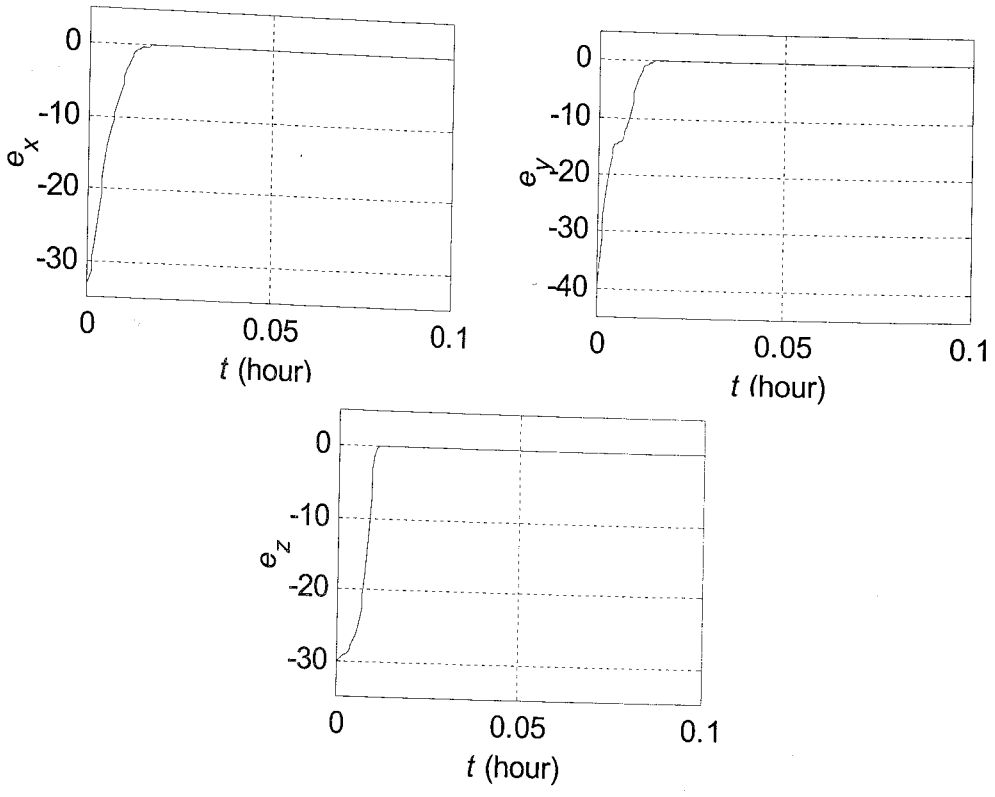

Figure 5.53. Stare errors of the Lorenz system for GRBFNN based controller

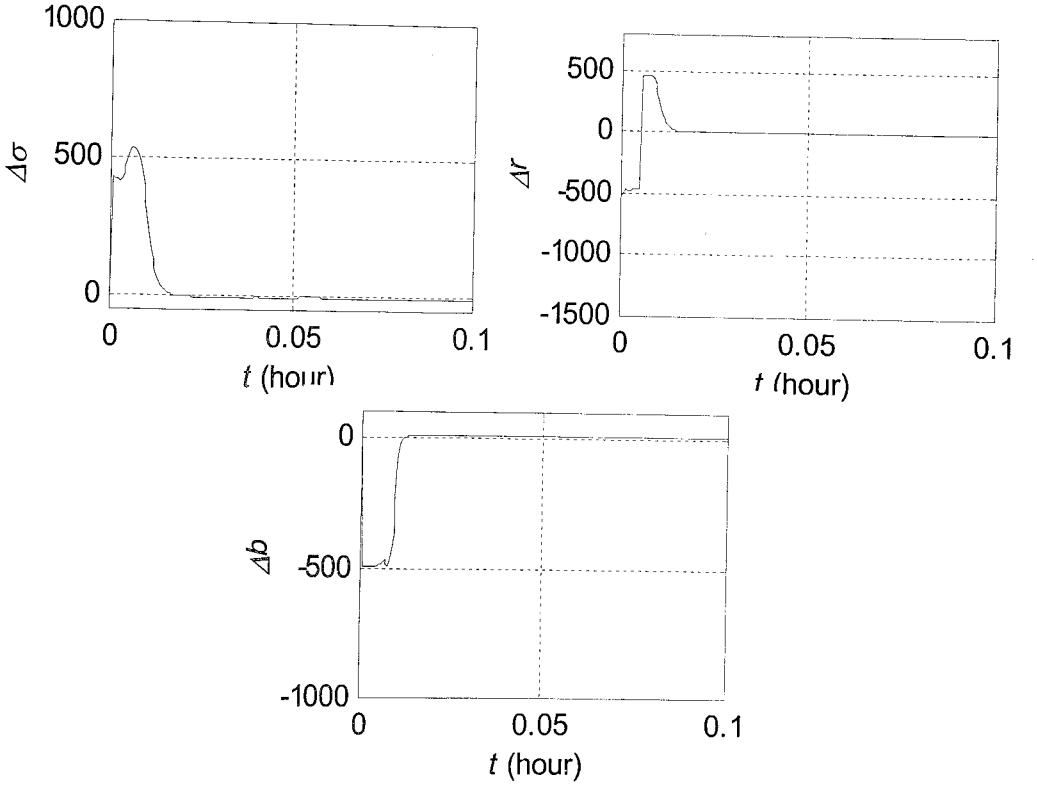

Figure 5.54. Transient phase of the control signals produced by GRBFNN based controller to control the Lorenz system

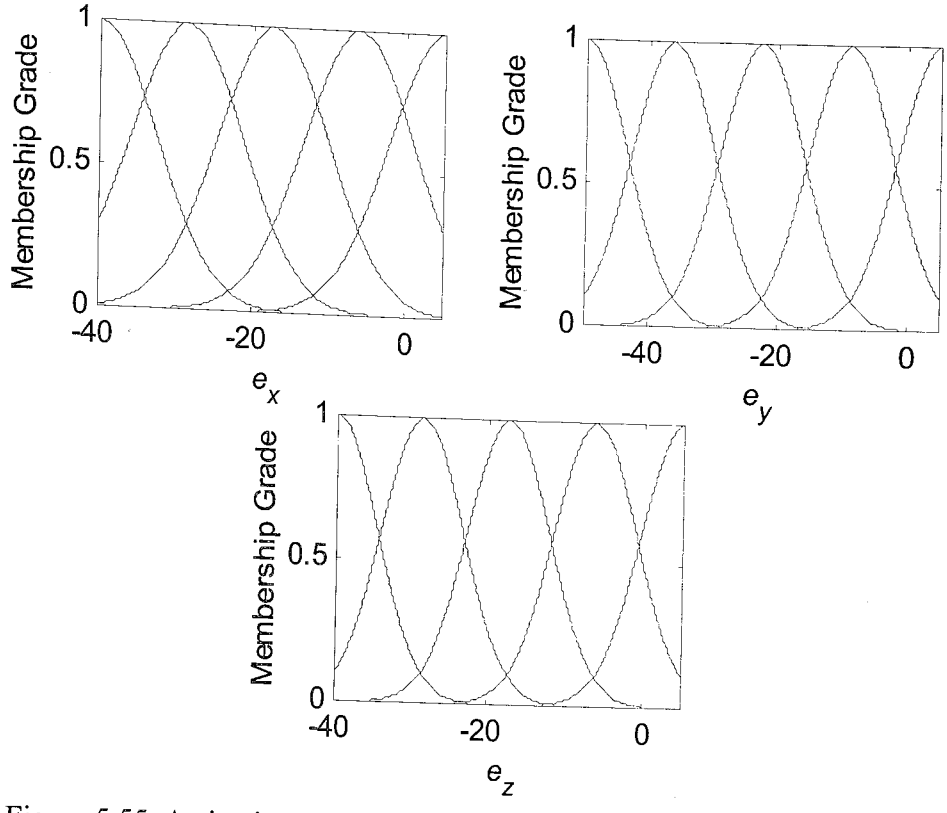

Figure 5.55. Activation functions of GRBFNN controllers for the Lorenz system

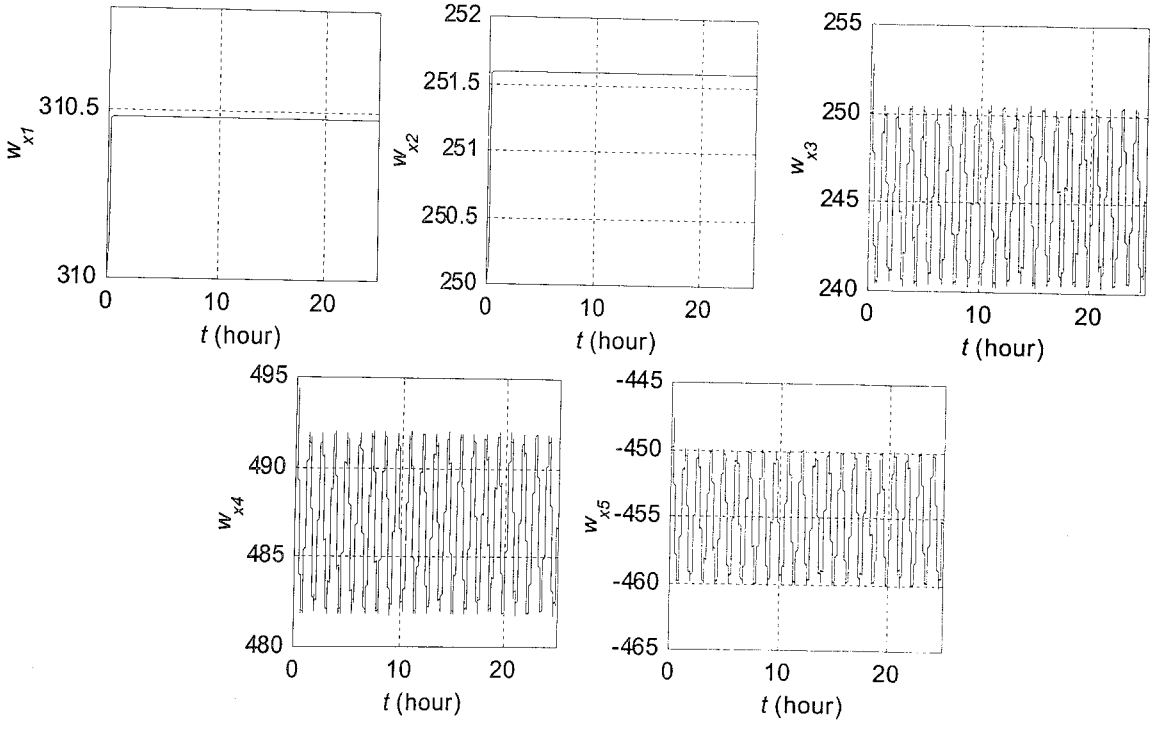

Figure 5.56. Time evolutions of parameters for the first GRBFNN based controller of the Lorenz system

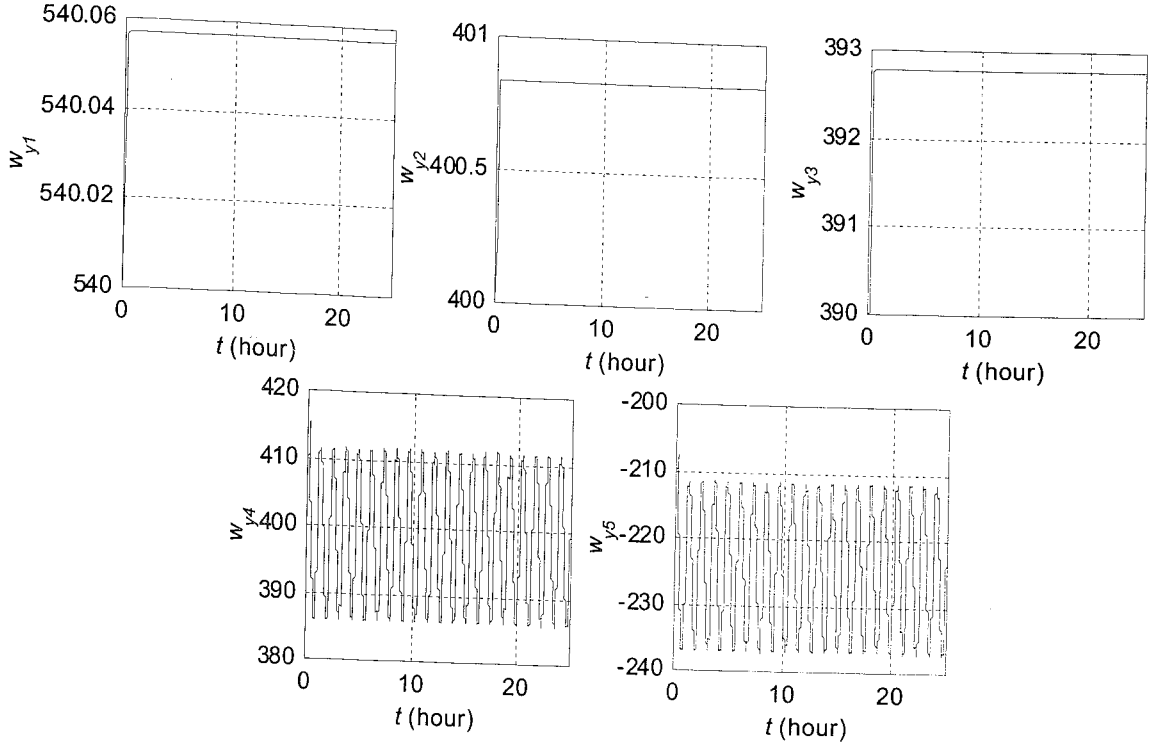

Figure 5.57. Time evolutions of parameters for the second GRBFNN based controller of the Lorenz system

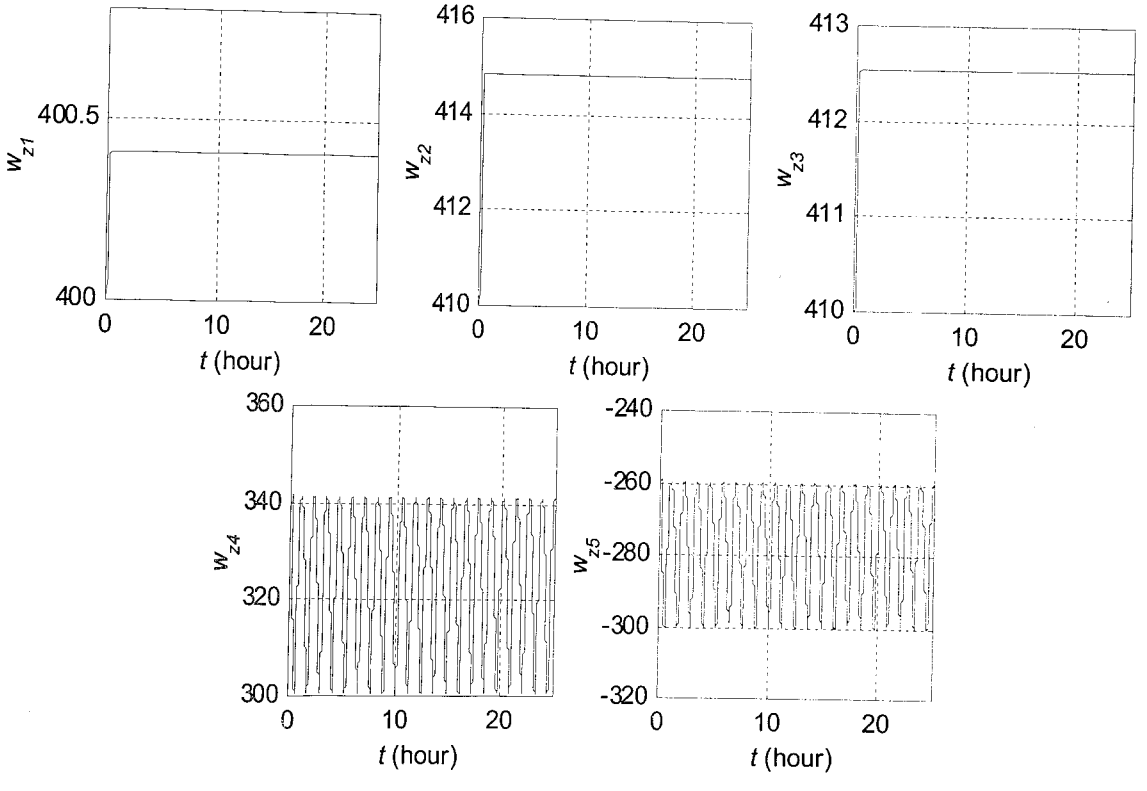

Figure 5.58. Time evolutions of parameters for the third GRBFNN based controller of the Lorenz system

5.3.4. Application with SFS Based Controller

Performance of SFS based controller is evaluated in this subsection. While the state trajectory of the system is shown in Figure 5.59, each component of the state vector is given in the Figure 5.60 as a function time. From these figure one can see that, although it is easy to realize that the state trajectories for SFS based controller differs from GRBFNN based controller in transient phase, the settling times of state components are approximately same for both architectures. Moreover, one can say that the peak values for control signals shown in Figure 5.61 are small when compared with ADALINE based controller. The rule bases of SFS architectures utilizes the bell shaped membership functions given in Figure 5.62 in order to partition error spaces. Moreover, the rule consequences vary with time as shown in Figure 5.63, Figure 5.64 and 5.65 under the adaptation law given in (4.13). As depicted in these figures parametric variances do not lead to any drift.

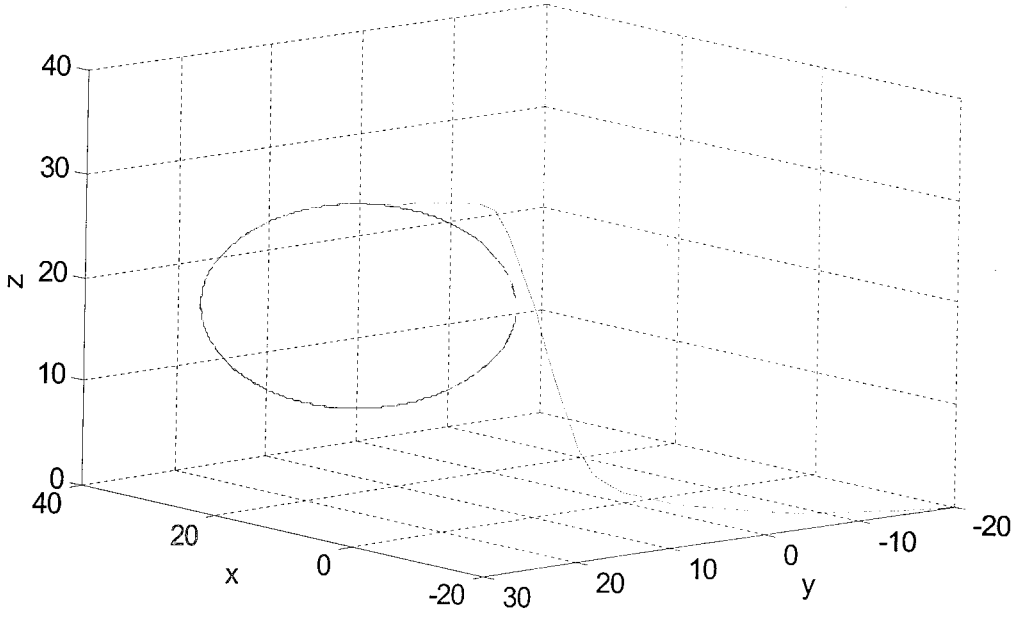

Figure 5.59. State and reference trajectories of the Lorenz system for SFS based controller

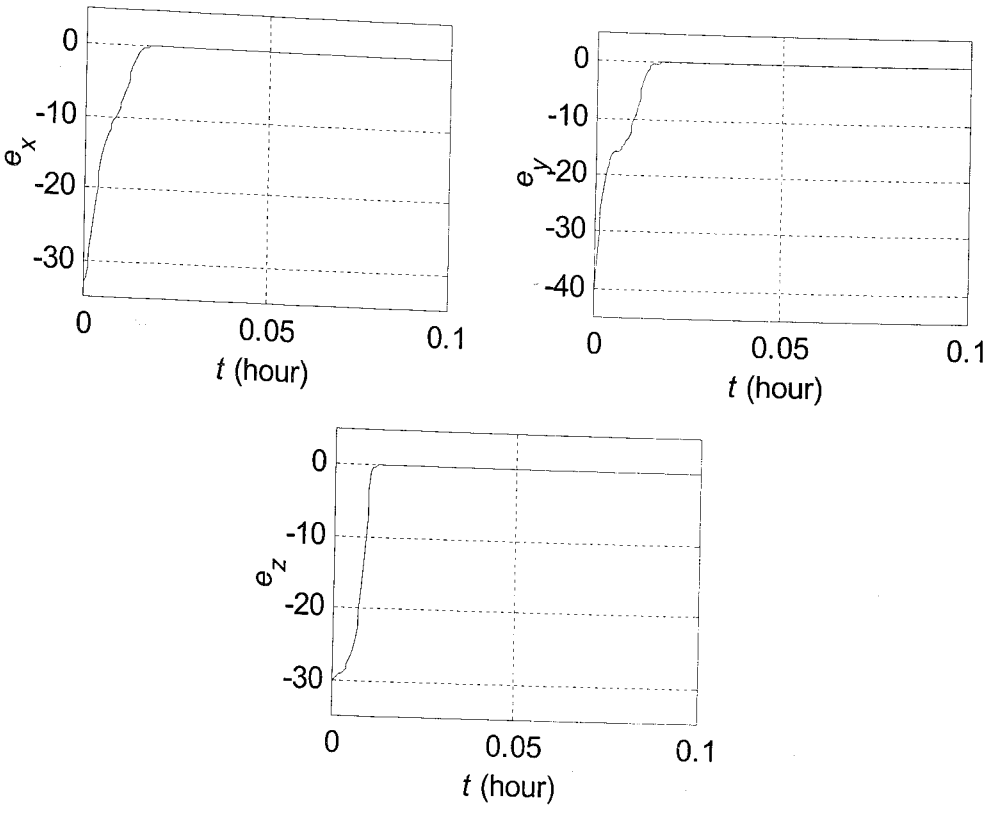

Figure 5.60. Stare errors of the Lorenz system for SFS based controller

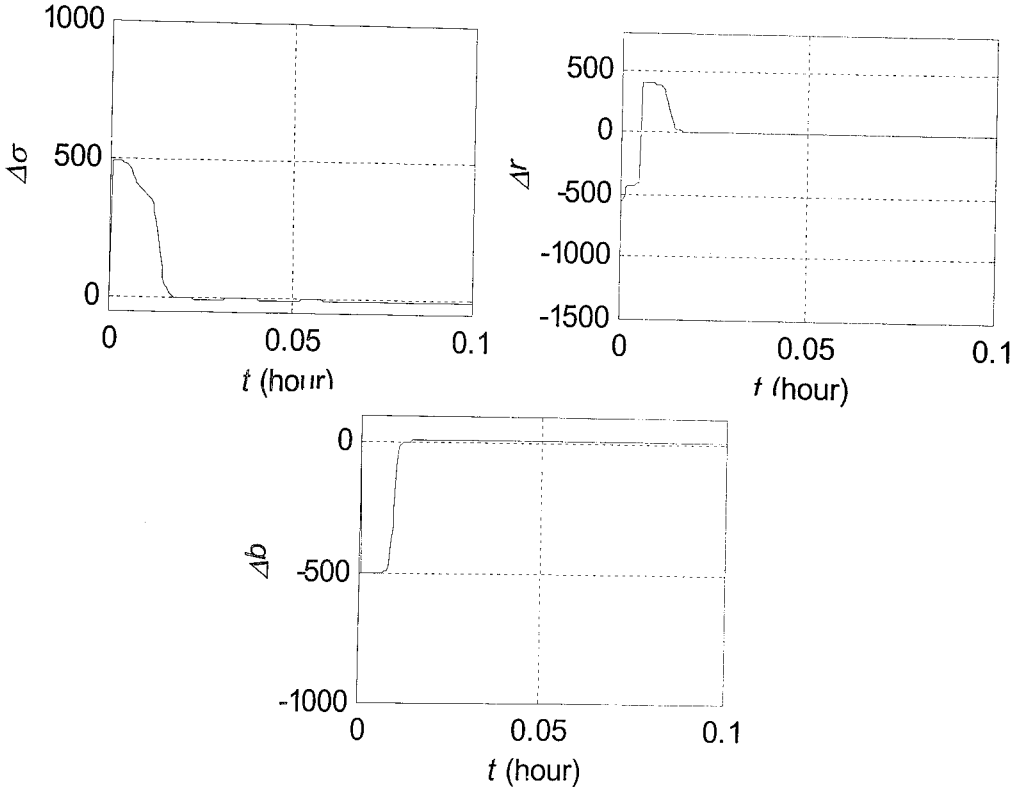

Figure 5.61. Transient phase of the control signals produced by SFS based controller to control the Lorenz system

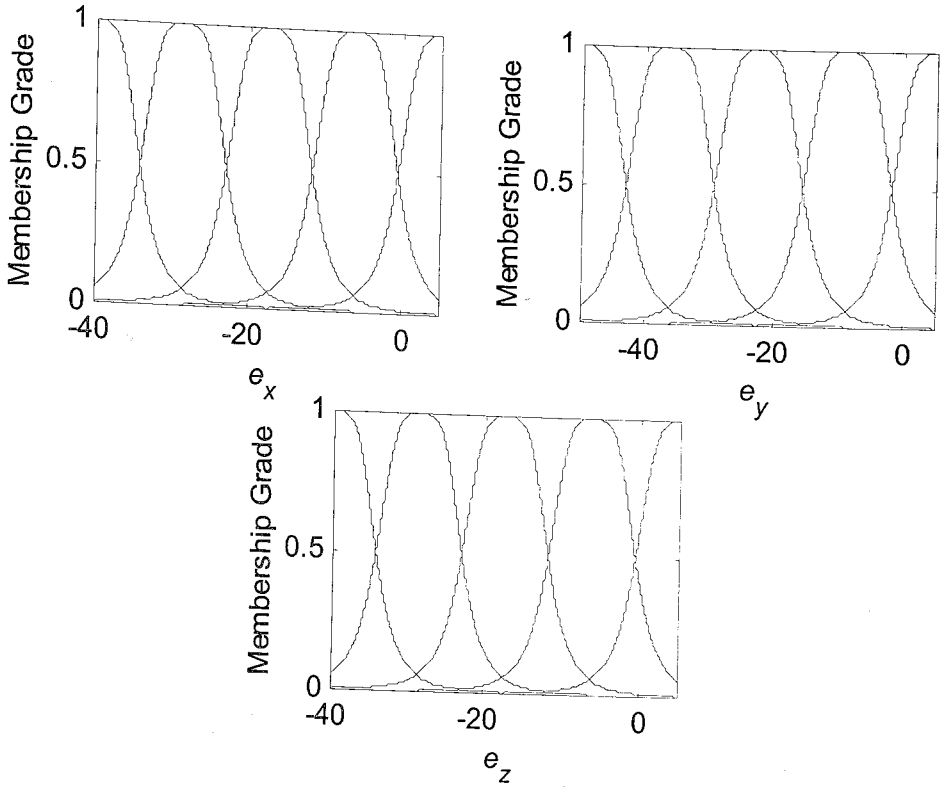

Figure 5.62. Membership functions of SFS architectures for the Lorenz system

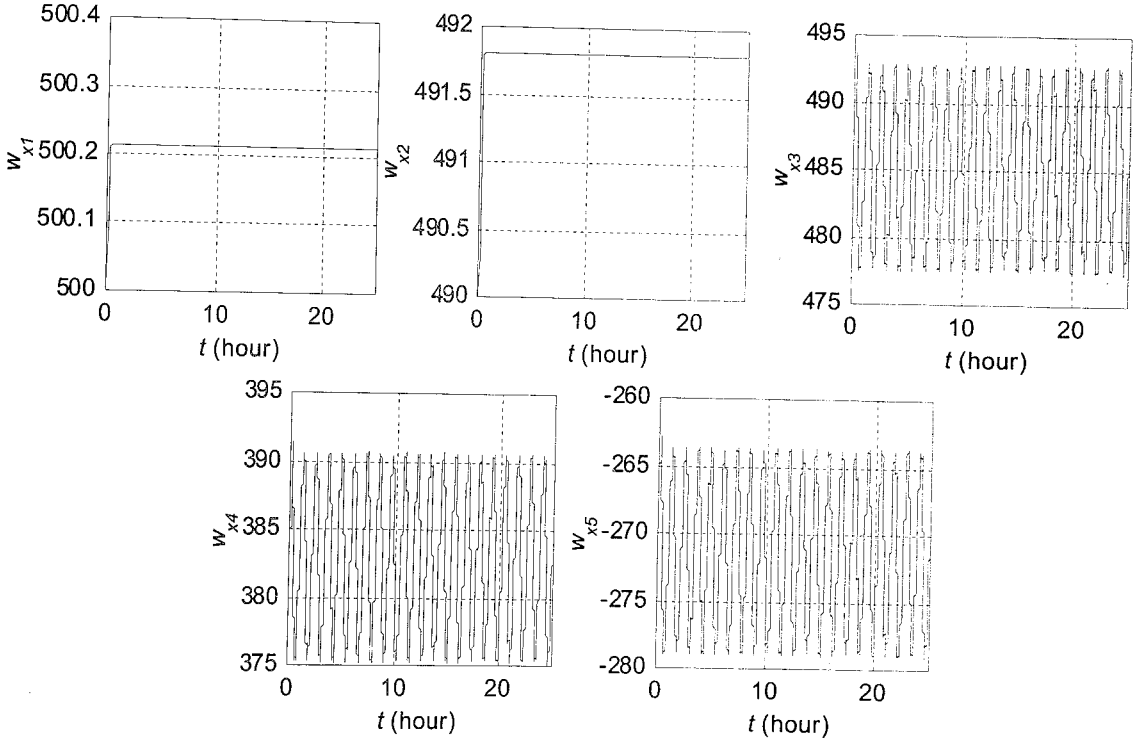

Figure 5.63. Time evolutions of parameters for the first SFS based controller of the Lorenz system

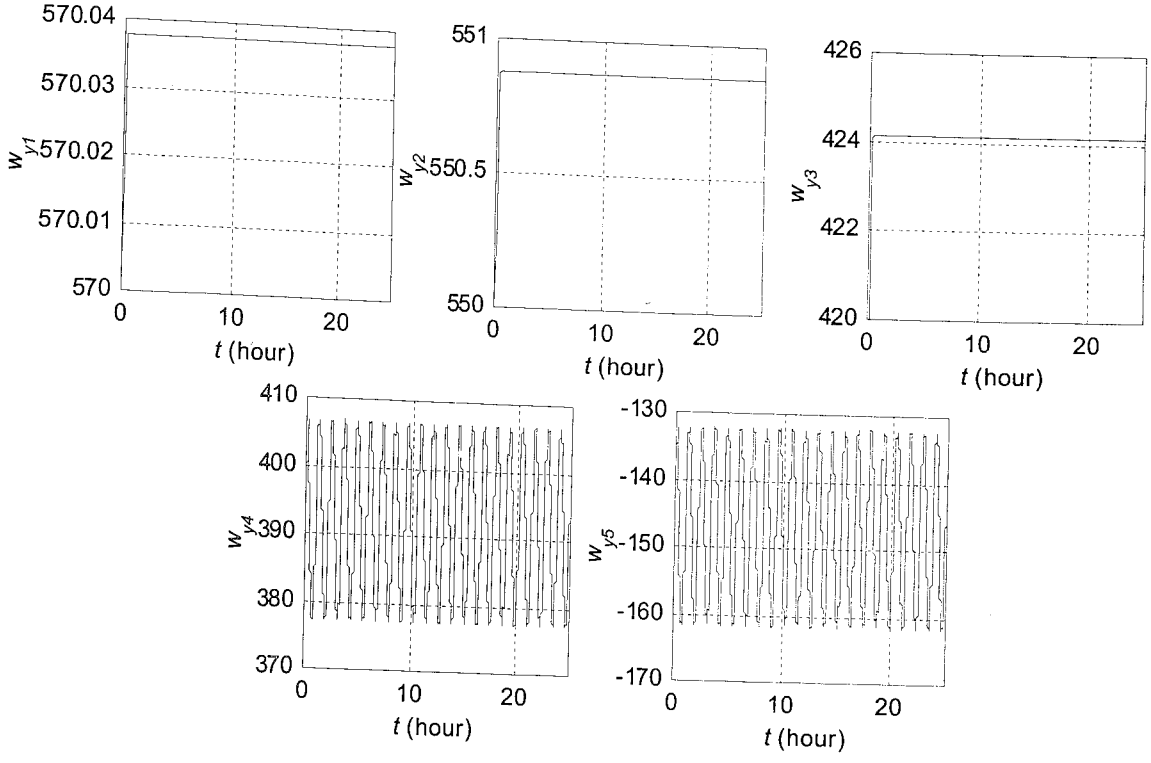

Figure 5.64. Time evolutions of parameters for the second SFS based controller of the Lorenz system

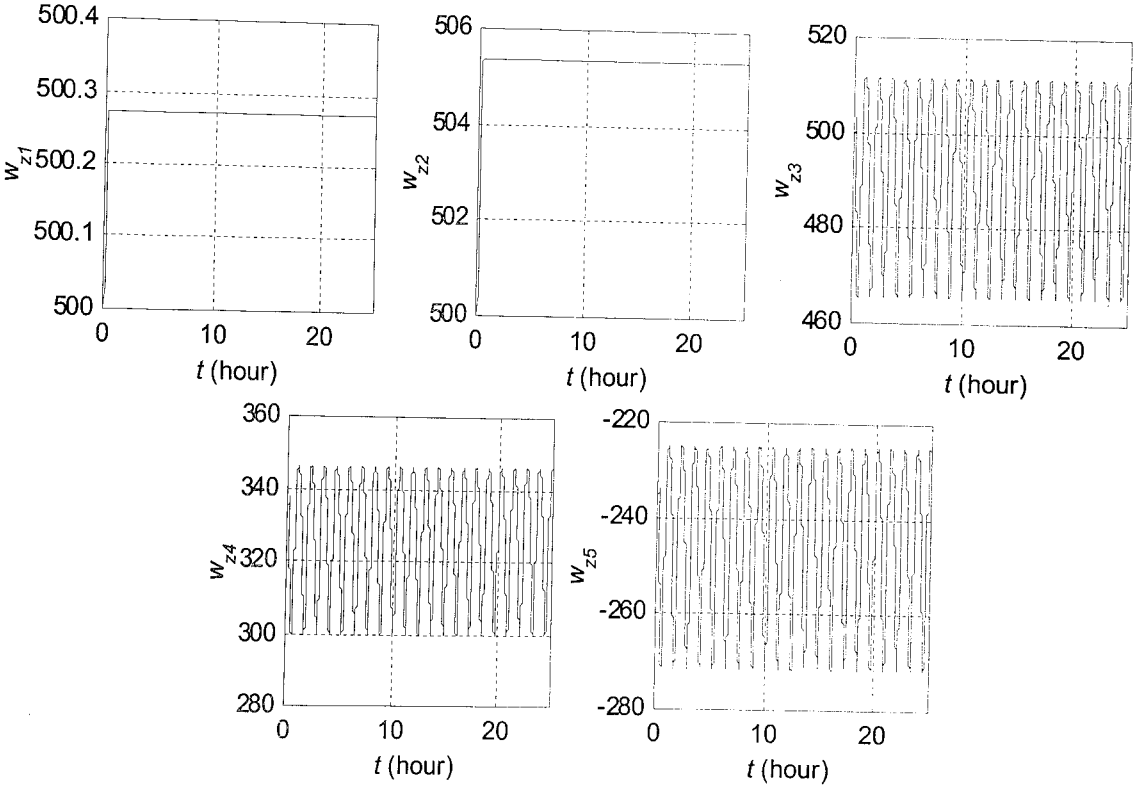

Figure 5.65. Time evolution of parameters for the third SFS based controller of the Lorenz system

5.3.5. Application with ANFIS Based Controller

The behavior of the Lorenz system for ANFIS based controller can be described by simulation results given below. Sates of the system follow the trajectory shown in Figure 5.66 under the control of ANFIS structure, parameters of which are updated by the adaptation algorithm introduced in section 4. Moreover, if one draws the components of this trajectory as a function of time, the graphs shown in 5.66 can be obtained. A comparison of these results with the ones given for SFS based controller reveals the similarity between the performances of both architectures. This similarity can be accounted to the fact that the control signals produced by SFS, Figure 5.61, and ANFIS, Figure 5.68, controllers are almost same. Each ANFIS structure utilized in this subsection uses one of the corresponding membership function sets given in Figure 5.69 in order to calculate firing strength of each rule. Then, these firing strengths are used in conjunction with the adjustable parameters, which are updated with the adaptation rule given in (4.14) and time evolutions of which are given in Figure 5.70-Figure 5.72, of the structure to obtain its overall output. As can be seen from the figures, each parameter remains bounded as time evolves.

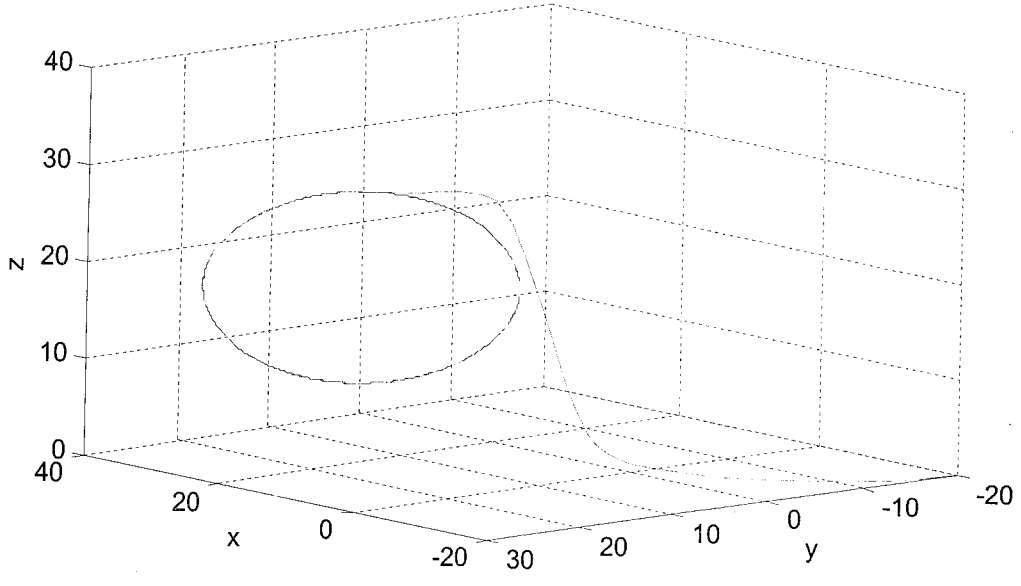

Figure 5.66. State and reference trajectories of the Lorenz system for ANFIS based controller

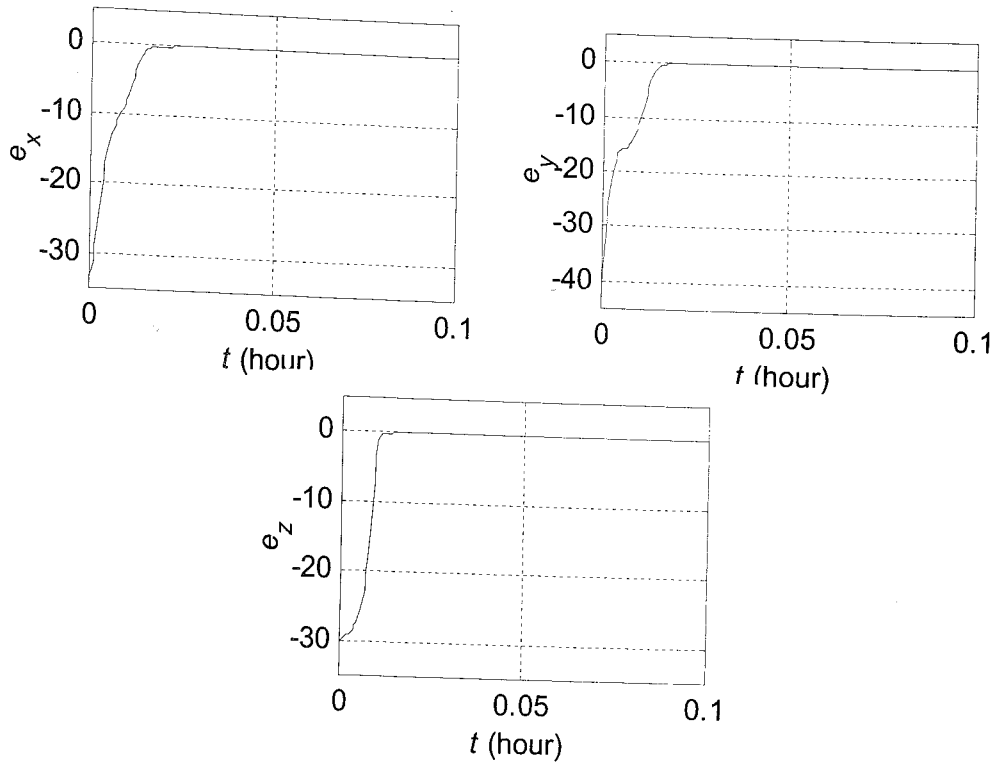

Figure 5.67. Stare errors of the Lorenz system for ANFIS based controller

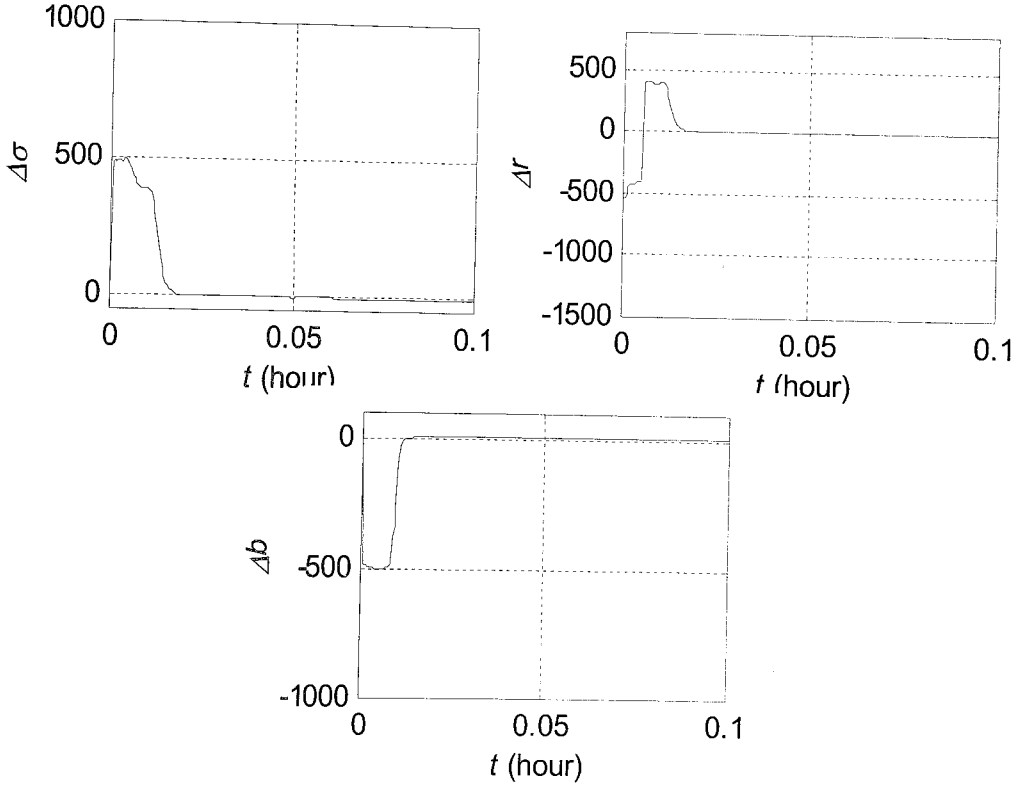

Figure 5.68. Transient phase of the control signals produced by ANFIS based controller to control the Lorenz system

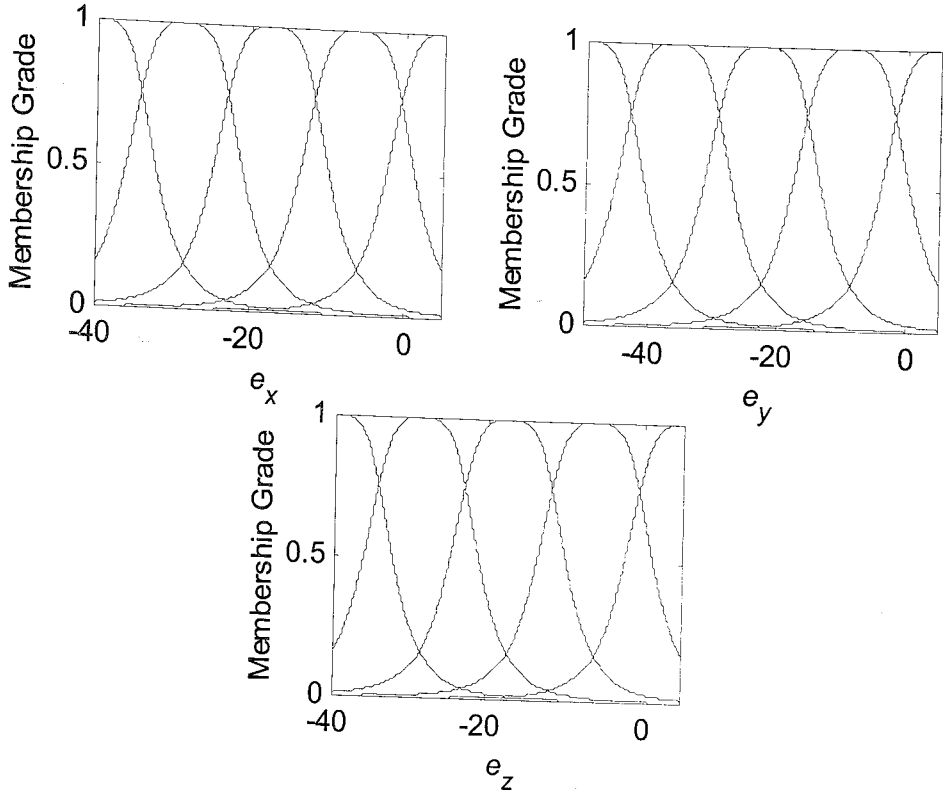

Figure 5.69. Membership functions of ANFIS architectures for the Lorenz system

5.3.6. A Discussion on the Results

The aim of this subsection is to adopt the proposed control strategy to the Lorenz system. In order to achieve this objective, the Lorenz system is divided into three subsystems, and each subsystem controller by a separate controller. The simulation results show that the closed loop dynamics is stable for all architectures, that is, the system states can follow the reference trajectory. Moreover, if one takes into account the fact that there are strong couplings between the states, which act as disturbances for each subsystem, and the state measurements are subject to noise, it can be said that all control architectures are similar in the sense that each of them shows a robust characteristic under the corresponding parameter adaptation law given in section 4. On the other hand, there are also some differences between these intelligent architectures, which can be exploited to choose the most suitable controller for the given problem. If one analyzes the applied control signal for different architectures, it can be seen that control effort required by ADALINE based controller is considerably high when compared with the others. Moreover, in terms of settling time metric GRBFNN based controller performs a little bit better than the other architectures. Consequently, if all these facts are taken into account, it can be said that the

most suitable architecture for the control of the Lorenz system is GRBFNN. However, one should also keep in mind that the performances of SFS and ANFIS structures are very close that of GRBFNN based controller, and hence, they constitute a good alternative. The last point to mention in this subsection is unbounded parameter evolution problem. As discussed before the original parameter update rules derived in section 4 leads to persistent increments in the absolute values of adjustable parameters. Because this situation causes to problems in applications, a remedy is proposed in this study to keep the parameters of each architecture bounded. As in the case of the previously discussed systems, the simulation results for the Lorenz system shows that the proposed method woks properly.

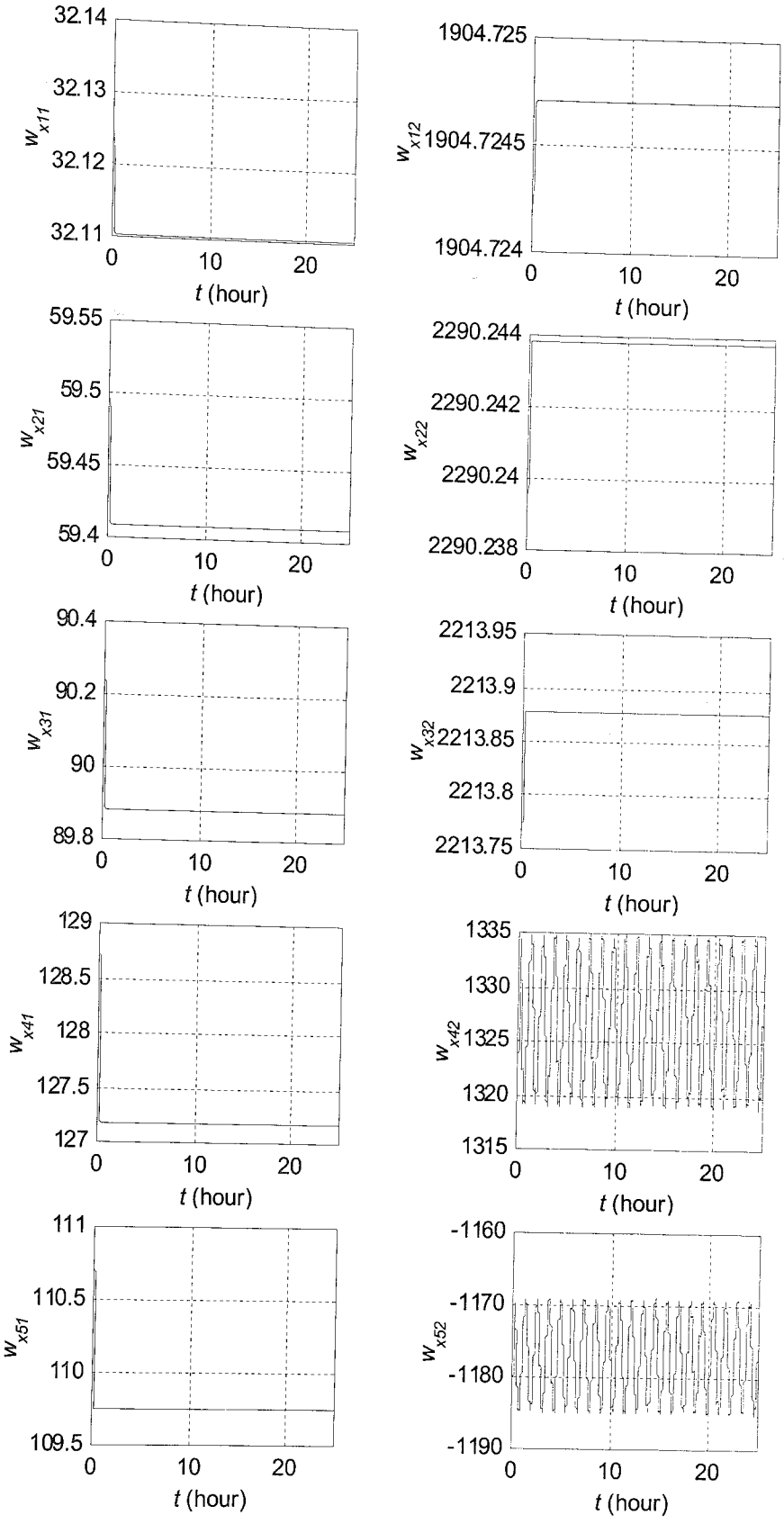

Figure 5.70. Time evolutions of parameters for the first ANFIS based controller of the Lorenz system

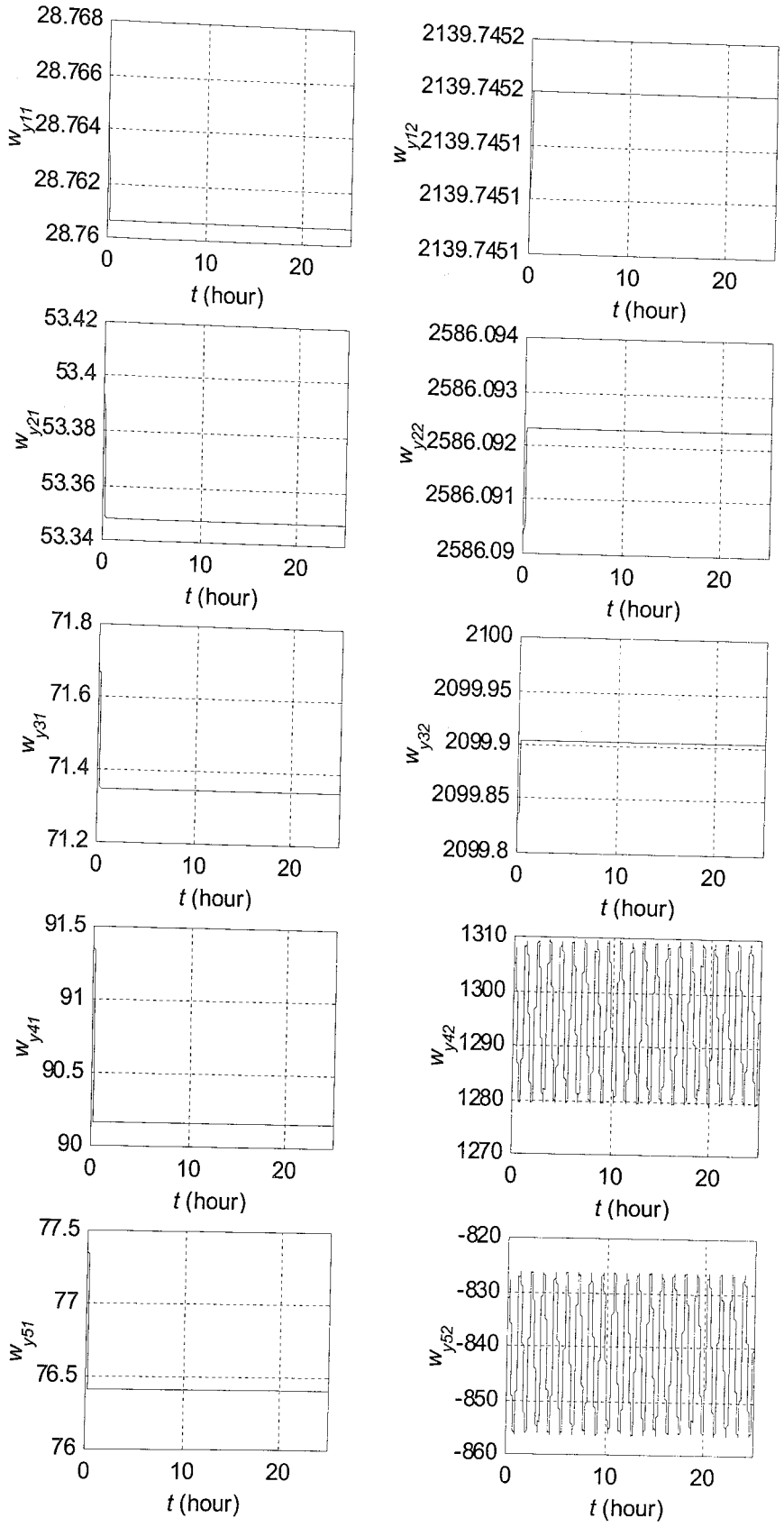

Figure 5.71. Time evolutions of parameters for the second ANFIS based controller of the Lorenz system

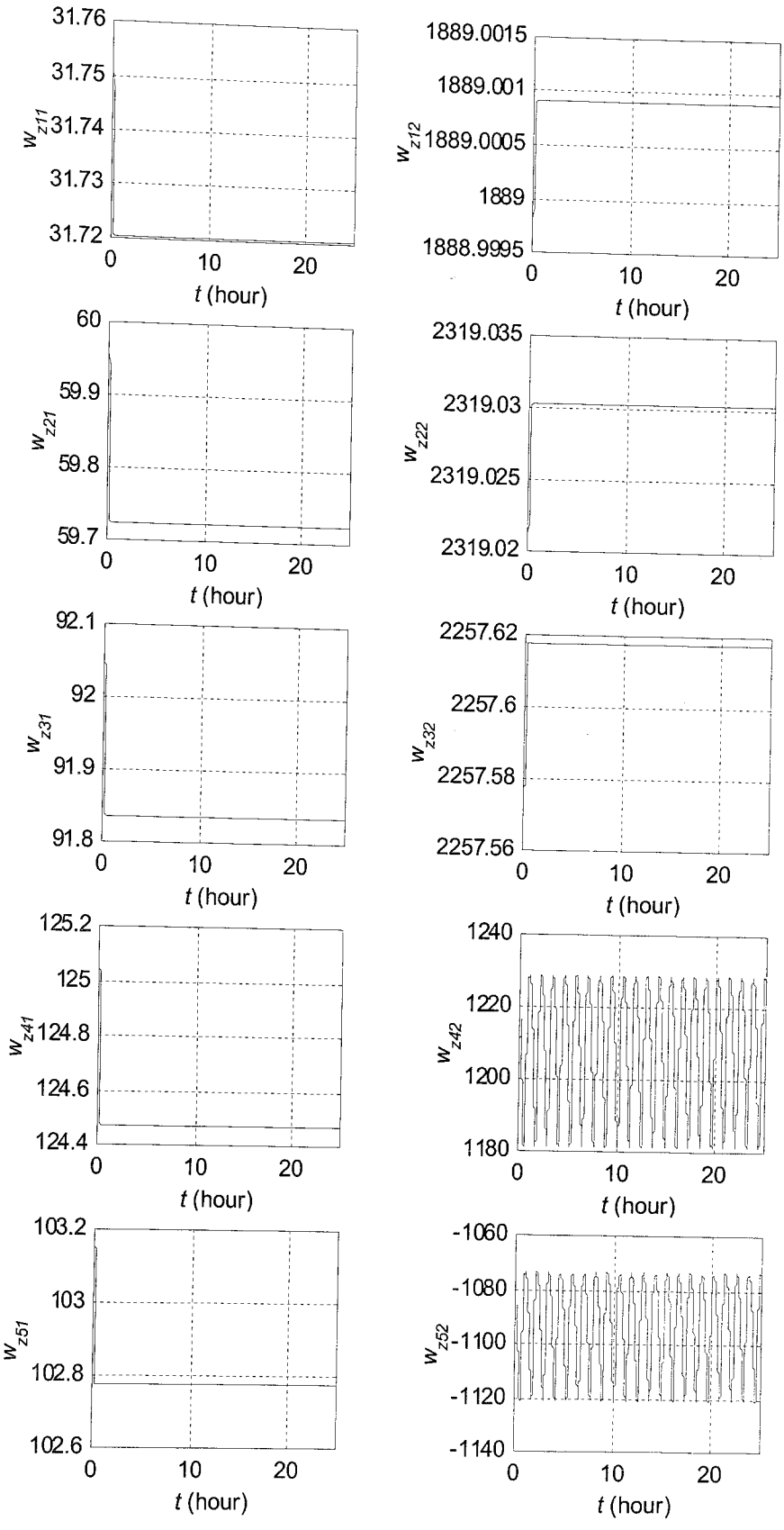

Figure 5.72. Time evolutions of parameters for the third ANFIS based controller of the Lorenz system

6. CONCLUSIONS

The area of intelligent control has emerged in the early second half of the 20th century to fulfill the requirements of low cost solutions to complex engineering systems, ability to deal with uncertainties and providing an interface between the human expertise and numerical computation methods. In this regard, several methodologies have been proposed and found to be successful in a variety of automatic control applications, including robotics and industrial process systems. The difference between these methodologies lies in the intelligent architectures utilized in the control loop and training algorithm used to update adjustable parameters of the flexible structures.

In this thesis, a recently developed parameter adaptation algorithm is utilized to train four different intelligent control architectures, namely Adaptive Linear Element, Gaussian Radial Bases Function Neural Network, Standard Fuzzy System and Adaptive Neuro-fuzzy Inference System. For this purpose, the update rule given in [10], which is originally formulated for second order systems, is modified in order to make it applicable to first order nonlinear systems in canonical form. Moreover, throughout the thesis, it has been shown that there is a strong relation between behaviors of different architectures under the proposed adaptation rule, that is, input-output curves of all architectures exhibit a similar characteristic as time evolves. One of the inevitable, and undesirable, outcome of this similarity is that parameters of all architectures evolves unboundedly if the update rule derived in section 4 is utilized directly. In order to alleviate this difficulty, adaptation algorithm of each architecture is modified while the system states in the vicinity of the desired states. The simulation results for different system prove that the modified update rule woks as expected.

Throughout the thesis, simulation results for four different architectures are investigated in order to evaluate their performances. However, it has been observed that in terms of applied control inputs and time evolution of system states, there is not a considerable difference between performances of those architectures. Therefore, one may not make a decision between different architectures based on these metrics. On the other hand, computational complexity and effort required for each architecture to determine the

optimal parameter set differs from one to another. As can be seen from Table 6.1 the complexity and required effort increases as the number of adjustable parameters increase.

Based on the above given observations, it can be concluded that even if the time scale of the system in hand is large, one should prefer simple structures like ADALINE network in order to reduce the time required for initial parameter adjustment and implementation of control algorithm.

Table 6.1. Complexity ranking of different architectures

	Amount of time required to implement the algorithm	Amount of time required to adjust parameters
ADALINE	1	1
GRBFNN	2	2
SFS	2	2
ANFIS	3	3

* In this table 1 stands for the minimum time while 3 for the maximum

REFERENCES

1. Jang, J.-S. R., C.-T. Sun and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, PTR Prentice Hall, 1997.

2. Kaynak, O., K. Erbatur and M. Erbatur, "The Fusion of Computationally Intelligent Methodologies and Sliding-Mode Control-A Survey", *IEEE Transactions on Industrial Electronics*, Vol. 48, No. 1, pp.4-17, February 2001.

3. Norgaard, M., O. Ravn, N.K. Poulsen and L.K. Hansen, *Neural Networks for Modelling and Control of Dynamic Systems,* Springer, 2000.

4. Sanner, R. M. and J.-J. E. Slotine, "Gaussian Networks for Direct Adaptive Control", *IEEE Transactions on Neural Networks*, Vol. 3, No. 6, pp. 837-863, November 1992.

5. Efe, M. O. and O. Kaynak, "A Comparative Study of Soft Computing Methodologies in Identification of Robotic Manipulators", *Robotics and Autonomous Systems,* Vol. 30, No. 3, pp. 221-230, 2000.

6. Narendra, K. S. and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, pp. 4-27, 1990.

7. Sira-Ramirez, H. and E. Colina-Morles, "A Sliding Mode Strategy for Adaptive Learning in Adalines", *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, Vol. 42, No.12, pp. 1001-1012, December 1995.

8. Yu, X., M. Zhihong and S. M. M. Rahman, "Adaptive Sliding Mode Approach for Learning in a Feedforward Neural Network", *Neural Computing & Applications*, Vol. 7, pp. 289-294, 1998.

9. Parma, G. G., B. R. Menezes and A. P. Braga, "Sliding Mode Algorithm for Training Multilayer Artificial Neural Networks", *Electronics Letters*, Vol. 34, No. 1, pp. 97-98, January 1998.

10. Efe, M. O., *Variable Structure Systems Theory Based Training Strategies for Computationally Intelligent Systems*, Ph.D. Thesis, Boğaziçi University, 2000

11. Zadeh, L. A., "Roles of Soft Computing and Fuzzy Logic in the Conception, Design and Development of Information/Intelligent Systems", in O. Kaynak, L. A. Zadeh, B. Turksen and I. J. Rudas, *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications*, pp. 1-9, NATO ASI Series, Vol. 162, Springer, 1996.

12. Fukuda, T. and K. Shimojima, "Intelligent Robotic Systems Based on Soft Computing-Adaptation, Learning and Evolution", in *Computational Intelligence –Soft Computing and Fuzzy –Neuro Integration with Applications*, pp. 458-489, Springer Verlag, Berlin, Germany, 1998.

13. McCulloch, W.S. and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133, 1943.

14. Bishop, C. M., *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

15. Slotine, J.-J. E. and W. Li, *Applied Nonlinear Control*, Prentice-Hall, New Jersey, 1991.

16. Ertuğrul, M., *Neuro-Sliding Mode Control of Robotic Manipulators*, Ph.D. Thesis, Boğaziçi University, 1999.

17. Boskovic, J. D., "Novel Feeding Strategy for Adaptive Control of Bioreactor Processes," *J. Proc. Cont.*, Vol.7, pp.209-217, 1997.

18. GE Industrial systems, "Cement mill", http://www.geindustrial.com/industrialsystems/solutions/cement-2.html

19. Magni, L., G. Bastin and V. Wertz, "Multivariable Nonlinear Predictive Control of Cement Mills", *IEEE Transactions on Control Systems Technology*, Vol.7, No.5, pp.502-508, July 1999.

20. Iplikci, S., *A Neural Network Based Local Control and Targeting Method for Chaotic Dynamics*, M.S. Thesis, Boğaziçi University, 1999.