Diploma Thesis

Bilinear Neuro-Fuzzy Indirect Adaptive Control of unknown Nonlinear Systems

Filipp N. Andreadis

Author

Technical University of Crete
Department of Electronic and Computer Engineering
Systems Division

Dedication

I would like to thank especially my supervisor professor Mr. Manolis Christodoulou for his unlimited and precious assistance which helped me a lot in order to finish this diploma thesis. A man who I had the great fortune to have met in my life and helped me in a very special way to be in the place I am today. This diploma thesis is dedicated to him.

Furthermore, I would like to thank form the bottom of my heart Mr. Ioannis Boutalis for his great help and his valuable hints along with his unlimited patience.

Last but not least, I would like to thank my parents and my girl who have always been there for me, for their moral support which has been the ally for all the difficulties I faced during this work. Thank you for being in my life. This diploma thesis is dedicated to them too.

Contents

List of Figures

Chapter 1

Abstract Chapter

1.1 Abstract

In this diploma thesis the main aspect was to develop an indirect adaptive regulation of unknown nonlinear dynamical systems. This method is based on a new Neuro-Fuzzy Dynamical Systems definition which uses the concept of *Fuzzy Dynamical Systems* (FDS) operating in conjunction with *High Order Neural Network Functions* (F-HONNFs). In this problem the plant is considered unknown, and so we first propose its approximation by a special form of a fuzzy dynamical system while in the sequel the fuzzy rules are approximated by appropriate HONNFs. Thus the identification scheme leads up to a Recurrent High Order Neural Network, which however takes into account the fuzzy output partitions of the initial FDS. This scheme does not require a-priori experts' information on the number and type of input variable membership functions making it less vulnerable to initial design assumptions. At first, we identify the system around an operation point, and then it is regulated to zero adaptively. Weight updating laws are provided for the HONNFs, which guarantee that both the identification error and the system states reach zero exponentially fast, while keeping all signals in the closed loop bounded. We assure the existence of the control signal by introducing a method of parameter hopping, which is incorporated in the weight updating law. The applicability of the method is tested on a DC Motor system, where it is shown that by following the proposed procedure one can obtain asymptotic regulation.

Chapter 2

Introductory Chapter

2.1 General Introduction

Non linear time invariant dynamical systems can be reprsented by general non-linear dynamical equations of the form

$$\dot{x} = f(x, u) \tag{2.1}$$

The mathematical description of the system under study is required, so that we are able to control it. However, the exact mathematical model of the plant, especially when this is highly complex and nonlinear, is rarely known and for this reason appropriate identification schemes have to be applied which will provide us with an approximate model of the plant.

It has been established that neural networks and fuzzy inference systems are universal approximators [1], [2], i.e., they can approximate any nonlinear function to any prescribed accuracy provided that sufficient hidden neurons and training data or fuzzy rules are available. Recently, the combination of these two different technologies has given rise to fuzzy neural or neuro fuzzy approaches, that are intended to capture the advantages of both fuzzy logic and neural networks.

The neural and fuzzy approaches are most of the time equivalent, differing between each other mainly in the structure of the approximator chosen. In order to bridge the gap between the neural and fuzzy approaches several researchers introduce adaptive schemes using a class of parameterized functions that include both neural networks and fuzzy systems[5] - [10].

In the neuro or neuro fuzzy approaches, most of the already presented works [10] - [16] deal with idirect adaptive control (trying first to identify the dynamics of the systems and then generating a control input according to the certainty equivalence principle), whereas few authors [17] and [18] face the direct approach (i.e. directly generating the control input to guarantee stability), because it is not always clear how to construct the control law without knowledge of the system dynamics.

Recently [20], [21], high order neural network function approximators (HONNFs) have been proposed in order to identify nonlinear dynamical systems of the form (2.1), approximated by a Fuzzy Dynamical System (FDS). The above approximation depends on the fact that fuzzy rules could be identified with the help of HONNFs.

In this diploma thesis HONNFs are also used for the neuro fuzzy indirect adaptive control of unknown nonlinear dynamical systems, which includes two interrelated phases: first the identification of the model-plant and second the adaptive control of it.

The identification phase usually consists of two main categories: structure identification and parameter identification. Structure identification involves finding the main input variables out of all possible, specifying the membership functions, the partition of the input space and determining the number of fuzzy rules which is often based on a substantial amount of heuristic observation to express proper strategy's knowledge. Most of structure identification methods are based on data clustering, such as subtractive clustering [12], mountain clustering [11] and fuzzy C-means clustering [9]. The above approaches require that all input-output data are ready before we start to identify the plant. So, those approaches are called off-line.

In our proposed approach structure identification is also made off-line and it is based either on human expertise or on gathered data. However, the required a-priori information obtained by linguistic information or data is very limited. The only required information is an estimate of the centers of the output fuzzy membership functions and it is not necessary on the underlying fuzzy rules, because this is automatically estimated by the HONNFs. Based on these facts the proposed method is less vulnerable to initial design assumptions.The parameter identification part is then easily addressed by HONNFs, based on the linguistic information regarding the structural identification of the output part and from the numerical data obtained from the actual system to be modeled.

One of our consideration is that the nonlinear system is affine in the control and could be approximated with the help of two independent fuzzy subsystems. Every fuzzy subsystem is approximated by a family of HONNFs, each one being related with a group of fuzzy rules. Weight updating laws are given and we prove that when the structural identification is appropriate then the error reaches zero very fast. Moreover, an appropriate state feedback is constructed in order to achieve asymptotic regulation of the output, while keeping all signals of the system bounded in the closed loop. The existence of the control signal is always assured by introducing a method of parameter hopping, which is incorporated in the weight updating law.

The diploma thesis is organized as follows. Section 3.1 presents some preliminaries related to the concept of Adaptive Fuzzy Systems(AFS) and the terminology used in the remaining thesis, while section 4.1 presets some preliminaries related to the Recurrent Neural Networks. Section 5.1 reports on the ability of HONNFs to act as fuzzy rule approximators. The indirect neuro fuzzy adaptive regulation of affine in the control dynamical systems is presented in Section 6.1, where the method of parameter hopping is explained and the associated weight adaptation laws are given. Simulation results on the control of a DC Motor sys-

tem are given in Section 7.1, showing that by following the proposed procedure one can obtain asymptotic regulation. Finally, Section 8.1 concludes the work of this diploma thesis, while the appendix includes the proofs of the theorems we used and the matlab code of the simulation of the DC motor.

Chapter 3

Preliminaries related to Adaptive Fuzzy Systems

3.1 Notion and Representation of Adaptive Fuzzy Systems

In this chapter of the diploma thesis it is briefly presented the notion of adaptive fuzzy systems and their conventional representation. It is also introduced the representation of of fuzzy systems using the fuzzy rule indicator functions, which is used for the development of the proposed method.

3.1.1 Adaptive Fuzzy Systems

Some basic characteristics of an adaptive fuzzy system representation, like the performance, complexity and adaptive law, can be quite different depending upon whether the representations is linear or nonlinear in its adjustable parameters. Adaptive fuzzy controllers depend also on the type of the adaptive fuzzy subsystems they use. According to [2], we classify adaptive fuzzy controllers into two main types:

- The fuzzy logic systems which are used in an adaptive fuzzy controller are linear in their adjustable parameters. This adaptive fuzzy controller is called *a first-type adaptive fuzzy controller*

- The fuzzy logic systems which are used in an adaptive fuzzy controller are nonlinear in their adjustable parameters. This adaptive fuzzy controller is called *a second-type adaptive fuzzy controller*

Both first and second types of adaptive fuzzy controllers are nonlinear adaptive controllers. Suppose that the adaptive fuzzy system is intended to approximate the nonlinear function $f(x)$. In the first-type adaptive fuzzy controller, Wang [2] uses the following fuzzy logic representation:

$$f(x) = \sum_{l=1}^{M} \theta_l \xi_l(x) = \theta^T \xi(x) \tag{3.1}$$

where M is the number of fuzzy rules, $\theta = (\theta_1, ..., \theta_M)^T$, $\xi(x) = (\xi_1(x), ..., \xi_M(x))^T$ and $\xi_l(x)$ is the fuzzy basis function defined by

$$\xi_l(x) = \frac{\prod_{i=1}^{n} \mu_{F_i^l}(x_i)}{\sum_{l=1}^{M} \prod_{i=1}^{n} \mu_{F_i^l}(x_i)}$$

θ_l are adjustable parameters, and $\mu_{F_i^l}$ are given membership functions of the input variables (can be Gaussian, triangular, or any other type of membership functions). Clearly, Eq. (3.1) is equivalent to the following equation assuming that $\mu_{F_i^l}$ are given: that is, $\mu_{F_i^l}$ will not change during the adaptation procedure.

$$f(x) = \frac{\sum_{l=1}^{M} y^l \left(\prod_{i=1}^{n} \mu_{F_i^l}(x) \right)}{\sum_{l=1}^{M} \left(\prod_{i=1}^{n} \mu_{F_i^l}(x) \right)} \tag{3.2}$$

In the second-type adaptive fuzzy controller, the following fuzzy logic system is used:

$$f(x) = \frac{\sum_{l=1}^{M} y^l \left(\prod_{i=1}^{n} \exp(-(\frac{x_i - x_i^l}{\sigma_i^l})^2) \right)}{\sum_{l=1}^{M} \left(\prod_{i=1}^{n} \exp(-(\frac{x_i - x_i^l}{\sigma_i^l})^2) \right)} \tag{3.3}$$

where y^l, x_i^l, σ_i^l are the adjustable parameters.

From the definitions we gave above it is apparent that the success of the adaptive fuzzy system representations in approximating the nonlinear function $f(x)$ depends on the careful selection of the fuzzy partitions of input and output variables, the selected type of the membership functions and the proper number of fuzzy rules. In approximating complex nonlinear functions, this number may become very large [8] leading to parameter explosion.

3.1.2 Fuzzy system description using rule indicator functions

Let us consider the system with input space $u \subset R^m$ and state - space $x \subset R^n$, with its i/o relation being governed by the following equation

$$z^t = f(x^t, u^t) \tag{3.4}$$

where $f(\cdot)$ is a continuous function and the superscript t denotes the temporal variable. In case the system is dynamic the above equation could be replaced by the following difference equation

$$x^{t+1} = f(x^t, u^t) \tag{3.5}$$

where the superscript t denotes the temporal variable, $t = 1, 2, ...$

By setting $y = [x, u]$ and omiting superscript t, Eq. (3.4) may be rewritten as follows

$$z = f(y) \tag{3.6}$$

In many practical situations, we are unable to measure accurately the states and inputs of a system of the form in (3.4); in most cases, we are provided with cheap sensors, expert's opinions, e.t.c which provide us with imprecise estimations of the state and input vectors. Thus, instead of vectors x and u we are provided with some linguistic variables $\tilde{x}_i$ and $\tilde{u}_i$, respectively.

Let now $\tilde{y} := (\tilde{x}, \tilde{u})$ and suppose that each linguistic variable $\tilde{y}_i$ belongs to a finite set L_i with cardinality k_i, i.e. $\tilde{y}_i$ takes one of k_i variables. Let also $\tilde{y}_{ij}$ denotes the *ith* element of the set L_i. Then we may define a function $\tilde{h}_i : R \to L_i$ to be the output function of the system in Eq. (3.6) in the case that

$$\tilde{y}_i = \tilde{h}_i(y_i) \tag{3.7}$$

Note that $\tilde{h}_i(\cdot)$ maps the real axis into a set of linguistic variables L_i , and thus $\tilde{h}_i(\cdot)$ is not defined in the usual way. In order to overcome such a problem we define the function $\tilde{h}_i : R \to \{1, 2, ..., k_i\}$ as follows

$$\tilde{h}_i(y_i) = \tilde{y}_{ij} \Longleftrightarrow h_i(y_i) = j \tag{3.8}$$

Since $h_i(\cdot)$ is very similar to $\tilde{h}_i(\cdot)$, we will call the function $h_i(\cdot)$ the *ith* output of the system in Eq. (3.6). Also, $\tilde{h}_i(\cdot)$ and consequently $h_i(\cdot)$ is related with the structural identification part mentioned in section 2.1 and arrive after using an automatic procedure based on system operation data or after consulting human experts advising on how to partition the system variables.

Following the standard approach in fuzzy systems theory we associate with each $\tilde{y}_{ij}$ a membership function $\tilde{\mu}_{ij}(y_i) \in [0, 1]$ which satisfies

$$\tilde{\mu}_{ij}(y_i) = \max_l \tilde{\mu}_{il}(y_i) \Longleftrightarrow h_i(y_i) = j \tag{3.9}$$

From the definition of the functions $\tilde{h}_i(\cdot)$ [or $h_i(\cdot)$] we have that the space $\mathcal{Y} = \mathcal{X} \times \mathcal{U}$ is partitioned in the following way: let $\mathcal{Y}_{ij}$ be defined as follows

$$\mathcal{Y}_{ij} = \{y_i \in R : h_i(y_i) = j\} \tag{3.10}$$

i.e. $\mathcal{Y}_{ij}$ denotes the set of all the variables y_i that output the same linguistic variable $\tilde{y}_{ij}$. Thus y is partitioned into disjoint subsets $\mathcal{Y}_{j_1,j_2,...,j_{n+m}}$ defined as follows

$$\mathcal{Y}_{j_1,j_2,...,j_{n+m}} := \mathcal{Y}_{1j_1} \times \cdots \times \mathcal{Y}_{(n+m)j_{n+m}}, j_i \in \{1, 2, ..., k_i\} \tag{3.11}$$

In a similar way we may define the sets $\mathcal{X}_{ij}$, $\mathcal{U}_{ij}$, $\mathcal{Z}_{ij}$ and the sets $\mathcal{X}_{j_1,j_2,...,j_n}$, $\mathcal{U}_{j_1,j_2,...,j_n}$ and $\mathcal{Z}_{j_1,j_2,...,j_n}$. Note now the following fact: for two vectors $(x^{(1)}, u^{(1)}) \in \mathcal{Y}_{j_1,j_2,...,j_{n+m}}$ and $(x^{(2)}, u^{(2)}) \in \mathcal{Y}_{j_1,j_2,...,j_{n+m}}$ there maybe

$$h_i(f_i(x^{(1)}, u^{(1)})) \neq h_i(f_i(x^{(2)}, u^{(2)})) \tag{3.12}$$

for some $i \in \{1, 2, ..., n\}$, i.e. two input vectors belonging to the same subset $\mathcal{Y}_{j_1,j_2,...,j_{n+m}}$ may point - through the vector - field $f(\cdot)$, to different subsets $\mathcal{Z}_{l_1,l_2,...,l_n}$. Let now $\Omega^{l_1,l_2,...,l_n}_{j_1,j_2,...,j_{n+m}}$ be defined as the subset of $\mathcal{Y}_{j_1,j_2,...,j_{n+m}}$ that points - through the vector - field $f(\cdot)$, to the subsets $\mathcal{Z}_{l_1,l_2,...,l_n}$, i.e

$\Omega^{l_1,l_2,...,l_n}_{j_1,j_2,...,j_{n+m}} :=$

$$= \{(x,u) \in y_{j_1,j_2,...,j_{n+m}} : h_1(z_1) = l_1, ..., h_n(z_n) = l_n\}$$

and define the transition possibilities $\pi^{l_1,...,l_n}_{j_1,...,j_{n+m}}$ as follows

$$\pi^{l_1,...,l_n}_{j_1,...,j_{n+m}} := \frac{\int_{(x,u)\in\Omega^{l_1,...,l_n}_{j_1,...,j_{n+m}}} dXdU}{\int_{(x,u)\in y_{j_1,...,j_{n+m}}} dXdU} \tag{3.13}$$

where $\pi^{l_1,...,l_n}_{j_1,...,j_{n+m}}$ is a number belonging to a set [0,1] that represents the fraction of the vectors (x,u) in $y_{j_1,...,j_{n+m}}$ that points - through the vector field $f(\cdot)$ to the set $\chi_{l_1,...,l_n}$. Obviously

$$\sum_{l_1,...,l_n} \pi^{l_1,...,l_n}_{j_1,...,j_{n+m}} = 1 \tag{3.14}$$

In order to present the lemma of Section 5.1, we define the indicator function: Let $I^{l_1,l_2,...,l_n}_{j_1,j_2,...,j_{n+m}}$ denote the indicator function of the subset $\Omega^{l_1,l_2,...,l_n}_{j_1,j_2,...,j_{n+m}}$, that is,

$$I^{l_1,...,l_n}_{j_1,...,j_{n+m}}(x,u) = \begin{Bmatrix} 1 & if\,(x,u) \in \Omega^{l_1,...,l_n}_{j_1,...,j_{n+m}} \\ 0 & otherwise \end{Bmatrix} \tag{3.15}$$

Using the above definitions, we can see that the system in Eq. (3.6) is described by fuzzy rules of the form

$$R^{l_1,...,l_n}_{j_1,...,j_{n+m}} \Leftrightarrow \begin{Bmatrix} IF\ y_1\ is\ \tilde{y}_{1j_1}\ AND... \\ AND\ y_{n+m}\ is\ \tilde{y}_{(n+m)j_{n+m}} \\ THEN \\ z_1\ is\ \tilde{z}_{1l_1}\ AND...AND\ z_n\ is\ \tilde{z}_{nl_n} \\ with\ possibility\ \pi^{l_1,...,l_n}_{j_1,...,j_{n+m}} \end{Bmatrix} \tag{3.16}$$

where obviously $\tilde{y}_{ij_i} = \tilde{h}_i(y^t_i)$ and $\tilde{z}_{il_i} = \tilde{h}_i(z_i) = \tilde{h}_i(f_i(x,u))$.

In the above notation, if $j_1 = l_1$, $j_2 = l_2$ and ... and $j_n = l_n$, then these points participate to the definition of the same fuzzy rule. If $j_1 \neq l_1$ or $j_2 \neq l_2$ or or $j_n \neq l_n$, then these points define alternative fuzzy rules describing this transition. Consider now the next definition.

Definition 1 *A Fuzzy System - (FS) is a set of Fuzzy Rules of the form* ($R^{l_1,l_2,...,l_n}_{j_1,j_2,...,j_{n+m}}$); *the system in Eq. (3.4) is called the Underlying System - (US) of the previously defined FS. Alternatively, the system in Eq. (3.4) will be called a Generator of the FS that is described by the rules* ($R^{l_1,l_2,...,l_n}_{j_1,j_2,...,j_{n+m}}$).

Due to the linguistic description of the variables of the FS it is not rare to have more than one systems of the form in Eq. (3.6) to be generators for the FS that is described by the rules ($R^{l_1,l_2,...,l_n}_{j_1,j_2,...,j_{n+m}}$).

Define now the following system

$$z = \sum \bar{z}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}} \times I^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(\chi, u) \tag{3.17}$$

Where $\bar{z}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}} \in R^n$ be any vector satisfying $h_i(\bar{z}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(i)) = l_i$ where $\bar{z}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(i)$ denotes the i^{th} entry of $\bar{z}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}$ Then, according to [20], [21] the system in (3.17) is a generator for the FS $(R^{l_1,l_2,\ldots,l_n}_{j_1,j_2,\ldots,j_{n+m}})$.

It is obvious that Eq. (3.17) can be also valid for dynamic systems. In its dynamical form it becomes

$$\chi^{t+1} = \sum \bar{x}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}} \times I^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(\chi^t, u^t) \tag{3.18}$$

Where $\bar{x}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}} \in R^n$ be any vector satisfying $h_i(\bar{x}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(i)) = l_i$ where $\bar{x}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(i)$ denotes the i^{th} entry of $\bar{x}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}$.

Chapter 4

The RHONNs

4.1 Identification of Dynamical Systems using RHONNs

The use of multi-layer neural networks for pattern recognition and for modeling of "static" systems is currently well-known. Given pairs of input-output data (which may be related by an unknown algebraic relation, a so-called "static" function) the network is trained to learn the particular input-output map. Theoretical work by several researchers, including [23], and [24], have proven that, even with one hidden layer, neural networks can approximate any continuous function uniformly over a compact domain, provided the network has a sufficient number of neural networks for modeling and identification of dynamical systems. These networks, which naturally involve dynamic elements in the form of feedback connections, are known as recurrent neural networks.

Several training methods for recurrent networks have been proposed in the literature. Most of these methods rely on the gradient methodology and involve the computation of partial derivatives, on sensitive functions. In this respect, they are extensions of the backpropagation algorithm for feedforward neural networks [25]. Examples of such learning algorithms include the recurrent backpropagation [26], the backpropagation-through-time algorithms [29], the real-time recurrent learning algorithm [30], and the dynamic backpropagation [28] algorithms. The last approach is based on the computation of sensitivity models for generalized neural networks. These generalized neural networks, which were originally proposed in [27], combine feedforward neural networks and dynamical components in the form of stable rational transfer functions.

Although the training methods mentioned above have been used successfully in many empirical studies, they share some fundamental drawbacks. One drawback is the fact that, in general, they rely on some type of approximation for computing the partial derivative. Furthermore, these training methods require a great deal of computational time. A third disadvantage is the inability to obtain analytical results concerning the convergence and stability of these schemes.

Recently, there has been a concentrated effort towards the design and analysis of learning algorithms that are based on the Lyapunov stability theory [31], [32], [34], [33], [35], [36], [37], [38], [39] targeted at providing stability, convergence and robustness proofs, in this way, bridging the existed gap between theory and applications.

In this section we discuss the identification problem which consists of choosing an appropriate identification model and adjusting its parameters according

to some adaptive law, such that the response of the model to an input signal (or a class of input signals), approximates the response of the real system to the same input. Since a mathematical characterization of a system is often a prerequisite to analysis and controller design, system identification is important not only for understanding and predicting the behavior of the system, but also for obtaining an effective control law. For identification models we use recurrent high-order neural networks. High-order networks are expansions of the first-order Hopfield [40] and Cohen-Grossberg [41] models that allow higher-order interactions between neurons. The superior storage capacity of has been demonstrated in [42], [43], while the stability properties of these models for fixed-weight values have been studied in [44],[45]. Furthermore, several authors have demonstrated the feasibility of using these architectures in applications such as grammatical inference [46] and target detection [47].

The idea of recurrent neural networks with dynamical components distributed throughout the network in the form dynamical neurons and their application for identification of dynamical systems was proposed in [39]. In this section we combine distributed recurrent networks with high-order connections between neurons. At first we show that recurrent high-order neural networks are capable of modeling a large class of dynamical systems. In particular, it is shown that if enough higher-order connections are allowed in the network then there exist weight values such that the input-output behavior of the RHONN model approximates that of an arbitrary dynamical system whose state trajectory remains in a compact set. In the sequel, we develop weight adjustment laws for system identification under the assumption that the system to be identified can be modeled exactly by the RHONN model. It is shown that these adjustment laws guarantee boundedness of all the signals and weights and furthermore, the output error converges to zero. Then, this analysis is extended to the case where there is a nonzero mismatch between the system and the RHONN model with optimal weight values. We apply this methodology to the identification of a simple robotic manipulator system and some final conclusions are drawn.

4.2 The RHONN Model

Recurrent neural networks (RNN) models are characterized by a two way connectivity between units (i.e. ,neurons). This distinguishes them from feedward neural networks, where the output of the unit is connected only to inputs of the next layer. In the most simple case, the state history of each neuron is governed by a differential equation of the form:

$$\dot{x}_i = -a_i x_i + b_i \sum_j w_{ij} y_j \tag{4.1}$$

Where x_i is the state of the $i-th$ neuron,a_i,b_i are constants, w_{ij} is the synaptic weight connecting the $j-th$ input to the $i-th$ neuron and y_j is the $j-th$ input to the above neuron. Each y_j is either an external input or the state of a neuron passed through a sigmoid function (i.e., $y_j = s(x_j)$), where $s(\cdot)$ denotes the sigmoid nonlinearity.

The dynamic behavior and the stability properties of neural network models of the form (4.1) have been studied extensively by various researchers [40],[41],[45],[44]. These studies exhibited encouraging results in application areas such as associative memories, but they also revealed the limitations inherent in such a simple model.

In a recurrent second order neural network, the input to the neuron is not only a linear combination of the components y_i, but also of their product $y_i y_k$. One can pirsue this line further to include higher order interactions represented by triplets $y_i y_k y_l$, quadraplets, etc. forming the recurrent high order neural networks (RHONNs).

Let us now consider a RHONN consisting of n neurons and m inputs. The state of each neuron is governed by a differential equation of the form:

$$\dot{x}_i = -a_i x_i + b_i \left[\sum_{k=1}^{M} w_{ik} \prod_{j \in I_k} y_j^{d_j(k)} \right] \tag{4.2}$$

Where $\{I_1, I_2, ..., I_L\}$ is a collection of L not-ordered subsets of $\{1, 2, ..., m+n\}$, a_i,b_i are real coefficients, w_{ik} are the adjustable synaptic weights of the neural network and $d_j(k)$ are non-negative integers. The state of the $i-th$ input neuron is again represented by x_i and $y := [y_1, y_2 \cdots y_{m+n}]^T$ is the input vector to each neuron defined by:

$$y = \begin{Bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \\ y_{n+1} \\ . \\ . \\ . \\ y_{n+m} \end{Bmatrix} = \begin{Bmatrix} s(x_1) \\ s(x_2) \\ . \\ . \\ . \\ s(x_n) \\ u_1 \\ u_2 \\ . \\ . \\ . \\ u_m \end{Bmatrix} \tag{4.3}$$

where $u := [u_1, u_2 \cdots u_m]^T$ is the external input vector to the network. The function $s(\cdot)$ is monotone-increasing, differentiable and is usually represented by sigmoids of the form:

$$s(x) = a\frac{1}{1+e^{-\beta x}} - \gamma \tag{4.4}$$

where the parameters a,β represent the bound and slope of sigmoid's curvature and γ is a bias constant. In the special case where $a = \beta = 1$,$\gamma = 0$, we obtain the logistic function and by setting $a = \beta = 2$,$\gamma = 1$, we obtain the hyperbolic tangent function. These are the sigmoid activation functions most commonly used in neural network applications.

We now introduce the $L-$dimensional vector z, which is defined as

$$z = \begin{Bmatrix} z_1 \\ z_2 \\ \cdot \\ \cdot \\ \cdot \\ z_L \end{Bmatrix} = \begin{Bmatrix} \prod\limits_{j \in I_1} y_j^{d_j(1)} \\ \prod\limits_{j \in I_2} y_j^{d_j(2)} \\ \cdot \\ \cdot \\ \cdot \\ \prod\limits_{j \in I_L} y_j^{d_j(L)} \end{Bmatrix} \tag{4.5}$$

Hence, the RHONN model (4.2) becomes

$$\dot{x}_i = -a_i x_i + b_i \left[\sum_{k=1}^{L} w_{ik} z_k \right]. \tag{4.6}$$

Moreover, if we define the adjustable parameter vector as

$w_i = b_i[w_{i1}, w_{i2} \cdots w_{iL}]^T$,

then (4.6) becomes

$$\dot{x}_i = -a_i x_i + w_i^T z. \tag{4.7}$$

The vectors $[w_i : i = 1, 2, \cdots, n]$ represent the adjustable weights of the network, while the coefficients $[a_i : i = 1, 2, \cdots, n]$ are part of the underlying network architecture and are fixed during training.

In order to guarantee that each neuron x_i is bounded-input bounded-output (BIBO) stable, we shall assume that $[a_i > 0, i = 1, 2, \cdots, n]$. In the special case of a continuous-time Hopfield model [40], we have $a_i = \frac{1}{R_i C_i}$, where $R_i > 0$ and $C_i > 0$ are the resistance and capacitance connected at the $i - th$ node of the network respectively.

The dynamic behavior of the overall network is described by expressing (4.7) in vector notation as:

$$\dot{x}_i = Ax + W^T z, \tag{4.8}$$

where $x = [x_1, x_2, \cdots, x_n]^T \in R^n, W = [w_1, w_2, \cdots, w_n]^T \in R^{Lxn}$ and $A = diag\,[-a_1, -a_2, \cdots, -a_n]$ is a nxn diagonal matrix. Since $[a_i > 0, i = 1, 2, \cdots, n]$, A is a stability matrix. Although it is not written explicitly, the vector z is a function of both the neural network state x and the external input u.

4.2.1 Approximation Properties

Consider now the problem of approximating a general nonlinear dynamical system whose input-output behavior is given by

$$\dot{\chi} = F(\chi, u), \tag{4.9}$$

where $\chi \in R^n$ is the system state, $u \in R^n$ is the system input and $F : R^{n+m} \rightarrow R^n$ is a smooth vector field defined on a compact set $\mathcal{Y} \subset R^{n+m}$.

The approximation problem consists of determining whether by allowing enough higher-order connections, there exists weights W, such that the RHONN model approximates the input-output behavior of an arbitrary dynamical system of the form (4.9).

In order to have a well-posed problem, we assume that F is continuous and satisfies a local Lipschitz condition such that (4.9) has a unique solution, in the sense of Caratheodory [50], and $\{\chi(t), u(t)\} \in \mathcal{Y}$ for all t in some time interval

$J_T = \{t : 0 \leq t \leq T\}$. The interval J_T represents the time period over which the approximation is to be performed. Based on the above assumptions we obtain the following result:

Theorem 1 *Suppose that the system (4.9) and the model (4.8) are initially at the same state $x(0) = \chi(0)$, then for any $\epsilon > 0$ and any finite $T > 0$, there exists an integer L and a matrix $W^* \in R^{Lxn}$ such that the state $x(t)$ of the RHONN model (4.8) with L high-order connections and weight values $W = W^*$ satisfies:*

$$\sup_{0 \leq t \leq T} |x(t) - \chi(t)| \leq \epsilon.$$

The proof of the above theorem can be studied in the Appendix.

The above theorem proves that if sufficiently large number of connections is allowed in the RHONN model then it is possible to approximate any dynamical system to any degree of accuracy. This is strictly an existence result; it does not provide any constructive method for obtaining the optimal weights W^*. In what follows, we consider the learning problem of adjusting the weights adaptively, such that the RHONN model identifies general dynamic systems.

4.3 Learning Algorithms

In this section we develop weight adjustment laws under the assumption that the unknown system is modeled exactly by a RHONN architecture of the form (4.8). This analysis is extended in the next section to cover the case where there exists a nonzero mismatch between the system and the RHONN model with optimal weights values. This mismatch is referred to as modeling error.

Although the assumption of no modeling error is not very realistic, the identification procedure of this section is useful for two reasons:

- The analysis is more straightforward and thus easier to understand.
- The techniques developed for the case of no modeling error are also very important in the design of weight adaptive laws in the presence of modeling errors.

Based on the assumption of no modeling error, there exist unknown weight vectors $w_i^*, i = 1, 2, \cdots, n$, such that each state χ_i of the unknown dynamic system (4.9) satisfies:

$$\dot{\chi}_i = -a_i\chi_i + w_i^* z(\chi, u), \;\; \chi_i(0) = \chi_i^0. \tag{4.10}$$

where χ_i^0 is the initial $i-th$ state of the system. In the following, unless there is no confusion, the arguments of the vector field z will be omitted.

As in standard in system identification procedures, we will assume that the input $u(t)$ and the state $\chi(t)$ remain bounded for all $t \geq 0$. Based on the definition of $z(\chi, u)$, as given in (4.5), this implies that $z(\chi, u)$ is also bounded. In the sections that follow we present different approaches for estimating the unknown parameters w_i^* of the RHONN model.

4.3.1 Filter Regressor RHONN

The following lemma is useful in the development of the adaptive identification scheme presented in this section.

Lemma 1 *The system described by*

$$\dot{\chi}_i = -a_i\chi_i + w_i^* z(\chi, u), \;\; \chi_i(0) = \chi_i^0 \tag{4.11}$$

can be expressed as

$$\dot{\zeta}_i = -a_i\zeta_i + z_i, \;\; \zeta_i(0) = 0, \tag{4.12}$$

$$\chi_i = w_i^{*T}\zeta_i + e^{-a_i t}\chi_i^0 \tag{4.13}$$

The proof of the above lemma can be studied in the Appendix.

Using *Lemma* 1, the dynamical system described by (4.9) is rewritten as

$$\chi_i = w_i^{*T}\zeta_i + \epsilon_i, \quad i = 1, 2, \cdots, n, \tag{4.14}$$

where ζ_i is a filtered version of the vector z(as described by (4.5)) and $\epsilon_i := e^{a_i t}\chi_i^0$ is an exponentially decaying term which appears if the system is in a nonzero initial state. By replacing the unknown weight vector w_i^* in (4.14), by its estimate w_i and ignoring the exponentially decaying term ϵ_i, we obtain the RHONN model:

$$x_i = w_i^T \zeta_i, \;\; i = 1, 2, \cdots, n. \tag{4.15}$$

The exponentially decaying term $\epsilon_i(t)$ can be omitted in (4.15) since, as we shall see later, it does not affect the convergence properties of the scheme. The state error $e_i = x_i - \chi_i$ between the system and the model satisfies:

$$e_i = \phi_i^T \zeta_i - \epsilon_i, \tag{4.16}$$

where $\phi_i = w_i - w_i^*$ is the *weight estimation error*. The problem now is to derive suitable adaptive laws for adjusting the weights w_i, for $i = 1, \cdots, n$. This can be achieved by using well-known optimization techniques for minimization of the *quadratic cost functional*

$$J(w_1, \cdots, w_n) = \frac{1}{2} \sum_{i=1}^{n} e_i^2 = \frac{1}{2} \sum_{i=1}^{n} \left[(w_i - w_i^*)^T \zeta_i - \epsilon_i \right]^2. \tag{4.17}$$

Depending on the optimization method that is employed, different wight adjustment laws can be derived. Here, we consider the gradient and the least-squares method [51]. The gradient method yields

$$\dot{w}_i = -\Gamma_i \zeta_i e_i, \;\; i = 1, 2, \cdots, n, \tag{4.18}$$

where Γ_i is a positive definite matrix referred to as the adaptive gain or learning rate. With this we obtain

$$\begin{cases} \dot{w}_i = -P_i \zeta_i e_i \\ \dot{P}_i = -P_i \zeta_i \zeta_i^T P_i \end{cases} \quad i = 1, 2, \cdots, n \tag{4.19}$$

where $P(0)$ is a symmetric positive definite matrix. In the above formulation, the least-squares algorithm can be thought of as a gradient algorithm with a time-varying learning rate.

The stability and convergence properties of the weight adjustment laws given by (4.18),(4.19) are well-known in the adaptive control literature(see, for example, [49],[52]).

Theorem 2 *Consider the RHONN model given by*

$$x_i = w_i^T \zeta_i, \;\; i = 1, 2, \cdots, n, \tag{4.20}$$

whose parameters are adjusted according to:

$$\dot{w}_i = -\Gamma_i \zeta_i e_i, \;\; i = 1, 2, \cdots, n, \tag{4.21}$$

where Γ_i is a positive definite matrix referred to as the adaptive gain or learning rate.
Then for $i = 1, 2, \cdots, n$ it is proved that:

a) $e_i, \phi_i \in L_\infty$ *(e_i and ϕ are uniformly bounded)*
b) $\lim_{t\to\infty} e_i(t) = 0$

The proof of the above theorem can be studied in the Appendix.

Remark 1 *The stability proof for the least-square algorithm:*

$$\dot{w}_i = -P_i \zeta_i e_i \dot{P}_i = -P_i \zeta_i \zeta_i^T P_i \tag{4.22}$$

where $i = 1, 2, \cdots, n$, where $P(0)$ is a symmetric positive definite matrix. In the above formulation, the least-squares algorithm can be thought of as a gradient with a time-varying learning rate.

proceeds along the same lines as in the proof of the previous theorem by considering the Lyapunov function:

$$V = \tfrac{1}{2} \sum_{i=1}^{N} (\phi_i^T P_i^{-1} \phi_i + \int_t^\infty \epsilon_i^2(\tau) d\tau).$$

A problem that may be encountered in the application of the least-squares algorithm is that P may become arbitrarily small and thus slow down adaptation in some directions [51],[49]. This so-called problem can be prevented by using one of various modifications which prevent $P(t)$ form going to zero. One such modification is the so-called, where if the smallest eigenvalue of $P(t)$ becomes smaller than ρ_1 then $P(t)$ is reset to $P(t) = \rho_o I$, where $\rho_o \geq \rho_1 > 0$ are some design constraints.

Remark 2 *The above theorem does not imply that the weight estimation error $\phi_i = w_i - w_i^*$ converges to zero. In order to achieve convergence of the weights to their correct value the additional assumption of persistent excitation needs to be persistently exciting if there exist positive scalars c and d and T such that for all $t \geq 0$*

$$cI \leq \int_{t}^{t+T} \zeta_i(\tau)\zeta_i(\tau)^T d\tau \leq dI,$$

where is the LxL identity matrix.

Remark 3 *The learning algorithms developed above can be extended to the case where the underlying neuron structure is governed by the higher-order Cohen-Grossberg model [41],[44]:*

$$\dot{x}_i = -a_i(x_i)\left\{ b_i(x_i) + \sum_{k=1}^{L} w_{ik} \prod_{j \in I_k} y_j^{d_j(k)} \right\} \tag{4.23}$$

where $a_i(\cdot), b_i(\cdot)$ satisfy certain conditions required for the boundedness of the state variables [44]. It can be seen readily that in (4.23) the differential equation is still linear in the weights and hence a similar parameter estimation procedure can be applied.

The filtered-regressor RHONN model considered in this subsection relies on filtering the vector z, which is sometimes referred to as the regressor vector. By using this filtering technique, it is possible to obtain a very simple algebraic expression for the error, which allows the application of well-known optimization procedures for designing and analyzing weight adjustment laws but there is an important drawback to this method, namely the complex configuration and heavy computational demands required in the filtering of the regressor. Generally, the dimension of the regressor will be larger than the dimension of the system, i.e., $L > n$, it might be very expensive computationally to employ to many filters. In the next subsection we consider a simple structure that requires only n filters and hence, fewer computations.

4.3.2 Filtered Error RHONN

In developing this identification scheme we start again from the differential equation that describes the unknown system, i.e.,

$$\dot{\chi}_i = -a_i\chi_i + w_i^{*T} z, \;\; i = 1, 2, \cdots, n. \tag{4.24}$$

Based on (4.24), the identifier is now chosen as:

$$\dot{x}_i = -a_i x_i + w_i^T z, \;\; i = 1, 2, \cdots, n. \tag{4.25}$$

where w_i is again the estimate of the unknown vector w_i^*. In this case the state error $e_i := x_i - \chi_i$ satisfies:

$$\dot{e}_i = -a_i e_i + \phi_i^T z, \;\; i = 1, 2, \cdots, n. \tag{4.26}$$

where $\phi_i = w_i - w_i^*$. The weights w_i, for $i = 1, 2, \ldots, n$ are adjustable according to the learning laws:

$$\dot{w}_i = -\Gamma_i z e_i, \tag{4.27}$$

where the adaptive gain Γ_i is a positive definite LxL matrix. In the special case that $\Gamma_i = \gamma_i I$, where $\gamma_i > 0$ is a scalar, then Γ_i in (4.27) can be replaced by γ_i.

The next theorem shows that the identification scheme has similar convergence properties as the filtered regressor RHONN model with the gradient method for adjusting the weights.

Theorem 3 *Consider the filtered error RHONN model given by (4.25) whose weights are adjustable according to (4.27). Then for* $i = 1, 2, \ldots, n$

$$\begin{array}{c} \textit{(a)}\ e_i, \phi_i \in L_\infty \\ \textit{(b)}\ \lim_{t\to\infty} e_i(t) = 0 \end{array}$$

The proof of the above theorem can be studied in the Appendix.

4.4 Robust Learning Algorithms

The derivation of the learning algorithms developed in the previous section made the crucial assumption of no modeling error. Equivalently, it was assumed that there exist weight vectors w_i^*, for $i = 1, 2, \cdots, n$ such that each state of the unknown dynamical system (4.9) satisfies

$$\dot{\chi}_i = -a_i \chi_i + w_i^{*T} z(\chi, u) \tag{4.28}$$

In many cases this assumption will be violated. This is mainly due to an insufficient number of higher-order terms in the RHONN model. In such cases, if standard adaptive laws are used for updating the weights, then the presence of the modeling error in problems related to learning in dynamic environments, may cause the adjusted weight values (and, consequently, the error $e_i = x_i - \chi_i$) to drift to infinity. Examples of such behavior, which is usually referred to as, can be found in the adaptive control literature of linear systems [51],[52].

In this section we shall modify the standard weight adjustment laws in order to avoid the parameter drift phenomenon. These modified weight adjustment laws will be referred to as *robust learning algorithms*.

In formulating the problem it is noted that by adding and subtracting $a_i\chi_i + w_i^{*T} z(\chi, u) + v_i(t)$, the dynamic behavior of each state of the system (4.9) can be expressed by a differential equation of the form:

$$\dot{\chi}_i = -a_i\chi_i + w_i^{*T} z(\chi, u) + v_i(t) \tag{4.29}$$

where the modeling error $v_i(t)$ is given by

$$v_i(t) := F_i(\chi(t), u(t)) + a_i\chi(t) - w_i^{*T} z(\chi(t), u(t)) \tag{4.30}$$

The function $F_i(\chi, u)$ denotes the $i-th$ component of the vector field $F(\chi, u)$, while the unknown optimal weight vector w_i^* is defined as the value of the weight vector w_i that minimizes the L_∞-norm difference between $F(\chi, u) + a_i\chi$ and $w_i^T z(\chi, u)$ for all $(\chi, u) \in \mathcal{Y} \subset R^{n+m}$, subject to the constraint that $|w_i| \leq M_i$, where M_i is a large design constraint. The region $\mathcal{Y}$ denotes the smallest compact subset of R^{n+m} that includes all the values that (χ, u) can take, i.e., $(\chi(t), u(t)) \in \mathcal{Y}$ for all $t \geq 0$. Since by assumption $u(t)$ is uniformly bounded and the dynamical system to be identified is BIBO stable, the existence of such $\mathcal{Y}$ is assured. It is pointed out that in our analysis we do not require knowledge of the region $\mathcal{Y}$, nor upper bounds for the modeling error $v_i(t)$.

In summary, for $i = 1, 2, \ldots, n$, the optimal weight vector w_i^* is defined as

$$w_i^* := \arg \min_{|w_i| \leq M_i} \left\{ \sup_{(\chi,u) \in \mathcal{Y}} |F_i(\chi, u) + a_i\chi - w_i^T z(\chi, u)| \right\} \tag{4.31}$$

The reason for restricting w_i^* to a ball of radius M_i is twofold: firstly, to avoid any numerical problems that m ay arise owing to having weight values that are too large, and secondly, to allow the use of the σ-modification [51], which will be developed below to handle the parameter drift problem. The formulation developed above follows the methodology of [31] closely. Using this formulation, we now have a system of the form (4.29) instead of (4.28). It is also noted that since $\chi(t)$ and $u(t)$ are bounded, the modeling error $v_i(t)$ is also bounded, i.e., $\sup_{t\geq 0} |v_i(t)| \leq 0$ for some finite constant $\bar{v}_i$.

In what follows we develop robust learning algorithms based on the filtered error RHONN identifier; however, the same underlying idea can be extended readily to the filtered-regressor RHONN. Hence, the identifier is chosen as in (4.25), i.e.,

$$\dot{x}_i = -a_i x_i + w_i^T z, \;\; i = 1, 2, \cdots, n \tag{4.32}$$

where w_i is the estimate of the unknown optimal weight vector w_i^*. Using (4.29),(4.32), the state error $e_i = x_i - \chi_i$ satisfies

$$\dot{e}_i = -a_i e_i + \phi_i^T z - v_i, \tag{4.33}$$

where $\phi_i = w_i - w_i^*$. Owing to the presence of the modeling error v_i, the learning laws given by (4.27) are modified as follows:

$$\dot{w}_i = \begin{Bmatrix} -\Gamma_i z e_i, \;\; if |w_i| \leq M_i \\ -\Gamma_i z e_i - \sigma_i \Gamma_i w_i, \;\; if |w_i| > M_i \end{Bmatrix} \tag{4.34}$$

where σ_i is a positive constant chosen by the designer. The above weight adjustment law is the same as (4.27) if w_i belongs to a ball of radius M_i. In the case that the weight leave this ball, the weight adjustment law is modified by the addition of the leakage term $\sigma_i \Gamma_i w_i$, whose objective is to prevent the weight values from drifting to infinity. This modification is known as the [51].

In the following theorem we use the vector notation $v := [v1, \cdots, v_n]^T$ and $e := [e_1, \cdots, e_n]^T$.

Theorem 4 *Consider the filtered error RHONN model given by (4.32) whose weights are adjusted according to (4.34). Then for* $i = 1, 2, \cdots, n$

$$\begin{array}{c} \textit{(a)}\ e_i, \phi_i \in L_\infty \\ \textit{(b)there exist constants}\ \lambda, m\ \textit{such that} \end{array}$$

$$\int_0^t |e(\tau)^2| d\tau \leq \lambda + m \int_0^t |v(\tau)^2| d\tau$$

The proof of the theorem can be studied in the Appendix.

Remark 4 *It is noted that the* σ *modification causes the adaptive law (4.34) to be discontinuous; therefore standard existence and uniqueness results of solutions to differential equations are in general not applicable. In order to overcome the problem of existence and uniqueness of solutions, the trajectory behavior of* $w_i(t)$ *can be made "smooth" on the discontinuity hypersurface* $\{w_i \in R^L : |w_i| = M_i\}$ *by modifying the adaptive law (4.34) to*

$$\dot{w}_i = \left\{ \begin{array}{c} -\Gamma_i z_i e_i, \ \ if\ \ \{|w_i| < M_i\}\ \ or\ \ \{|w_i| = M_i\ \ and\ \ w_i^T \Gamma_i z e_i > 0\} \\ \frac{-\Gamma_i z e_i + w_i^T \Gamma_i z e_i}{w_i^T \Gamma_i w_i} \Gamma_i w_i \ \ , if\ \ \{|w_i| = M_i\}\ \ and\ \ \{-\sigma_i w_i^T \Gamma_i w \leq w_i^T \Gamma_i z e_i \leq 0\} \\ -\Gamma_i z e_i - \sigma_i \Gamma_i w_i \ \ , if\ \ \{|w_i| > M_i\}\ \ or\ \ \{|w_i| = M_i\}\ \ and\ \ \{w_i^T \Gamma_i z e_i < -\sigma_i w_i^T \Gamma_i w\} \end{array} \right\} \tag{4.35}$$

As shown in [53], the adaptive law (4.35) retains all the properties of (4.34) and, in addition, guarantees the existence of a unique solution, in the sense of Caratheodory [50]. The issue of existence and uniqueness of solutions in adaptive systems is treated in detail in [53].

Chapter 5

The HONNF's

5.1 The HONNF's as Fuzzy Rule Approximators

The main idea in presenting the basic result of this section of the thesis lies on the fact that functions of high order neurons are capable of approximating discontinuous functions. So, we use high order neural networks functions in order to approximate the indicator functions $I^{l_1,...,l_n}_{j_1,...,j_{n+m}}$ .g disc However, in order the approximation problem to make sense the space $\mathcal{Y} := \mathcal{X} \times \mathcal{U}$ must be compact. Thus, our first assumption is the following:

(**A.1**) $\mathcal{Y} := \mathcal{X} \times \mathcal{U}$ is a compact set.

Notice that since $\mathcal{Y} \subset \Re^{n+m}$ the above assumption is identical to the assumption that it is closed and bounded. Also, it is noted that even if $\mathcal{Y}$ is not compact we may assume that there is a time instant T such that (x^t, u^t) remain in a compact subset of $\mathcal{Y}$ for all $t < T$; i.e. if $\mathcal{Y}_T := \{(x^t, u^t) \in \mathcal{Y}, t < T\}$ We may replace assumption (A.1) by the following assumption

(**A.2**) $\mathcal{Y}_T$ is a compact set.

It is worth noticing, that while assumption (A.1) requires the system in Eq. (3.5) solutions to be bounded for all $u^t \in U$ and $x^0 \in X$, assumption (A.2) requires the system in Eq. (3.5) solutions to be bounded for a finite time period; thus, assumption (A.1) requires the system in Eq. (3.5) to be BIBS stable while assumption (A.2) is valid for systems that are not BIBS stable and, even more, for unstable systems and systems with finite escape times.

We are now ready to show that high order neural network functions are capable of approximating the indicator functions $I^{l_1,...,l_n}_{j_1,...,j_{n+m}}$ Let us define the following high order neural network functions (HONNFs).

$$N(x, u; w, L) = \sum_{k=1}^{L} w_k \prod_{j \in I_k} \Phi_j^{d_j(k)} \tag{5.1}$$

Where $\{I_1, I_2, ..., I_L\}$ is a collection of L not-ordered subsets of $\{1, 2, ..., m + n\}$, $d_j(k)$ are non-negative integers, Φ_j are sigmoid functions of the state or the input, which are the elements of the following vector

$$\Phi = \begin{bmatrix} \Phi_1 \\ . \\ . \\ . \\ \Phi_n \\ \Phi_{n+1} \\ . \\ . \\ . \\ \Phi_{m+n} \end{bmatrix} = \begin{bmatrix} S(x_1) \\ . \\ . \\ . \\ S(x_n) \\ S(u_1) \\ . \\ . \\ . \\ S(u_m) \end{bmatrix} \tag{5.2}$$

where

$$S(u) \, or \, S(x) = a\frac{1}{1+e^{-\beta x}} - \gamma \tag{5.3}$$

and $w := [w_1 \cdots w_L]^T$ are the HONNF weights. Eq. (5.1) can also be written

$$N(x,u;w,L) = \sum_{k=1}^{L} w_k s_k(x,u) \tag{5.4}$$

Where $s_k(x,u)$ are high order terms of sigmoid functions of the state and/or input.

The next lemma [20] states that a HONNF of the form in Eq. (5.4) can approximate the indicator function $I_{j_1,\ldots,j_{n+m}}^{l_1,\ldots,l_n}$.

Lemma 2 *Consider the indicator function $I_{j_1,\ldots,j_{n+m}}^{l_1,\ldots,l_n}$ and the family of the HONNFs $N(x,u;w,L)$. Then for any $\epsilon > 0$ there is a vector of weights $w^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n}$ and a number of $L^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n}$ high order connections such that*

$$\sup_{(x,u)\in\bar{Y}} \{I_{j_1,\ldots,j_{n+m}}^{l_1,\ldots,l_n}(x,u) - \\ -N(x,u;w^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n}, L^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n})\} < \varepsilon$$

where $\bar{y} \equiv y$ if assumption (A.1) is valid and $\bar{y_T} \equiv y$ if assumption (A.2) is valid.

Let us now keep $L^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n}$ constant, i.e. let us preselect the number of high order connections, and let us define the optimal weights of the HONNF with $L^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n}$ high order connections as follows

$$\bar{w}^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n} := \arg \min_{w\in R^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n}} \times$$

$$\left\{ \sup_{(x,u)\in\bar{y}} \left| I^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(x,u) - N(x,u;w,L^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n}) \right| \right\}$$

and the modelling error as follows

$$\nu^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(x,u) = I^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(x,u) -$$

$$-N(x,u;w^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n}, L^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n})$$

It is worth noticing that from Lemma 2 we have that $\sup_{(x,u)\in\bar{y}} \left| \nu^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(x,u) \right|$ can be made arbitrarily small by simply selecting appropriately the number of high order connections.

Using the approximation Lemma 2 it is natural to approximate system in Eq. (3.18) by the following dynamical system

$$z^{t+1} = \sum \bar{x}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(x,u) \times$$

$$\times N(z^t,u^t;w^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n}, L^{j_1,\ldots,j_{n+m};l_1,\ldots,l_n})$$

Let now $\chi^t(\chi^0,u^t)$ denote the solution in Eq. (3.18) given that the initial state at $t=0$ is equal to χ^0 and the input is u^t. Similarly we define $z^t(z^0,u^t)$. Also let

$$\nu(z^t,u^t) = \sum (\bar{x}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(x,u) \times \nu^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}(z^t,u^t)) \tag{5.5}$$

Then, it can be easily shown that

$$z^t(z^0,u^t) = \chi^t(z^0,u^t) + \nu(z^t,u^t) \tag{5.6}$$

Note now that from the approximation Lemma 2 and the definition of $\nu(z^t,u^t)$ we have that modeling error can be made arbitrarily small provided that (z^t,u^t) remain in a compact set (e.g. $\bar{y}$).

Theorem 5 *Consider the FDS ($R^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}}$) and suppose that system in Eq. (3.5) is its underlying system. Assume that either assumptions (**A.1**) or (**A.2**) hold. Also consider the RHONN in [21]. Then, for any $\varepsilon > 0$ there exists a matrix Θ^* and a number L^* high order connections and $\Theta = \Theta^*$ is a generator for the FDS described by the rules*

$$R^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}} \Leftrightarrow \left\{ \begin{array}{c} IF\ y_1\ is\ \tilde{y}_{1j_1}\ AND... \\ AND\ y_{n+m}\ is\ \tilde{y}_{(n+m)j_{n+m}} \\ THEN \\ \chi_1\ is\ \tilde{y}_{1l_1}\ AND...AND\ \chi_n\ is\ \tilde{y}_{nl_n} \\ with\ possibility\ \widehat{\pi}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}} \end{array} \right\}.$$

where

$$\max \left| \pi^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}} - \widehat{\pi}^{l_1,\ldots,l_n}_{j_1,\ldots,j_{n+m}} \right| < \varepsilon.$$

Chapter 6

Bilinear Neuro-Fuzzy Indirect Adaptive Control

6.1 Indirect Adaptive Neuro-Fuzzy Control

6.1.1 Neuro-Fuzzy Representation and Identification

At first we consider affine in the control, nonlinear ,in general, dynamical systems of the form

$$\dot{x} = f(x) + G(x) \cdot u \tag{6.1}$$

where the state $x \in R^n$ is assumed to be completely measured, the control signal u is in R^n, f is an unknown smooth vector field which is called the drift term and G ia a matrix with columns the unknown smooth controlled vector fields g_i, $i = 1, 2, ..., n$ and $G = [g_1, g_2, \ldots, g_n]$. The above class of continuous-time nonlinear systems are called affine, because in (6.1) the control input appears linear with respect to G. The main reason for considering this class of nonlinear systems is that most of the systems encountered in engineering, are by nature or technical design, affine. Furthermore, we note that non affine systems of the form given in (4.11) can be converted into affine, by passing the input through integrators, a procedure which is widely known as dynamic extension.

In our approach, referred to as indirect adaptive fuzzy-HONNF control, the parameters of the plant are estimated on-line except of the fuzzy partitions which are used to calculate the controller parameters. The basic structure of the indirect fuzzy-RHONN controller is shown in Fig. (6.1).

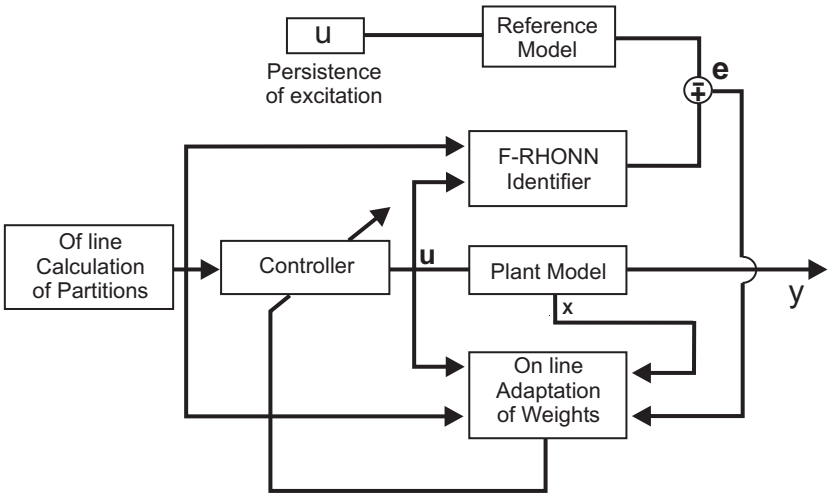

Figure 6.1: Overall scheme of the proposed indirect adaptive neuro-fuzzy control system.

The following mild assumptions are also imposed on (6.1), to guarantee the existence and the uniqueness of solution for any finite initial condition and $u \in U$.

Proposition: Given a class U of admissible inputs, then for any $u \in U$ and any finite initial condition, the state trajectories are uniformly bounded for any finite $T > 0$. Hence, $|x(T)| < \infty$.

Proposition: The vector fields f, g_i, $i = 1, 2, ..., n$ are continuous with respect to their arguments and satisfy a local Lipchitz condition so that the solution $x(t)$ of (6.1) is unique for any finite initial condition and $u \in U$.

We are using an affine in the control fuzzy dynamical system, which approximates the system in (6.1) and uses two fuzzy subsystem blocks for the description of $f(x)$ and $G(x)$ as follows:

$$f(\chi) = A\chi + \sum \bar{f}^{l_1,...,l_n}_{j_1,...,j_n} \times I^{l_1,...,l_n}_{j_1,...,j_n}(\chi) \tag{6.2}$$

$$g_i(\chi) = \sum (\bar{g}_i)^{l_1,...,l_n}_{j_1,...,j_n} \times I_1{}^{l_1,...,l_n}_{j_1,...,j_n}(\chi) \tag{6.3}$$

where the summation is carried out over the number of all available fuzzy rules, I, I_1 are appropriate fuzzy rule indicator functions and the meaning of indices $\bullet^{l_1,...,l_n}_{j_1,...,j_n}$ has already been described in Section 3.1.

According to Lemma for indicator HONNF's, every indicator function can be approximated with the help of a suitable HONNF. Therefore, every I, I_1 can be replaced with a corresponding HONNF as follows:

$$f(\chi) = A\chi + \sum \bar{f}^{l_1,...,l_n}_{j_1,...,j_n} \times N^{l_1,...,l_n}_{j_1,...,j_n}(\chi) \tag{6.4}$$

$$\bar{g}_i(\chi) = \sum (\bar{g}_i)^{l_1,...,l_n}_{j_1,...,j_n} \times N_1{}^{l_1,...,l_n}_{j_1,...,j_n}(\chi) \tag{6.5}$$

where N, N_1 are appropriate HONNFs.

In order to simplify the model structure, since some rules result to the same output partition, we could replace the NNs associated to the rules having the

same output with one NN and therefore the summations in (6.4),(6.5) are carried out over the number of the corresponding output partitions. Therefore, the affine in the control fuzzy dynamical system in (6.2), (6.3) is replaced by the following equivalent affine Recurrent High Order Neural Network (RHONN), which depends on the centers of the fuzzy output partitions $\bar{f}_l$ and $\bar{g}_{i,l}$

$$\dot{\hat{\chi}} = A\hat{\chi} + \sum_{l=1}^{Npf} \bar{f} \times N_l(\chi) + \sum_{i=1}^{n} \left(\sum_{l=1}^{Npg_i} (\bar{g}_i)_l \times N_{1l}(\chi) \right) u_i \tag{6.6}$$

Or in a more compact form

$$\dot{\hat{\chi}} = A\hat{\chi} + XWS(\chi) + X_1 W_1 S_1(\chi) u \tag{6.7}$$

Where A is a $n \times n$ stable matrix which for simplicity can be taken to be diagonal as $A = diag[a_1, a_2, ..., a_n]$, X, X_1 are matrices containing the centres of the partitions of every fuzzy output variable of $f(x)$ and $g(x)$ respectively, $S(\chi), S_1(\chi)$ are matrices containing high order combinations of sigmoid functions of the state χ and W, W_1 are matrices containing respective neural weights according to (6.6) and (5.4). The dimensions and the contents of all the above matrices are chosen so that $XWS(\chi)$ is a $n \times 1$ vector and $X_1 W_1 S_1(\chi)$ is a $n \times n$ matrix. Without compromising the generality of the model we assume that the vector fields in (6.3) are such that the matrix G is diagonal. For notational simplicity we assume that all output fuzzy variables are partitioned to the same number, m, of partitions. It should be noted that each output fuzzy variable may have a different number of let's say m_i partitions where

$$k = \sum_{i=1}^{n} m_i$$

Then the matrix X is of dimension $n \times k$ and is block diagonal. Without loss of generality (the same results are true for non-equal partition-numbers for each variable), X is a $n \times n \cdot m$ block diagonal matrix of the form $X = diag(X^1, X^2, \ldots, X^n)$ with each X^i being an m-dimensional raw vector of the form

$$X^i = \begin{bmatrix} \bar{f}_1^i & \bar{f}_2^i & \cdots & \bar{f}_m^i \end{bmatrix}$$

where $\bar{f}_p^i$ denotes the centre of the $p-th$ partition of f_i. Also, $S(\chi) = \begin{bmatrix} s_1(\chi) & \ldots & s_k(\chi) \end{bmatrix}^T$, where each $s_i(\chi)$ with $i = \{1, 2, ..., k\}$, is a high order combination of sigmoid functions of the state variables and W is a $n \cdot m \times k$ matrix with neural weights. W assumes the form $W = \begin{bmatrix} W^1 & \cdots & W^n \end{bmatrix}^T$, where each W^i is a matrix $\left[w_{j\,l}^i \right]_{m \times k}$. Also, X_1 is a $n \times n \cdot m$ block diagonal

matrix $X_1 = diag({}^1X^1, {}^1X^2, \ldots, {}^1X^n)$ with each ${}^1X^i$ being an m-dimensional raw vector of the form

$$ {}^1X^i = \begin{bmatrix} \bar{g}_1^{i,i} & \bar{g}_2^{i,i} & \cdots & \bar{g}_m^{i,i} \end{bmatrix}, $$

where $\bar{g}_k^{i\,i}$ denotes the center of the k-th partition of g_{ii}. W_1 is a $m \cdot n \times n$ block diagonal matrix $W_1 = diag({}^1W^1, {}^1W^2, \ldots, {}^1W^n)$, where each ${}^1W^i$ is a column vector $\left[{}^1w_{j\,l}^i \right]_{m \times 1}$ of neural weights. Finally, $S_1(\chi)$ is a $n \times n$ diagonal matrix with each diagonal element $s_i(\chi)$ being a high order combination of sigmoid functions of the state variables.

According to the above definitions the configuration of the F-HONNF approximator is shown in Fig. (6.2). When the inputs are given into the fuzzy-neural network shown in Fig. (6.2), the output of layer IV gives indicator function outputs which activate the corresponding rules and are calculated by Eq. (5.4). At layer V, each node performs a fuzzy rule while layer VI gives the function output.

The approximator of indicator functions, has four layers. At layer I, the input nodes represent input and/re state measurable variables. At layer II, the nodes represent the values of the sigmoidal functions. At layer III, the nodes are the values of high order sigmoidal combinations. The links between layer III and layer IV are fully connected by the weighting factors $W = \begin{bmatrix} W^1 & \cdots & W^n \end{bmatrix}^T$, the adjusted parameters. Finally, at layer IV the output represents the values of indicator functions.

6.1.2 Parametric and partition centers uncertainty

We assume the existence of uncertainty in the partition centers and parameter weight uncertainty, so, we can take into account that the actual system (6.1) can be modeled by the following neural form:

$$ \dot{\chi} = A\chi + X^* W^* S(\chi) + X_1^* W_1^* S_1(\chi) u \tag{6.8} $$

Define now, the error between the identifier states and the real states as

$$ e = \hat{\chi} - \chi \tag{6.9} $$

Then from (6.7) and (6.8) we obtain the error equation

$$ \dot{e} = Ae + X^* W^* S(\chi) + X_1^* W_1^* S_1(\chi) u $$

$-XWS(\chi) - X_1W_1S_1(\chi)u$

To this end add and subtract to the above error equation the terms $X^*WS(\chi)$ and $X_1^*W_1S(\chi))u$
and define: $\tilde{W} = W - W^*$ and $\tilde{W}_1 = W_1 - W_1^*$.
Then the error equation becomes:

$$\dot{e} = Ae - X^*\tilde{W}S(\chi) - \tilde{X}WS(\chi) - X_1^*\tilde{W}_1S_1(\chi)u - \tilde{X}_1W_1S_1(\chi)u \tag{6.10}$$

Our objective is to find suitable control and learning laws to drive both e and χ to zero, while all other signals in the closed loop remain bounded. Taking u to be equal to

$$u = -\left[X_1W_1S_1(\chi)\right]^{-1} XWS(\chi) \tag{6.11}$$

and substituting it into (7.4) we finally obtain

$$\dot{\hat{\chi}} = A\hat{\chi} \tag{6.12}$$

In the next theorem the weight update laws are given, which can serve identification and control objectives, provided the updating of the weights of matrices X_1 and W_1 is performed in such a way, so that the existence of $\left[X_1W_1S_1(\chi)\right]^{-1}$ is assured.

Theorem 6 *Consider the identification scheme given by (6.10). Provided that* $\left[X_1W_1S_1(\chi)\right]^{-1}$ *exists the learning laws:*
a) For the elements of W *and* X

$$\begin{cases} \dot{W} = sgn(X^*)^T PeS^T \\ \dot{X} = PeS^TW^T \end{cases} \tag{6.13}$$

b) For the elements of W_1 *and* X_1

$$\begin{cases} \dot{W}_1 = sgn(X_1^*)^T Peu^T S_1^T \\ \dot{X}_1 = Peu^T S_1^T W_1^T \end{cases} \tag{6.14}$$

guarantee the following properties.

- $e, \hat{\chi}, \tilde{W}, \tilde{W}_1 \in L_\infty, \quad e, \hat{\chi} \in L_2$
- $\lim_{t\to\infty} e(t) = 0, \quad \lim_{t\to\infty} \hat{\chi}(t) = 0$

- $\lim_{t\to\infty} \dot{\tilde{W}}(t) = 0, \quad \lim_{t\to\infty} \dot{\tilde{W}}_1(t) = 0$

where the matrices $sgn(X^*)$ and $sgn(X_1^*)$ are defined in the proof.

The proof can be studied in the Appendix.

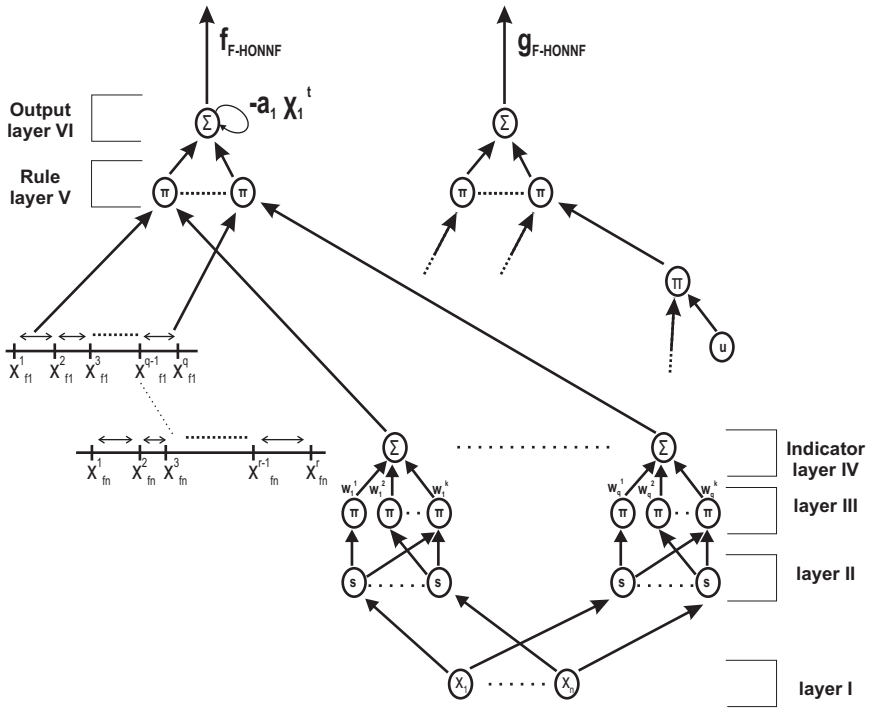

Figure 6.2: Overall scheme of the proposed indirect adaptive neuro-fuzzy control system.

6.1.3 Introduction to the parameter hopping

The weight updating laws presented previously in Section 6.1.2 are valid when the control law signal in (6.11) exists. Therefore, the existence of $[X_1 W_1 S_1(\chi)]^{-1}$ has to be assured. Since $S_1(\chi)$ is diagonal with its elements $s_i(\chi) \neq 0$ and X_1, W_1 are block diagonal the existence of the inverse is assured when ${}^1X^i \cdot {}^1W^i \neq 0, \forall i = 1, \dots n$. Therefore, W_1 has to be confined such that $\left|{}^1X^i \cdot {}^1W^i\right| \geq \theta_i > 0$, with θ_i being a design parameter. In case the boundary defined by the above confinement is nonlinear the updating W_1 can be modified by using a projection algorithm [54]. However, in our case the boundary surface is linear and the direction of updating is normal to it because $\nabla \left[{}^1X^i \cdot {}^1W^i\right] = {}^1X^i$. Therefore, the projection of the updating vector on the boundary surface is of no use. Instead, using concepts from multidimensional vector geometry we modify the updating law such that, when the weight vector approaches (within a safe distance θ_i) the forbidden hyper-plane ${}^1X^i \cdot {}^1W^i = 0$ and the direction of updating is toward the forbidden hyper-plane, it introduces a *hopping* which drives the weights in the direction of the updating but on the other side of the space, where here the weight space is divided into two sides by the forbidden hyper-plane. This procedure is depicted in Fig. 6.3, where a simplified 2-dimensional representation is given. Theorem 7 below introduces this *hopping* in the updating law.

Theorem 7 *Consider the control scheme (6.10), (6.11), (6.12). The updating law:*

a) For the elements of W^i *given by (6.13)*

b) For the elements of ${}^1W^i$ *given by the modified form:*

$$
\begin{aligned}
&{}^1\dot{W}^i = -({}^1X^i)^T p_i e_i u_i s_i(\chi) \quad if \ \left|{}^1X^i \cdot {}^1W^i\right| > \theta_i > 0 \\
&or \quad \left|{}^1X^i \cdot {}^1W^i\right| = \theta_i \ and \quad {}^1X^i \cdot {}^1\dot{W}^i \leq 0
\end{aligned}
$$

$$
{}^1\dot{W}^i = -({}^1X^i)^T p_i e_i u_i s_i(\chi) - \\
- \frac{2}{tr\{({}^1X^i)^T \, {}^1X^i\}} \, {}^1X^i \, {}^1W^i \, ({}^1X^i)^T \quad otherwise
$$

guarantees the properties of theorem 6 and assures the existence of the control signal.

Vectorial proof of parameter hopping

In selecting the terms involved in parameter hopping we start from the vector definition of a line, of a plane and the distance of a point to a plane. The equation of a line in vector form is given by $r = a + \lambda t$, where a is the position vector of a given point of the line, t is a vector in the direction of the line and λ is a real

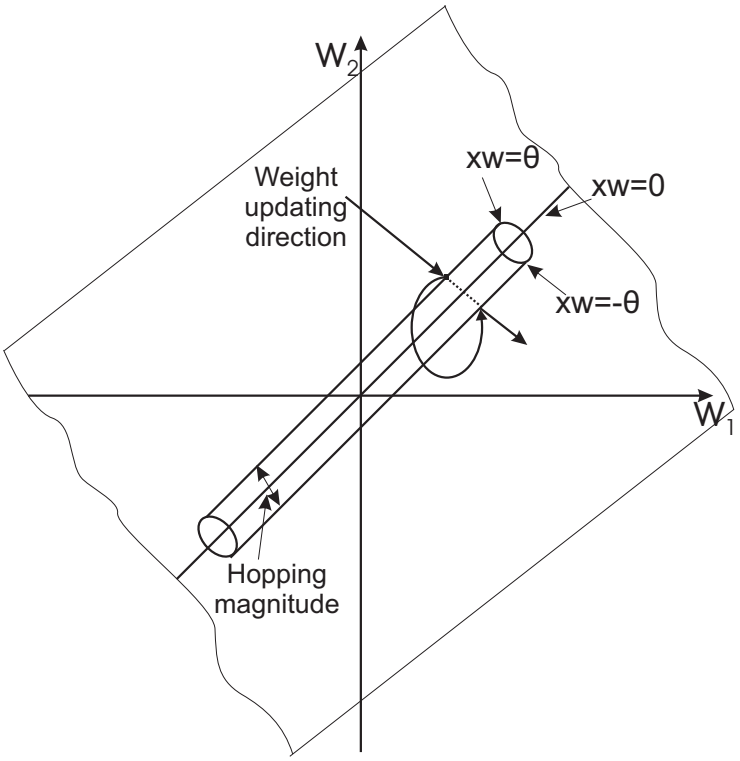

Figure 6.3: Pictorial Representation of parameter hopping)

scalar. By giving different numbers to λ we get different points of the line each one represented by the corresponding position vector r. The vector equation of a plane can be defined by using one point of the plane and a vector normal to it. In this case $r \cdot n = a \cdot n = d$ is the equation of the plane, where a is the position vector of a given point on the plane, n is a vector normal to the plane and d is a scalar. When the plane passes through zero, then apparently $d = 0$. To determine the distance of a point B with position vector b from a given plane we consider Fig. 6.4 and combine the above definitions as follows. Line BN is perpendicular to the plane and is described by vector equation $r = b + \lambda n$, where n is the normal to the plane vector. However, point N also lies on the plane and in case the plane passes through zero

$$r \cdot n = 0 \Rightarrow (b + \lambda n) \cdot n = 0 \Rightarrow \lambda = \frac{-b \cdot n}{\|n\|}.$$

Apparently, if one wants to get the position vector of B′ (the symmetrical of B in respect to the plane), this is given by

$$r = b - 2\frac{b \cdot n}{\|n\|} n.$$

In our problem $b = {}^1W^i$, our plane is described by the equation ${}^1X^i \cdot {}^1W^i = 0$ and as it has already been mentioned the normal to it is the vector ${}^1X^i$.

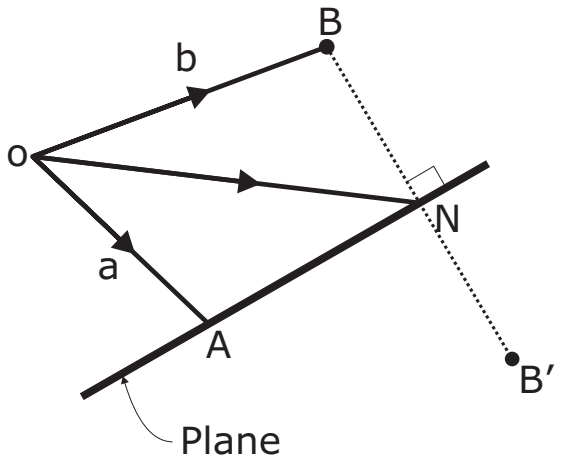

Figure 6.4: Vector explanation of parameter hopping)

Chapter 7

Simulation and Results

7.1 Simulation of a DC Motor

In this section we apply the proposed approach to solve the problem of controlling the speed of a 1 KW DC motor with a normalized model described by the following dynamical equations

$$\begin{array}{c} T_a \frac{dI_a}{dt} = -I_a - \Phi\Omega + V_a \\ T_m \frac{d\Omega}{dt} = \Phi I_a - K_0\Omega - m_L \\ T_f \frac{d\Phi}{dt} = -I_f + V_f \\ \Phi = \frac{aI_f}{1+bI_f} \end{array} \tag{7.1}$$

(7.2)

Traditionally, the angular velocity of a DC motor is controlled with changes in its armature voltage, while keeping constant the field excitation. Thus, the aboe nonlinear model is linearized and reduced to

$$\begin{array}{c} T_a \frac{dI_a}{dt} = -I_a - \Phi\Omega + V \\ \\ T_m \frac{d\Omega}{dt} = \Phi I_a - K_0\Omega - m_L \end{array} \tag{7.3}$$

now with Ω a constant value parameter.

So, the regulation problem of a DC motor is translated as follows: Find a state feedback to force the angular velocity Ω and the armature current I_a to go to zero, while the magnetic flux varies.

To achieve such a goal, assuming that the dynamics of the system are unknown, we first assume that the system is described, within a degree of accuracy, by a neuro-fuzzy system of the form

$$\dot{\hat{\chi}} = A\hat{\chi} + XWS(\chi) + X_1W_1S_1(\chi)u \tag{7.4}$$

Where A is a $n \times n$ stable matrix which for simplicity can be taken to be diagonal as $A = diag[a_1, a_2, ..., a_n]$, X, X_1 are matrices containing the centres

Table 7.1: Parameter values for the DC motor.

Parameter	Value
$1/T_a$	148.88
$1/T_m$	42.91
K_0/T_m	0.0129
T_f	31.88
T_L	0.0
a	2.6
β	1.6

of the partitions of every fuzzy output variable of $f(x)$ and $g(x)$ respectively, $S(\chi), S_1(\chi)$ are matrices containing high order combinations of sigmoid functions of the state χ and W, W_1 are matrices containing respective neural weights.

The number of states being $n = 2$, the number of fuzzy partitions being $m = 5$ and the depth of high order sigmoid terms $k = 11$. In this case $s_i(x)$ assumes high order connection up to the second order.

Also, to regulate the motor speed to zero we apply the control law

$$u = -\left[X_1 W_1 S_1(\chi)\right]^{-1} XWS(\chi) \tag{7.5}$$

where the number of fuzzy partitions of each $g_{i\,i}$ is $m = 3$.

We simulated a 1 KW DC motor with parameter values that can be seen in Table 7.1. Our two stage algorithm, was applied.

We considered the identification procedure known and for the indirect adaptive control we used the following values

For the block diagonal matrix $X = diag(X^1, X^2, \cdots, X^n)$ with $n = 2$ and the sub-matrices X^1, X^2, we gave the following values for the centers of the fuzzy partitions:

$$X = \begin{bmatrix} X^1 & 0 \\ 0 & X^2 \end{bmatrix}$$

where the centers of the fuzzy partitions of X^1, X^2 are the following:

$$X^1 = \begin{bmatrix} -163.3061 & -148.9226 & -153.1720 & -79.9453 & -175.4806 & -18.1483 \end{bmatrix}$$

$$X^2 = \begin{bmatrix} 25.1305 & 9.1845 & -15.2403 & 18.0842 & -9.2918 & -0.1840 \end{bmatrix}$$

For the block diagonal matrix $X_1 = diag({}^1X^1, {}^1X^2, \ldots, {}^1X^n)$ with $n = 2$ and the sub-matrices ${}^1X^1, {}^1X^2$, we gave the following values for the centers of the fuzzy partitions:

$$X_1 = \begin{bmatrix} {}^1X^1 & 0 \\ 0 & {}^1X^2 \end{bmatrix}$$

where the centers of the fuzzy partitions of X^1, X^2 are the following:

$${}^1X^1 = \begin{bmatrix} 148 & 149 & 150 \end{bmatrix}$$

$${}^1X^2 = \begin{bmatrix} 42 & 43 & 44 \end{bmatrix}$$

In the sequel, we give the values of the matrices (W, W_1) which contain the neural weights

For the block diagonal (2*6x2) matrix W, the initial values are

$$W = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

And for the block diagonal (2*3x2) matrix W_1, the initial values

$$W = \begin{bmatrix} 0.04 & 0 \\ 0.04 & 0 \\ 0.04 & 0 \\ 0 & 0.04 \\ 0 & 0.04 \\ 0 & 0.04 \end{bmatrix}$$

In order to estimate our actual system-model in the proposed neural form, we need to calculate the function sigmoidals, i.e., the matrices $S(\chi), S_1(\chi)$. For that reason, we used the following values for their parameters a, b, c:

$a_1 = 0.1;\ \ b_1 = 1;\ \ c_1 = 0$ sigmoidal parameters used in w update.

$a_2 = 6;\ \ b_2 = 1;\ \ c_2 = 0$ sigmoidal parameters used in w_1 update.

Last, we need to find an appropriate matrix $A > 0$ which is chosen in order to satisfy the Lyapunov equation

$$PA + A^T P = -I,$$

and a matrix A we found in order to achieve the satisfaction of the above equation is

$$A = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

This matrix P, which is calculated by the Lyapunov equation is used in the update laws for $W, X, W1, X1$.

The figures 7.1,7.2,7.3 give the evolution of the states of the DC motor,i.e., the armature current I_a, the angular velocity Ω and the magnetic flux Φ respectively.We used the initial values $\Omega = 0.3$ and $I_a = 0.3$ for the angular velocity and the armature current respectively.

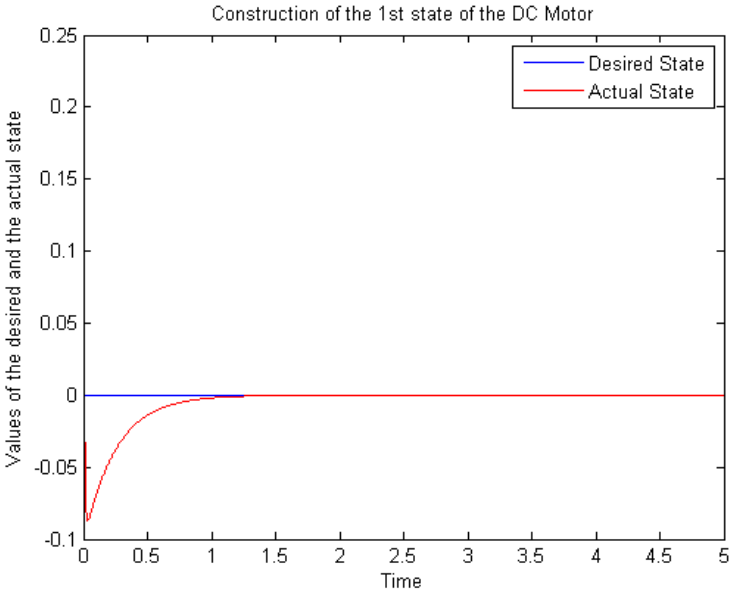

Figure 7.1: Evolution of the armature current of the DC motor system .

As can be seen,both Ω and I_a converge to zero very fast as desired.

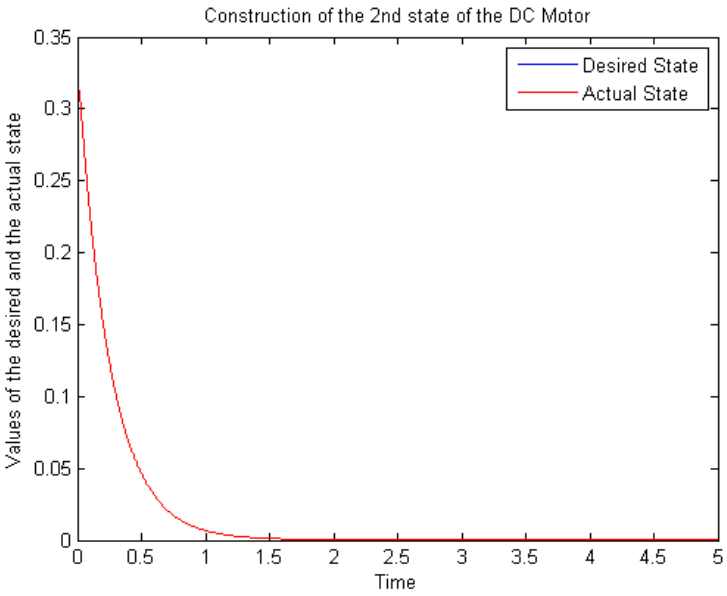

Figure 7.2: Evolution of the angular velocity of the DC motor system .

As we can see below the magnetic flux remains bounded,as desired.

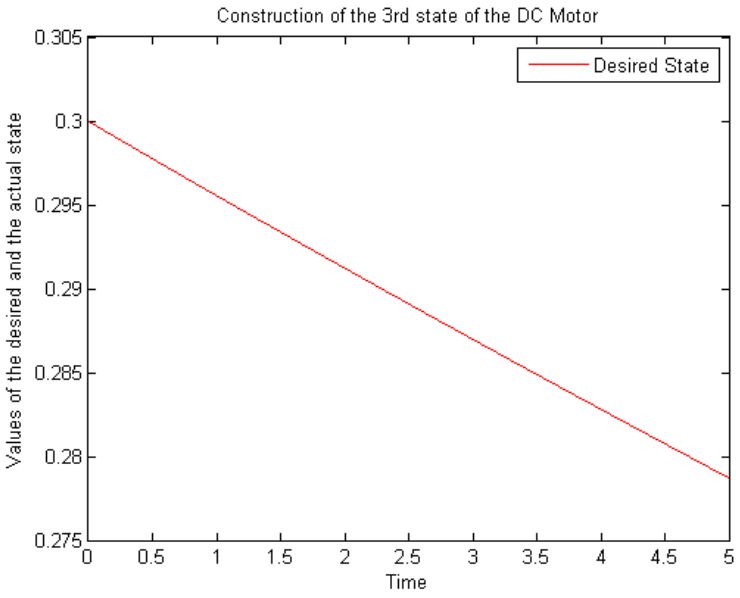

Figure 7.3: Evolution of the magnetic flux of the DC motor system .

In the sequel, we give the evolution of the control signal u, which is calculated by the controller stage.

As can be seen below, the $2-$ *state* control input remains bounded and converges to zero as time evolves.

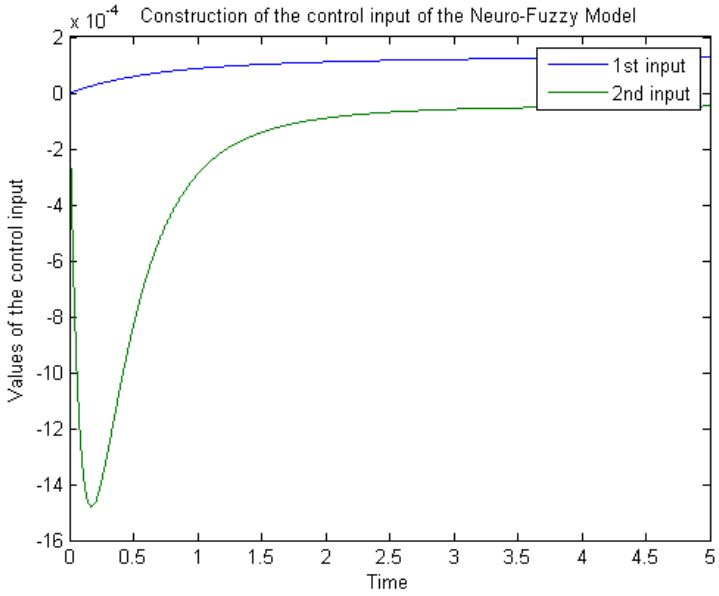

Figure 7.4: Evolution of the control signal of the proposed scheme .

Last, we give the evolution of the error between the F-RHONN approximator and the actual system.

As can be seen below, the error between the F-RHONN approximator and the actual system converges to zero as desired.

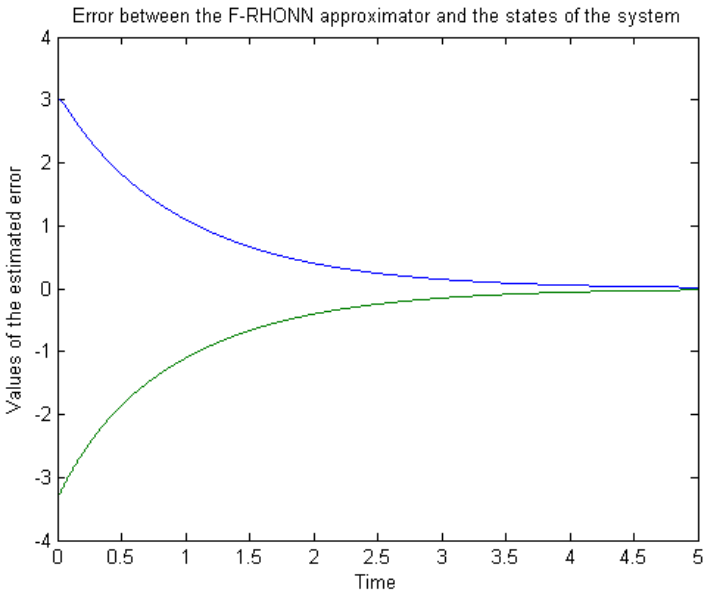

Figure 7.5: Evolution of the error signal between the F-HONNF approximator and the actual system .

In the sequel we present the figures 7.6,7.7,7.8 for the same variables, but now we put initial values for $I_a, \Omega = 0.1$.

The three states of the DC motor system are the following

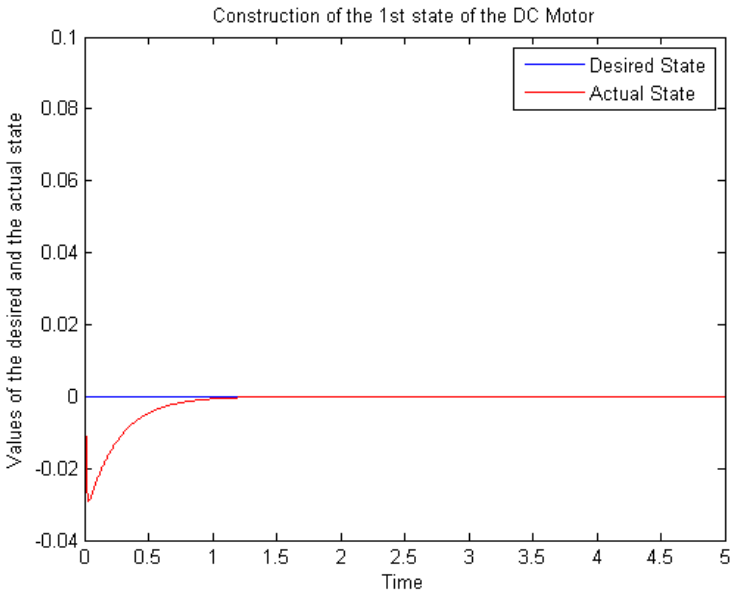

Figure 7.6: Evolution of the armature current of the DC motor system .

As can be seen,both Ω and I_a converge to zero very fast as desired despite the change of the initial values.

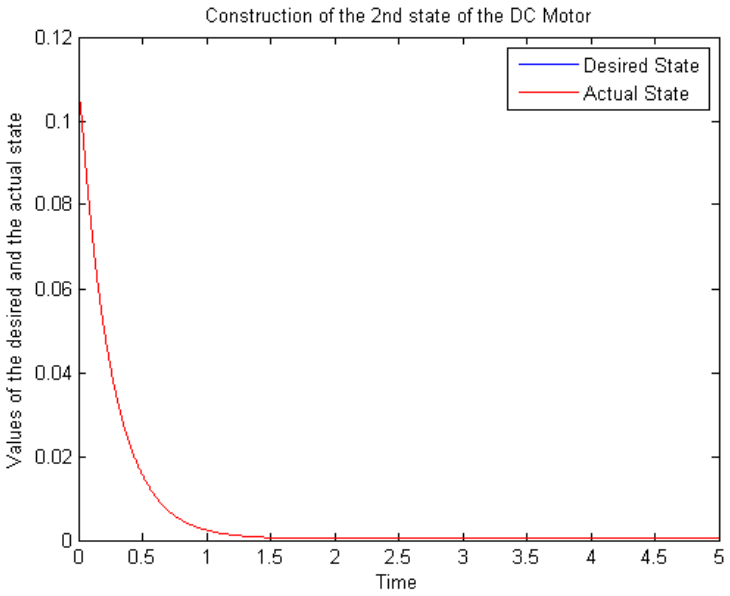

Figure 7.7: Evolution of the angular velocity of the DC motor system .

As we can see below the magnetic flux remains bounded as well,as desired.

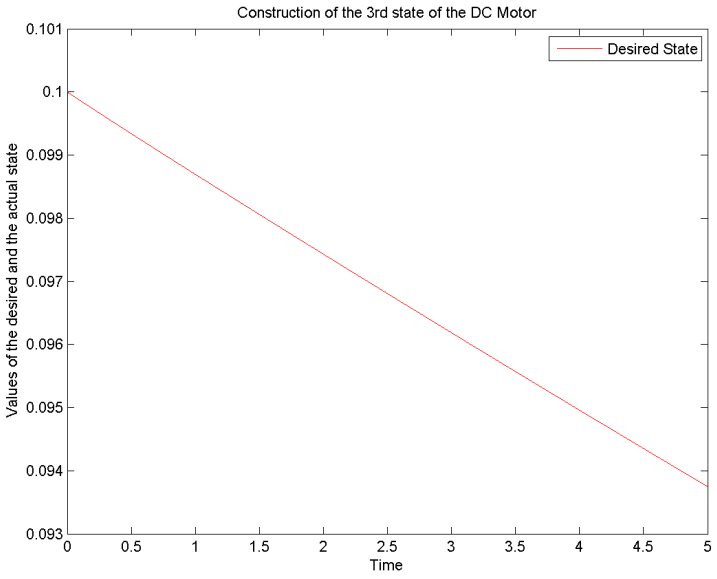

Figure 7.8: Evolution of the magnetic flux of the DC motor system .

In the sequel, we give the evolution of the control signal u, which is calculated by the controller stage.

As can be seen below, the $2-state$ control input remains bounded and converges to zero as time evolves.

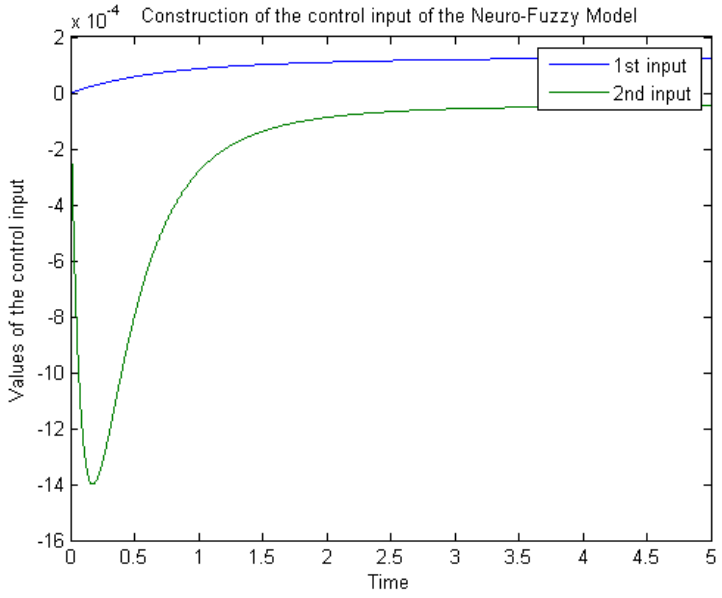

Figure 7.9: Evolution of the control signal of the proposed scheme .

Last, we give the evolution of the error between the F-RHONN approximator and the actual system.

As can be seen below, the error between the F-RHONN approximator and the actual system converges to zero as desired.

We proved the stability the stability of the proposed scheme for different values of the initial parameters which guarantees the viable results of the mentioned theory.

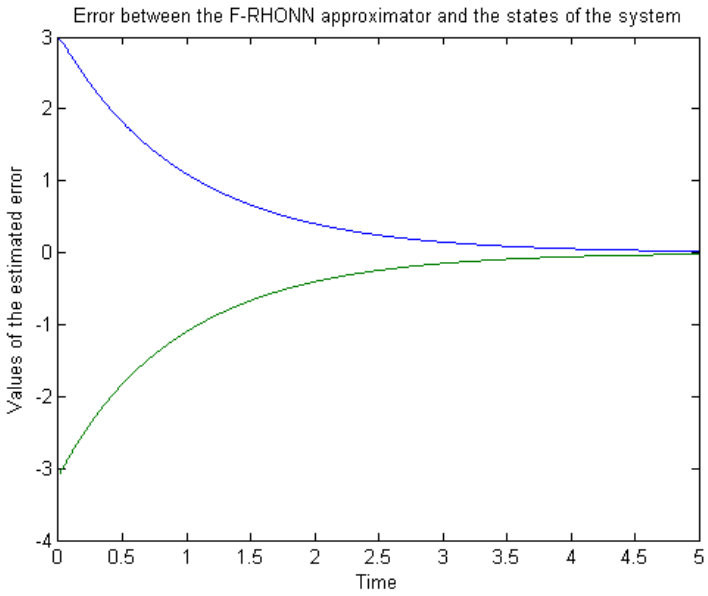

Figure 7.10: Evolution of the error signal between the F-HONNF approximator and the actual system .

Chapter 8

Conclusions Chapter

8.1 Conclusions

In this diploma thesis we considered an indirect adaptive control scheme in order to regulate unknown nonlinear plants. This approach is based on a new Neuro-Fuzzy Dynamical Systems definition, which uses the concept of Fuzzy Dynamical Systems (FDS) operating in conjunction with High Order Neural Network Functions (HONNFs). Since the plant is initially considered unknown, we first propose its approximation by a special form of an affine in the control fuzzy dynamical system (FDS) and in the sequel the fuzzy rules are approximated by appropriate HONNFs. Once the system is identified around an operation point is regulated to zero adaptively. The proposed scheme does not require a-priori experts' information on the number and type of input variable membership functions making it less vulnerable to initial design assumptions. Weight updating laws for the involved HONNFs are provided, which guarantee that both the identification error and the system states reach zero exponentially fast, while keeping all signals in the closed loop bounded. A method of parameter hopping assures the existence of the control signal and is incorporated in the weight updating law. Simulations illustrate the potency of the method by comparing its performance with this of conventional approaches. More specifically, the applicability of the method was tested on a DC Motor system where it is shown that by following the proposed procedure one can obtain asymptotic regulation.

Chapter 9

Bibliography-References

Bibliography

[1] K. Hornic, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", *Neural Networks*, vol. 2, pp. 359-366, 1989.

[2] L. Wang, *Adaptive Fuzzy Systems and Control*, Prentice Hall, Englewood Cliffs, NJ, 1994.

[3] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system", *IEEE Trans. Syst. Man. Cyber.*, vol. 23, pp. 665-684, 1993.

[4] C.T. Lin, "A neural fuzzy control system with structure and parameter learning", *Fuzzy Sets and Systems*, vol. 70, pp. 183-212, 1995.

[5] K.B. Cho and B.H.Wang, "Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction", *Fuzzy Sets and Systems*, vol. 83, pp. 325-339, 1996.

[6] C.F.Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications", *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 12-32, 1998.

[7] R. P. Li and M. Mukaidono, "A new approach to rule learning based on fusion of fuzzy logic and neural networks", *IEICE Trans. Fuzzy Syst*, vol. E78-d, pp. 1509-1514, 1995.

[8] H.L. Hiew, C.P. Tsang, "Fuzzy Chaos and recursive partitioning", in B.Kosko(Ed.), Fuzzy Engineering, Prentice-Hall International Inc. new Jersey, 1997.

[9] S. L. Chiu, "Fuzzy model identification based on cluster estimation", *Journal of Intelligent and Fuzzy Systems*, vol. 2, 1994.

[10] Y. H. Lin and G. A. Cunningham, "A new approach to fuzzy-neural system modelling", *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 190-197, 1995.

[11] C. F. Jang and C. T. Lin, "An online self-constructing neural fuzzy inference network and its applications", *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 12-32, 1998.

[12] S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: survey in soft computing framework", *IEEE Trans. On Neural Networks*, vol. 11, pp. 748-768, 2000.

[13] B. S. Chen, C. H. Lee, and Y. C. Chang, "Tracking design of uncertain nonlinear siso systems: Adaptive fuzzy approach", *IEEE Trans. Fuzzy Syst.*, vol. 4, pp. 32-43, Feb. 1996.

[14] J. T. Spooner and K. M. Passino, "Stable adaptive control using fuzzy systems and neural networks", *IEEE Trans. Fuzzy Syst.*, vol. 4, pp. 339-359, June 1996.

[15] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks", *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27, Feb.1990.

[16] M. M. Polycarpou and M. J. Mears, "Stable adaptive tracking of uncertain systems using nonlinearly parameterized online approximators", *Int. J. Control*, vol. 70, no. 3, pp. 363-384, 1998.

[17] G. A. Rovithakis and M. A. Christodoulou, "Adaptive control of unknown plants using dynamical neural networks", *IEEE Trans. Syst, Man, Cybern.*, vol. 24, , pp. 400-412 Mar. 1994.

[18] F.C. Chen and C.C. Liu, "Adaptively controlling nonlinear continuous-time systems using multilayer neural networks", *IEEE Trans. Automat. Control*, vol.39, June 1994, pp.1306-1310.

[19] N. Golea, A. Golea, K. Benmahammed, "Stable indirect fuzzy adaptive control", *Fuzzy Sets and Systems*, vol. 137, pp. 353-366, 2003.

[20] E. B. Kosmatopoulos and M. A. Christodoulou, "Recurrent neural networks for approximation of fuzzy dynamical systems", *Int. Journal of Intelligent Control and Systems*, vol. 1, pp. 223-233, 1996.

[21] M. A. Christodoulou, D. C. Theodoridis, and Y. S. Boutalis, "Building Optimal Fuzzy Dynamical Systems Description Based on Recurrent Neural Network Approximation", *in Proc. Int. Conf. of Networked Distributed Systems for Intelligent Sensing and Control*, Kalamata, Greece, June, 2007.

[22] W. Yu, X. Li, "Fuzzy neural identification by online clustering with application on crude oil blending", *Int. Conf. on Fuzzy Systems*, Vancouver, BC, Canada, July 16-21, 2006.

[23] Cybenko G.,(1989) "Approximations by superpositions of a sigmoidal function", *Mathematics of Control Signals and Systems*, vol.2, pp. 303-314.

[24] Funahashi K.,(1989) "On the approximate realization of continuous mappings by neural networks",*Neural Networks*, vol.2, pp. 183-192.

[25] Rumelhart D.,Hinton D.,Williams G.,(1986) "Learning internal respectations by error propagation", in *Parallel Distributed Processing*, D.Rumelhart and F.McClelland, Eds, Vol.1, Cambridge, MA, MIT Press.

[26] Pineda F.J., (1988) "Generalization of backpropagation to recurrent networks", *Phys. Rev. Lett.*, vol.59, no.19, pp.2229-2232.

[27] Narendra K.S., Parthasarathy K., (1991) "Gradient methods for the optimization of dynamical systems containing neural networks", *IEEE Transactions on Neural Networks*, vol.2, no.2, pp.252-262.

[28] Narendra K.S., Parthasarathy K., (1990) "Identification and Control of Dynamical Systems using Neural Networks", *IEEE Transactions on Neural Networks*, vol.1, no.1, pp.4-27.

[29] Werbos P.J.,(1990) "Backpropagation through time: what it does and how we do it", *Proc. IEEE*, vol.78, pp.1550-1560.

[30] Williams R.J., Zipser D., (1989) "A learning algorithm for continually running fully recurrent neural networks", *Neural Computation*, vol.1, pp.270-280.

[31] Polycarpou M.M.,Ioannou P.A.,(1991) "Identification and Control of Nonlinear Systems using neural networks models: design and stability analysis", *Tech. Rep. 91-09-01, Univ. of Southern Cal., Los Angeles*

[32] Sanner R.M., Slotine J.-J.E., (1992) "Gaussian neural networks for direct adaptive control", *IEEE Trans. Neural Networks*, vol.3, no.6., pp. 837-863.

[33] Chen F.-C., Liu C.-C., (1994) "Adaptively controlling nonlinear continuous-time systems using multi-layer neural networks",*IEEE Trans. Automat. Control*, vol.39, no.6, pp. 1306-1310.

[34] Chen F.-C., Khalil H.K., (1995) "Adaptive control of a class of nonlinear discrete time systems using neural networks",*IEEE Trans. Automat. Control*, vol.40, no.5, pp. 791-801.

[35] Lewis F.L.,Liu K.,Yesildirek A., (1995) "Neural net robot controller with quaranteed tracking performance", *IEEE Trans. Neural Natworks*, vol.6, no.3, pp.703-715.

[36] Sadegh N.,(1991) "Nonlinear identification and control via neural networks", *Control Systems with Inexact Dynamic Models*, DSC-vol. 33, ASME Winter Annual Meeting.

[37] Sadegh N.,(1993) "A perceptron network for functional identification using radial gaussian networks", *IEEE Trans. Neural Networks*, vol.4, no.6, pp.982-988.

[38] Rovithakis G.A., Christodoulou M.A., (1994) "Adaptive Control of Unknown Plants using Dynamical Neural Networks", *IEEE Trans. Systems Man Cybernetics*, vol.24, no.3, pp.400-412.

[39] Kosmatopoulos E.B., Polycarpou M., Christodoulou M.A., Ioannou P.A., (1995) "High-order neural networks structures for identification of dynamical systems", *IEEE Trans. Neural Networks*, vol.6, no.2, pp.422-431.

[40] Hopfield J.J., (1984) "Neurons with graded response have collective computational properties like those of two-state neurons", *Proc.Natl.Acad.Sci.*, vol.81, pp. 3088-3092.

[41] Cohen M.A.,Grossberg S., (1983) "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks", *IEEE Trans. on Systems, Man., and Cybernetics*, vol. SMC-13, pp. 815-826.

[42] Paretto P., Niez J.J., (1986) "Long term memory storage capacity of multiconnected neural networks", *Biol.Cybern.*, vol.54, pp.53-63.

[43] Baldi P., (1988) "Neural networks, orientations of the hypercube and algebraic threshold functions", *IEEE Transactions on Information Theory*, vol. IT-34, pp.523-530.

[44] Dempo A.,Faroimi O., Kailath T., (1991) "High-order absolutely stable neural networks", *IEEE Transactions on Ciruits and Systems*, vol.38, no.1.

[45] Kamp Y.,Hasler M., (1990) "Recursive Neural Networks for Associative Memory", J.Wiley and Sons.

[46] Giles C.L., Chen D., Miller C.B., Chen H.H., Sun G.Z., and Lee Y.C., (1991) "Second-order recurrent neural networks for grammatical inference", *Proc. of Inter. Joint Conf. on Neural Networks*, IJCNN91, vol.2, pp.273-281.

[47] Liou R.,Azimi-Sadjadi M.R., Dent R., (1991) "Detection of dim targets in high cluttered backround using high order correlation neural networks", *Proc. of Iner. Joint Conf. on Neural Networks*, IJCNN91, vol.1, pp.701-706.

[48] Cotter N.E., (1990) "The Stone-Weierstrass theorem and its application to neural network", *IEEE Trans. on Neural Networks*, vol.1, no.4, pp. 290-295.

[49] Goodwin G.C., Sin K.S., (1984) "Adaptive filtering, Prediction and Control", Prentice Hall, New Jersey.

[50] Hale J.K., (1969) "Ordinary Differential Equations", New York, NY, Wiley-InterScience.

[51] Ioannou P.A., Datta A., (1991) "Robust Adaptive Control: a unified approach", *Proceedings of the IEEE*, December.

[52] Narendra K.S., Annaswamy A.M., (1989) "Stable Adaptive Systems", Englewood Cliffs, NJ, Prentice-Hall.

[53] Polycarpou M.M., Ioannou P.A., (1994) "On the existence and uniqueness of solutions in adaptive control systems", *IEEE Trans. on Automatic Control*, vol. 30, no.3, pp.474-479.

[54] P.Ioannou and B.Fidan, "Adaptive Control Tutorial", SIAM:Advances in Design and Control Series, 2006.

Chapter 10

Appendix

10.1 Proofs of Theorems

In this section we present the proofs of the theorems and lemmas we used in the previous sections of this diploma thesis.

Theorem 8 *Consider the system:*

$$\dot{\chi} = F(\chi, u), \tag{10.1}$$

where $\chi \in R^n$ is the system state, $u \in R^n$ is the system input and $F : R^{n+m} \rightarrow R^n$ is a smooth vector field defined on a compact set $\mathrm{y} \subset R^{n+m}$

and the model

$$\dot{x} = Ax + W^T z, \tag{10.2}$$

where $x = [x_1, x_2, \cdots, x_n]^T \in R^n, W = [w_1, w_2, \cdots, w_n]^T \in R^{Lxn}$ and $A = diag\,[-a_1, -a_2, \cdots, -a_n]$ is a nxn diagonal matrix. Since $[a_i > 0, i = 1, 2, \cdots, n]$, A is a stability matrix. Although it is not written explicitly, the vector z is a function of both the neural network state x and the external input u.

Suppose that the system (10.1) and the model (10.2) are initially at the same state $x(0) = \chi(0)$; then for any $\epsilon > 0$ and any finite $T > 0$, there exists an integer L and a matrix $W^* \in R^{Lxn}$ such that the state $x(t)$ of the RHONN model (10.2) with L high-order connections and weight values $W = W^*$ satisfies

$$\sup_{0 \leq t \leq T} |x(t) - \chi(t)| \leq \epsilon.$$

Proof 1 *From (10.2), the dynamic behavior of the RHONN model is described by*

$$\dot{x} = Ax + W^T z(x, u). \tag{10.3}$$

Adding and subtracting $A\chi$, (10.1) is rewritten as

$$\dot{\chi} = A\chi + G(\chi, u), \tag{10.4}$$

where $G(x, u) = F(x, u) - A\chi$. Since $x(0) = \chi(0)$, the state error $e = x - \chi$ satisfies the differential equation

$$\dot{e} = Ae + W^T z(x,u) - G(\chi,u), \;\; e(0) = 0. \tag{10.5}$$

By assumption, $(\chi(t), u(t)) \in \mathrm{y}$ *for all* $t \in [0,T]$*, where* y *is a compact subset of* R^{n+m}*. Let*

$$\mathrm{y}_e = \left\{(\chi,u) \in R^{n+m} : |(\chi,u) - (\chi_y,u_y)| \leq \epsilon, \;\; (\chi_y,u_y) \in \mathrm{y}\right\}. \tag{10.6}$$

It can be seen easily that y_e *is also a compact subset of* R^{n+m} *and* $\mathrm{y} \subset \mathrm{y}_e$*. In simple words* y_e *is* ϵ *larger than* y*, where* ϵ *is the required degree of approximation. Since,* z *is a continuous function, it satisfies a Lipschitz condition in* y_e*, i.e., there exists a constant* l *such that for all* $(x_1,u),(x_2,u)$ *in*y_e

$$|z(x_1,u) - z(x_2,u)| \leq l|x_1 - x_2|. \tag{10.7}$$

In what follows, we show that the function $W^T z(x,u)$ *satisfies the conditions of the Stone-Weierstrass Theorem and can approximate any continuous function over a compact domain, therefore.*

From (4.2),(4.3) it is clear that $z(x,u)$ *is in the standard polynomial expansion with the exception that each component of the vector* x *is preprocessed by a sigmoid function* $s(\cdot)$*. As shown in [14], preprocessing of input via a continuous invertible function does not affect the ability of a network to approximate continuous functions; therefore, it can be shown readily that if* L *is sufficiently large, then there exist weight values* $W = W^*$ *such that* $W^*Tz(x,u)$ *can approximate* $G(x,u)$ *to any degree of accuracy, for all* (x,u) *in a compact domain.Hence, there exists* $W = W^*$ *such that*

$$\sup_{(\chi,u)\in \mathrm{y}_e} |W^*Tz(\chi,u) - G(\chi,u)| \leq \delta, \tag{10.8}$$

where δ *is a constant to be designed in the sequel.*

The solution of (10.5) is

$$e(t) = \int_0^t e^{A(t-\tau)} \left[W^*Tz(x(\tau),u(\tau)) - G(\chi(\tau),u(\tau))\right] d\tau =$$

$$\int_0^t e^{A(t-\tau)} \left[W^*Tz(x(\tau),u(\tau)) - W^*Tz(\chi(\tau),u(\tau))\right] d\tau +$$

$$\int_0^t e^{A(t-\tau)} \left[W^*Tz(\chi(\tau),u(\tau)) - G(\chi(\tau),u(\tau))\right] d\tau.$$

Since A is a diagonal stability matrix, there exists a positive constant α such that $\|e^{At}\| \leq e^{-\alpha t}$ for all $t \geq 0$. Also, let $L = l\|W^*\|$.Based on the aforementioned definitions of the constants α, L, ϵ, let δ in (10.8) be chosen as:

$$\delta = \frac{\epsilon\alpha}{2} e^{\frac{-L}{\alpha}} > 0. \tag{10.9}$$

First consider the case where $(x(t), u(t)) \in \mathrm{y}_e$ for all $t \in [0, T]$. Starting from the equation of the solution of (10.5), taking the norms on both sides and using (10.7),(10.8) and (10.9), the following inequalities hold for all $t \in [0, T]$:

$$\begin{aligned}
|e(t)| \leq \int_0^t \|e^{A(t-\tau)}\| \left|W^{*T} z(x(\tau), u(\tau)) - W^{*T} z(\chi(\tau), u(\tau))\right| d\tau \\
+ \int_0^t \|e^{A(t-\tau)}\| \left|W^{*T} z(\chi(\tau), u(\tau)) - G(\chi(\tau), u(\tau))\right| d\tau, \\
\leq \int_0^t e^{-\alpha(t-\tau)} L|e(\tau)| d\tau + \int_0^t \delta e^{-\alpha(t-\tau)} d\tau, \\
\leq \tfrac{\epsilon}{2} e^{-\frac{L}{\alpha}} + L \int_0^t e^{-\alpha(t-\tau)|e(\tau)|} d\tau.
\end{aligned}$$

Using the Bellman-Gronwall Lemma [34], we obtain

$$|e(t)| \leq \frac{\epsilon}{2} e^{-\frac{L}{\alpha}} + e^L \int_0^t e^{-\alpha(t-\tau)} d\tau \ \leq \frac{\epsilon}{2}. \tag{10.10}$$

Now suppose (for the sake of contradiction), that (x, u) does not belong to y_e for all $t \in [0, T]$; then, by continuity of $x(t)$, there exist a T^*, where $0 \leq T^* \leq T$, such that $(x(T^*), u(T^*)) \in \partial \mathrm{y}_e$, where $\partial \mathrm{y}_e$ denotes the boundary of y_e. If we carry out the same analysis for $t \in [0, T^*]$ we obtain that in this interval $|x(t) - \chi(t)| \leq \frac{\epsilon}{2}$, which is clearly a contradiction. Hence, (10.10) holds for all $t \in [0, T]$.

Lemma 3 *The system described by*

$$\dot{\chi}_i = -a_i \chi_i + w_i^* z(\chi, u), \ \ \chi_i(0) = \chi_i^0. \tag{10.11}$$

can be expressed as

$$\dot{\zeta}_i = -a_i \zeta_i + z_i, \ \ \zeta_i(0) = 0, \tag{10.12}$$

$$\chi_i = w_i^{*T} \zeta_i + e^{-a_i t} \chi_i^0. \tag{10.13}$$

Proof 2 *From (10.12) we have*

$$\zeta_i = \int_0^t e^{-a_i(t-\tau)} z(\chi(\tau), u(\tau)) d\tau$$

therefore,

$$w_i^{*T}\zeta_i + e^{-a_i t}\chi_i^0 = e^{-a_i t} + \int_0^t e^{-a_i(t-\tau)} w_i^{*T} z(\chi(\tau), u(\tau)) d\tau. \tag{10.14}$$

Using (10.11), the right hand side of (10.14) is equal to $\chi(t)$ and this concludes this proof.

Theorem 9 *Consider the RHONN model*

$$x_i = w_i^T \zeta_i, \quad i = 1, 2, \cdots, n \tag{10.15}$$

whose parameters are adjusted according to

$$\dot{w}_i = -\Gamma_i \zeta_i e_i, \ \ i = 1, 2, \cdots, n. \tag{10.16}$$

Then for $i = 1, 2, \cdots, n$

a) $e_i, \phi_i \in L_\infty$ (e_i and ϕ are uniformly bounded)
b) $\lim_{t\to\infty} e_i(t) = 0$

Proof 3 *Consider the Lyapunov function candidate*

$$V = \frac{1}{2}\sum_{i=1}^{n}\left(\phi_i^T \Gamma_i^{-1} \phi_i + \int_t^\infty \epsilon_i^e(\tau) d\tau\right). \tag{10.17}$$

Using (10.16) and $e_i = \phi_i^T \zeta_i - \epsilon_i$, where $\phi_i = w_i - w_i^*$ is the weight estimation error, the time derivative of V in (10.17) is expressed as

$$\dot{V} = \sum_{i=1}^{n}\left(-e_i \phi_i^T \zeta_i - \tfrac{1}{2}\epsilon_i^2\right) = \sum_{i=1}^{n}\left(-e_i(e_i + \epsilon_i) - \tfrac{1}{2}\epsilon_i^2\right)$$

$$= -\tfrac{1}{2}\sum_{i=1}^{n}\left(e_i^2 + (e_i + \epsilon_i)^2\right) \leq 0.$$

Since $\dot{V} \leq 0$, we obtain that $\phi_i \in L_\infty$. Moreover, using $e_i = \phi_i^T \zeta_i - \epsilon_i$ and the boundedness of ζ_i, we have that e_i is also bounded. To show that $e_i(t)$ converges to zero, we first note that since V is a non-increasing function of time and also bounded from below, the $\lim_{t\to\infty} V(t) = V_\infty$ exists; therefore, by integrating both sides of the above expression of the derivative $\dot{V}$ from $t = 0 to \infty$, and taking bounds we obtain

$$\int_0^\infty \sum_{i=1}^n e_i^2(\tau) d\tau \leq 2(V(0) - V_\infty),$$

so for $i = 1, 2, \cdots, n$ $e_i(t)$ is square integrable. Furthermore, using $e_i = \phi_i^T \zeta_i - \epsilon_i$:

$$\dot{e}_i(t) = \dot{\phi}_i^T \zeta_i + \phi_i^T \dot{\zeta}_i - \dot{\epsilon}_i = -e_i \zeta_i^T \Gamma_i \zeta_i - a_i \phi_i^T \zeta_i + \phi_i^T z - \dot{\epsilon}_i$$

Since $e_i, \zeta_i, \phi_i, \dot{\epsilon}_i$ are all bounded, $\dot{e}_i \in L_\infty$. Hence, by applying Barbalat's Lemma [73] we obtain that $\lim_{t\to\infty} e_i(t) = 0$.

Theorem 10 *Consider the filtered error RHONN model given by*

$$\dot{x}_i = -a_i x_i + w_i^T z, \quad i = 1, 2, \cdots, n, \tag{10.18}$$

whose weights are adjusted according to

$$\dot{w}_i = -\Gamma_i z e_i \tag{10.19}$$

where the adaptive gain Γ_i is a positive definite LxL matrix, and w_i is the estimate of the unknown vector w_i^*. Then for $i = 1, 2, \cdots, n$

(a) $e_i, \phi_i \in L_\infty$
(b) $\lim_{t\to\infty} e_i(t) = 0$

Proof 4 *Consider the Lyapunov function candidate*

$$V = \frac{1}{2} \sum_{i=1}^n \left(e_i^2 + \phi_i^T \Gamma_i^{-1} \phi_i \right) \tag{10.20}$$

Then, using $\dot{e}_i = -a_i e_i + \phi_i^T z, \ i = 1, 2, \cdots, n, where \phi_i := w_i - w_i^*$, and (10.19), and the fact that $\dot{\phi}_i = \dot{w}_i$, the time derivative of V in (10.20) satisfies

$$\dot{V} = -\sum_{i=1}^n a_i e_i^2 \leq 0 \tag{10.21}$$

Since $\dot{V} \leq 0$, from (10.20) we obtain that $e_i, \phi_i \in L_\infty$ for $i = 1, 2, \cdots, n$. Using this result in $\dot{e}_i = -a_i e_i + \phi_i^T z,\ \ i = 1, 2, \cdots, n, where \phi_i := w_i - w_i^*$, we also have that $\dot{e}_i \in L_\infty$. Now by employing the same techniques as in proof of theorem *9 it can be shown readily that $e_i \in L_2$, i.e., $e_i(t)$ is square integrable; therefore, by applying Barbalat's Lemma we obtain that $\lim_{t\to\infty} e_i(t) = 0$.*

Theorem 11 *Consider the filtered error RHONN model given by*

$$\dot{x}_i = -a_i x_i + w_i^T z, \quad i = 1, 2, \cdots, n, \tag{10.22}$$

whose weights are adjusted according to

$$\dot{w}_i = \begin{Bmatrix} -\Gamma_i z e_i, \ \ if |w_i| \leq M_i \\ -\Gamma_i z e_i - \sigma_i \Gamma_i w_i, \ \ if |w_i| > M_i \end{Bmatrix} \tag{10.23}$$

Then for $i = 1, 2, \cdots, n$

(a) $e_i, \phi_i \in L_\infty$
(b)there exist constants λ, m such that

$$\int_0^t |e(\tau)^2| d\tau \leq \lambda + m \int_0^t |v(\tau)^2| d\tau$$

Proof 5 *Consider the Lyapunov function candidate*

$$V = \frac{1}{2} \sum_{i=1}^{n} \left(\phi_i^T \Gamma_i^{-1} \phi_i + \int_t^\infty \epsilon_i^e(t) dt \right). \tag{10.24}$$

Using $\dot{e}_i = -a_i e_i + \phi_i^T z - v_i$,and (10.23) it can be shown that

$$\dot{V} = \sum_{i=1}^{n} (-a_i e_i^2 - e_i v_i - I_{w_i}^* \sigma_i \phi^T w_i). \tag{10.25}$$

where $I_{w_i}^*$ is the indicator function defined as $I_{w_i}^* = 1$ if $|w_i| > M_i$ and $I_{w_i}^* = 0$ if $|w_i| \leq M_i$. Since $\phi_i = w_i - w_i^*$, we have that

$\phi_i^T w_i = \frac{1}{2}\phi_i^T \phi_i + \frac{1}{2}(\phi_i^T \phi_i + 2\phi_i^T w_i^*) = \frac{1}{2}|\phi_i|^2 + \frac{1}{2}|w_i|^2 - \frac{1}{2}|w_i^*|^2$.

Since, by definition, $|w_i^*| \leq M_i$ and $|w_i| > M_i$ for $I_{w_i}^* = 1$, we have that

$$I_{w_i}^* \tfrac{\sigma_i}{2}(|w_i|^2 - |w_i^*|^2) \geq 0;$$

therefore, (10.25) becomes

$$\dot{V} \leq \sum_{i=1}^{n}(-a_i e_i^2 - I_{w_i}^* \frac{\sigma_i}{2}|\phi_i|^2 - e_i v_i), \tag{10.26}$$

$$\leq \sum_{i=1}^{n}(-\tfrac{a_i}{2} e_i^2 - \tfrac{\sigma_i}{2}|\phi_i|^2) + \sum_{i=1}^{n}\Big((1 - I_{w_i}^*)\tfrac{\sigma_i}{2}|\phi_i|^2 - \tfrac{a_i}{2}(e_i^2 + \tfrac{2}{a_i}e_i v_i)\Big).$$

So. we have that

$$\dot{V} \leq -\alpha V + \sum_{i=1}^{n}\left((1 - I_{w_i}^*)\frac{\sigma_i}{2}|\phi_i|^2 + \frac{v_i^2}{2a_i}\right), \tag{10.27}$$

where

$$\alpha := \min\Big\{a_i,\ \tfrac{\sigma_i}{\lambda_{\max}(\Gamma_i^{-1})};\ i = 1, 2, \cdots, n\Big\}$$

and $\lambda_{\max}(\Gamma_i^{-1}) > 0$ *denotes the maximum eigenvalue of* Γ_i^{-1}. *Since*

$$(1 - I_{w_i}^*)\tfrac{\sigma_i}{2}|\phi_i|^2 = \left\{\begin{array}{c} \frac{\sigma_i}{2}|\phi_i|^2 \quad if|w_i| \leq M_i \\ 0 \quad otherwise \end{array}\right\}$$

we obtain that $(1 - I_{w_i}^*)\frac{\sigma_i}{2}|\phi_i|^2 \leq \sigma_i M_i^2$. *Hence, (10.27) can be written in the form*

$$\dot{V} \leq -\alpha V + K,$$

where $K := \sum_{i=1}^{n}(\sigma_i M_i^2 + \bar{v}_i^2/2a_i)$ *and* $\bar{v}_i$ *is an upper bound for* v_i*; therefore, for* $V \geq V_0 = K/\alpha$, *we have that* $\dot{V} \leq 0$, *which implies that* $V \in L_\infty$. *Hence* $e_i, \phi_i \in L_\infty$.

To prove the second part, we note that by completing the square in (10.26) we obtain

$$\dot{V} \leq \sum_{i=1}^{n}(-a_i e_i^2 - e_i v_i) \leq \sum_{i=1}^{n}\left(-\frac{a_i}{2}e_i^2 + \frac{v_i^2}{2a_i}\right). \tag{10.28}$$

Integrating both sides of (10.28) yields

$$V(t) - V(0) \leq \sum_{i=1}^{n}\left(-\tfrac{a_i}{2}\int_0^t e_i(\tau)^2 d\tau + \tfrac{1}{2a_i}\int_0^t v_i(\tau)^2 d\tau\right),$$

$\leq -\frac{a_{min}}{2}\int_0^t |e(\tau)|^2 d\tau + \frac{1}{2a_{min}}\int_0^t |v(\tau)|^2 d\tau,$

where $a_{min} := \min a_i \ ; \ i = 1, 2, \cdots, n$; *therefore,*

$\int_0^t |e(\tau)|^2 d\tau \leq \frac{2}{2a_{min}}[V(0) - V(t)] + \frac{1}{2a_{min}^2}\int_0^t |v(\tau)|^2 d\tau,$

$\leq \lambda + \mu \int_0^t |v(\tau)|^2 d\tau,$

where $\lambda := (2/a_{min}) sup_{t\geq 0}[V(0) - V(t)]$ *and* $\mu := 1/a_{min}^2$. *This proves part (b) and concludes the proof of this theorem.*

Theorem 12 *Consider the identification scheme given by*

$$\dot{e} = Ae - X^*\tilde{W}S(\chi) - \tilde{X}WS(\chi) - X_1^*\tilde{W}_1 S_1(\chi)u - \tilde{X}_1 W_1 S_1(\chi)u \tag{10.29}$$

Provided that $[X_1 W_1 S_1(\chi)]^{-1}$ *exists the learning laws:*

a) For the elements of W *and* X

$$\begin{cases} \dot{W} = sgn(X^*)^T PeS^T \\ \dot{X} = PeS^T W^T \end{cases} \tag{10.30}$$

b) For the elements of W_1 *and* X_1

$$\begin{cases} \dot{W}_1 = sgn(X_1^*)^T Peu^T S_1^T \\ \dot{X}_1 = Peu^T S_1^T W_1^T \end{cases} \tag{10.31}$$

guarantee the following properties.

- $e, \hat{\chi}, \tilde{W}, \tilde{W}_1 \ \in \ L_\infty, \quad e, \hat{\chi} \in L_2$
- $\lim_{t\to\infty} e(t) = 0, \quad \lim_{t\to\infty} \hat{\chi}(t) = 0$
- $\lim_{t\to\infty} \dot{\tilde{W}}(t) = 0, \quad \lim_{t\to\infty} \dot{\tilde{W}}_1(t) = 0$

where the matrices $sgn(X^*)$ *and* $sgn(X_1^*)$ *are defined in the proof.*

Proof 6 *Consider the Lyapunov function candidate*

$V(e, \hat{\chi}, \tilde{X}, \tilde{W}, \tilde{X}_1, \tilde{W}_1) = \frac{1}{2}e^T Pe + \frac{1}{2}\hat{\chi}^T P\hat{\chi} +$

$+\frac{1}{2}tr\{\tilde{X}^T\tilde{X}\}+\frac{1}{2}tr\{\tilde{W}^T\Delta\tilde{W}\}$

$+\frac{1}{2}tr\{\tilde{X}_1^T\tilde{X}_1\}+\frac{1}{2}tr\{\tilde{W}_1^T\Delta_1\tilde{W}_1\}$

Where $P>0$ *is chosen to satisfy the Lyapunov equation*

$$PA+A^TP=-I$$

and matrices Δ *and* Δ_1 *are both diagonal* $n\cdot m\times n\cdot m$ *and defined as follows:*

$\Delta=diag\{(|\bar{f}_1^{1*}|,|\bar{f}_2^{1*}|,\ldots,|\bar{f}_m^{1*}|),(|\bar{f}_1^{2*}|,|\bar{f}_2^{2*}|,\ldots,|\bar{f}_m^{2*}|),\ldots,$

$(|\bar{f}_1^{m*}|,|\bar{f}_2^{m*}|,\ldots,|\bar{f}_m^{m*}|)\}$

and

$\Delta_1=diag\{(|\bar{g}_1^{1,1*}|,|\bar{g}_2^{1,1*}|,\ldots,|\bar{g}_m^{1,1*}|),$

$(|\bar{g}_1^{2,2*}|,|\bar{g}_2^{2,2*}|,\ldots,|\bar{g}_m^{2,2*}|),\ldots,$

$(|\bar{g}_1^{m,m*}|,|\bar{g}_2^{m,m*}|,\ldots,|\bar{g}_m^{m,m*}|)\}$

Thus $\Delta\geq 0$ *and* $\Delta_1\geq 0$.

Taking the derivative of the Lyapunov function candidate and taking into account (6.12) we get

$\dot{V}=\frac{1}{2}e^T\left(A^TP+PA\right)e+\frac{1}{2}\hat{\chi}^T\left(A^TP+PA\right)\hat{\chi}+$

$+\frac{1}{2}tr\{\dot{\tilde{X}}^T\tilde{X}\}+\frac{1}{2}tr\{\dot{\tilde{W}}^T\Delta\tilde{W}\}$

$+\frac{1}{2}tr\{\dot{\tilde{X}}_1^T\tilde{X}_1\}+\frac{1}{2}tr\{\dot{\tilde{W}}_1^T\Delta_1\tilde{W}_1\}\Rightarrow$

$\dot{V}=\frac{1}{2}e^T\left(A^TP+PA\right)e+\frac{1}{2}\hat{\chi}^T\left(A^TP+PA\right)\hat{\chi}+$

$\left(-\frac{1}{2}e^TP\tilde{X}WS-\frac{1}{2}e^TP\tilde{X}WS\right)-$

$\left(-\frac{1}{2}e^TPX^*\tilde{W}S-\frac{1}{2}e^TPX^*\tilde{W}S\right)-$

$-\left(\frac{1}{2}e^TP\tilde{X}_1W_1S_1u-\frac{1}{2}e^TP\tilde{X}_1W_1S_1u\right)-$

$-\left(\frac{1}{2}e^TPX_1^*\tilde{W}_1S_1u-\frac{1}{2}e^TPX_1^*\tilde{W}_1S_1u\right)+$

$tr\{\dot{\tilde{W}}^T\Delta\tilde{W}\}+tr\{\dot{\tilde{W}}_1^T\Delta_1\tilde{W}_1\}\Rightarrow$

$tr\{\dot{\tilde{X}}^T\tilde{X}\}+tr\{\dot{\tilde{X}}_1^T\tilde{X}_1\}\Rightarrow$

$\dot{V} = -\frac{1}{2}e^T e - \frac{1}{2}\hat{\chi}^T\hat{\chi} + e^T PX\tilde{W}S + e^T PX\tilde{W}_1 S_1 U + tr\{\dot{\tilde{W}}^T\tilde{W}\} + tr\{\dot{\tilde{W}}_1^T\tilde{W}_1\} \Rightarrow$
Take:

$$\begin{cases} tr\{\dot{\tilde{W}}^T\Delta\tilde{W}\} = e^T PX^*\tilde{W}S \\ tr\{\dot{\tilde{X}}^T\tilde{X}\} = e^T P\tilde{X}WS \\ tr\{\dot{\tilde{W}}_1^T\Delta_1\tilde{W}_1\} = e^T PX_1^*\tilde{W}_1 S_1 u \\ tr\{\dot{\tilde{X}}_1^T\tilde{X}_1\} = e^T P\tilde{X}_1 W_1 S_1 u \end{cases}$$

Then the Lyapunov function becomes:

$\dot{V} = -\frac{1}{2}e^T e - \frac{1}{2}\hat{\chi}^T\hat{\chi} \leq 0$

Using the fact that whenever $tr\{\dot{X}^T\tilde{X}\} = A\tilde{X}B$, where A is a row and B is a column vector, $\Rightarrow \dot{X} = A^T B^T$, we get:

$$\begin{cases} \Delta\dot{W} = X^{*T} PeS^T \\ \dot{X} = PeS^T W^T \\ \Delta_1\dot{W}_1 = X_1^{*T} Peu^T S_1^T \\ \dot{X}_1 = Peu^T S_1^T W_1^T \end{cases} \tag{10.32}$$

We write $X^{*T} = \Delta\{sgn(X^*)\}^T$
and $X_1^{*T} = \Delta_1\{sgn(X_1^*)\}^T$

where:

$sgn(X^*) = diag\{sgn(X^{1*}), sgn(X^{2*}), \ldots, sgn(X^{n*})\}$

where:

$sgn(X^{i*}) = [sgn(\bar{f}_1^{\,i,*}), sgn(\bar{f}_2^{\,i,*}), \ldots, sgn(\bar{f}_m^{\,i,*})]$

and:

$sgn(X_1^*) = diag\{sgn(X_1^{1*}), sgn(X_1^{2*}), \ldots, sgn(X_1^{n*})\}$

where:

$sgn(X_1^{i*}) = [sgn(\bar{g_1}^{i,i,*}), sgn(\bar{g_2}^{i,i,*}), \ldots, sgn(\bar{g_m}^{i,i,*})]$

Then equations (10.32) become:

$$\begin{cases} \dot{W} = sgn(X^*)^T PeS^T \\ \dot{X} = PeS^T W^T \\ \dot{W}_1 = sgn(X_1^*)^T Peu^T S_1^T \\ \dot{X}_1 = Peu^T S_1^T W_1^T \end{cases} \tag{10.33}$$

The update laws (10.33) are implementable, provided we know the signs of the partitions, which is a very reasonable assumption. However the centers of the partitions are automatically selected by our algorithm optimally.

Using the above Lyapunov function candidate V and proving that $\dot{V} \leq 0$ all properties of the theorem are assured [54].

For the proof of the next theorem we shall use some basic equations of the section of Neuro-Fuzzy Indirect Adaptive Control, and those are the following

$$\dot{e} = Ae - X^*\tilde{W}S(\chi) - \tilde{X}WS(\chi) - X_1^*\tilde{W}_1 S_1(\chi)u - \tilde{X}_1 W_1 S_1(\chi)u \tag{10.34}$$

$$u = -\left[X_1 W_1 S_1(\chi)\right]^{-1} XWS(\chi) \tag{10.35}$$

$$\dot{\hat{\chi}} = A\hat{\chi} \tag{10.36}$$

Theorem 13 *Consider the control scheme (10.34), (10.35), (10.36). The updating law:*

a) For the elements of W^i given by (10.30)

b) For the elements of ${}^1W^i$ given by the modified form:

$${}^1\dot{W}^i = -({}^1X^i)^T p_i e_i u_i s_i(\chi) \quad if \ \left|{}^1X^i \cdot {}^1W^i\right| > \theta_i > 0$$
$$or \quad \left|{}^1X^i \cdot {}^1W^i\right| = \theta_i \ and \quad {}^1X^i \cdot {}^1\dot{W}^i \leq 0$$

$${}^1\dot{W}^i = -({}^1X^i)^T p_i e_i u_i s_i(\chi) -$$
$$- \frac{2}{tr\{({}^1X^i)^T \, {}^1X^i\}} \, {}^1X^i \, {}^1W^i \, ({}^1X^i)^T \quad otherwise$$

guarantees the properties of theorem 12 and assures the existence of the control signal.

Proof 7 *In order the properties of theorem 12 to be valid it suffices to show that by using the modified updating law for* ${}^1W^i$ *the negativeness of the Lyapunov function is not compromised. Indeed the* **if** *part of the modified form of* ${}^1\dot{W}^i$ *is exactly the same with (10.31) and therefore according to theorem 12 the negativeness of* V *is in effect. The* if *part is used when the weights are at a certain distance (condition if* $\left|{}^1X^i \cdot {}^1W^i\right| > \theta_i$ *)from the forbidden plane or at the safe limit (condition* $\left|{}^1X^i \cdot {}^1W^i\right| = \theta_i$*) but with the direction of updating moving the weights far from the forbidden plane (condition* ${}^1X^i \cdot {}^1\dot{W}^i \leq 0$*).*

In the **otherwise** *part of* ${}^1\dot{W}^i$*, term* $-\frac{2}{tr\{({}^1X^i)^T\ {}^1X^i\}}\ {}^1X^i\,{}^1W^i\ ({}^1X^i)^T$ *determines the magnitude of weight* hopping*, which as explained later and is depicted in Fig. 6.4 has to be two times the distance of the current weight vector to the forbidden hyper-plane. Therefore the* ***existence*** *of the control signal is assured because the weights never reach the forbidden plane. Regarding the* ***negativeness*** *of* $\dot{V}$ *we proceed as follows.*

Let that ${}^1W^{*i}$ *contains the initial values of* ${}^1W^i$ *provided from the identification part such that* $\left|{}^1X^i \cdot {}^1W^{*i}\right| >> \theta_i$ *and that* ${}^1\tilde{W}^i = {}^1W^i - {}^1W^{*i}$*. Then, the weight hopping can be equivalently written with respect to* ${}^1\tilde{W}^i$ *as* $-2\theta_i {}^1\tilde{W}^i/\|{}^1\tilde{W}^i\|$*. Under this consideration the modified updating law is rewritten as* ${}^1\dot{W}^i = -({}^1X^i)^T p_i e_i u_i s_i(\chi) - 2\theta_i {}^1\tilde{W}^i/\|{}^1\tilde{W}^i\|$*. With this updating law it can be easily verified that* $\dot{V} = -\frac{1}{2}e^T e - \frac{1}{2}\hat{\chi}^T\hat{\chi} - \Theta$*, with* Θ *being a positive constant expressed as* $\Theta = \sum 2\theta_i \left(({}^1\tilde{W}^i)^T){}^1\tilde{W}^i)\right)/\|{}^1\tilde{W}^i\|$*, where the summation includes all weight vectors which require hopping. Therefore, the negativeness of* $\dot{V}$ *is actually enhance*

10.2 Matlab Code

In this section we present the matlab code we constructed in order to simulate the DC motor system according to the theory we developed in this diploma thesis.

Below, it is presented the main program.

```
%Simulation of a dc motor described by differential equations in bilinear
%form using neuro-fuzzy indirect adaptive control.
%Programed by Filipp Andreadis.

close all;
clear all;
clc;

global SP;
SP = 0.001; % sampling period

%Time edges.
T0 = 0;
Tf = 5;

x_initial = [ 3; -3];
x_init = [0.3; 0.3; 0.3]; %We mean here 0.3 of the nominal value. For example
%if the nominal angular velocity is 2500 rpm here we mean 0.3*2500 = 750
%rpm. The objective is to drive the angular velocity and the current to
%zero.This is called the regulation problem. If we want the states to
% follow some values or value trajectories we call it the tracking problem.

%Definition of the time space.
time_plot = T0:SP:Tf;

ind=0; %Index for the iterations.

%Desired Values for the states.
xdes(1,:) = 0*(sin(time_plot));
xdes(2,:) = 0*(cos(time_plot));

x = x_init;
xhut = x_initial;

%Definition of matrices W,W1 from the learning laws.

nf_order = 2;% Order of the neuro fuzzy model.
```

```
ho_w_depth = 2; % number of sigmoidal terms used in W.
ho_w1_depth = 2; % number of sigmoidal terms used in W1. Warning, the number
                 % of sigmoidal terms in S1 is equal to nf_order, because S1 is
                 % diagonal n x n, where n is the number of states

%Number of output membership functions partitions.
part_x = 6;
part_x1 = 3;

%Next the subvectors of matrices X and X1, which are considered known.

X(1,:) = [-163.3061 -148.9226 -153.1720 -79.9453  -175.4806 -18.1483];

X(2,:) = [25.1305 9.1845 -15.2403 18.0842 -9.2918 -0.1840];

Xmat = [X(1,:) zeros(1,part_x); zeros(1,part_x) X(2,:)];

Xmat_steady(1,1:part_x) = -1 ;
Xmat_steady(1,part_x+1:nf_order*part_x) = 0 ;
Xmat_steady(2,1:part_x) = 0 ;
Xmat_steady(2,part_x+1:nf_order*part_x) = [1 1 -1 1 -1 -1] ;

X1(1,:) = [148 149 150];

X1(2,:) = [42 43 44];

X1mat = [X1(1,:) zeros(1,part_x1); zeros(1,part_x1) X1(2,:)];

X1mat_steady(1,1:part_x1) = 1 ;
X1mat_steady(1,part_x1+1:nf_order*part_x1) = 0 ;
X1mat_steady(2,1:part_x1) = 0;
X1mat_steady(2,part_x1+1:nf_order*part_x1) = 1;

%Initial values for W and W1  weights.
%Initial values of W can be zeros. However initial values of
% W1 cannot be zero in order to avoid X1*W1*S1 to go to zero. In this case
% the inverse of the matrix does not exist. Dont forget that W1 has to be a
% block diagonal matrix.
Wmat = zeros(nf_order*part_x,ho_w_depth);

W1mat = zeros(nf_order*part_x1,ho_w1_depth);

for pp = 1:ho_w1_depth

    W1mat((pp-1)*part_x1+1:pp*part_x1,pp) = ones(part_x1,1)*0.04;
```

```
end

%Computatiuon of the matrices S(x),S1(x) calling the function sigmoidals.
a1 = 0.1 ; b1 = 1 ; % sigmoidal parameters used in w update
a2 = 6 ; b2 = 1 ; % sigmoidal parameters used in w1 update

sigm_W = Sigmoidals(x,a1,b1) ;
sigm_W1 = Sigmoidals(x,a2,b2) ;

sigm_W1_diag = diag(sigm_W1) ;

%Computation of the matrix P used in learning laws of W,W1.

%Initial Values for nxn stable matrix A which is used in Ricatti Equation.
A = [ -1 0 ; 0 -1 ] ;

%Definition of an apropriate matrix.
C = -[ 1 0 ; 0 1 ] ;
B = zeros(2) ;

%Use of are function of matlab solving Algebraic Ricatti Equations.
P = are(A,B,C) ;

for i = T0:SP:Tf  %Beggining of the iterations.
    ind = ind+1 ;

    x1plot(ind) = Xmat(1,1) ; %store the elements of Xmat.
    x2plot(ind) = Xmat(1,2) ;
    x3plot(ind) = Xmat(1,3) ;
    % Computation of the control input.
    u = - inv(X1mat*W1mat*sigm_W1_diag) * (Xmat*Wmat*sigm_W) ;
    u_plot(ind,:) = u' ;

    xold = x;
    % Real system dynamic equations.
    parameters = struct('t', {i}, 'update', {x}, 'u', {u}) ;
    x = RungeKutta(parameters, 'DC_MOT') ;
    x_plot(ind,:) = x' ;

    % Computation of the error.
    % The error is not between the desired and the actual
    % states of the system, but between the F-RHONN approximator and the
    % states of the system.
    xhut_old = xhut;
    parameters = struct( 't',{i}, 'update', {xhut}, 'Xmat_sign',{Xmat}
    ,'Wmat_sign',{Wmat}, 'sigm_W_sign',{sigm_W}, 'X1mat_sign',{X1mat},
```

```
'W1mat_sign',{W1mat}, 'sigm_W1_diag_sign',{sigm_W1_diag}, 'u',{u},
 'Amat_sign',{A} );
xest(:,ind) = RungeKutta(parameters,'Neural_Form');
xhut = xest(:,ind);

sigm_W = Sigmoidals(xhut,a1,b1) ;
sigm_W1 = Sigmoidals(xhut,a2,b2) ;
sigm_W1_diag = diag(sigm_W1) ;

e = xhut-x(1:2,:) ;
plot_e(ind,:) = e' ;

%Computation of the matices W,X,W1,X1.

for ii=1:nf_order
    %For segments of matrix W and matrix X.
    Xmat_temp(1,:) = Xmat(ii,(ii-1)*part_x+1 : ii*part_x); % take the row vector
    segment from X

    Wmat_temp = Wmat((ii-1)*part_x+1 : ii*part_x , 1:ho_w_depth);
    % take the submatrix W^i of W, that has to be updated
    Wmat_temp_old = Wmat_temp; % keep the not updated Wmat_temp in a separate matrix
    % now update submatrix W^i of W
    parameters = struct( 't',{i}, 'x_signal',{Xmat_temp}, 'update',
    {Wmat_temp}, 'e_signal',{e(ii)}, 'matrixS',{sigm_W}, 'Pmat',{P(ii,ii)} ) ;
    Wmat_temp = RungeKutta(parameters, 'Diff_eq_W') ; % updating of the submatrix
    Wmat((ii-1)*part_x+1 : ii*part_x , 1:ho_w_depth) = Wmat_temp;
    % store the updated submatrix into the large matrix W
    % now update vector X^i of X
    parameters = struct( 't',{i}, 'W_signal',{Wmat_temp_old},'update',
    {Xmat_temp}, 'e_signal',{e(ii)}, 'matrixS',{sigm_W}, 'Pmat',{P(ii,ii)} ) ;
    Xmat_temp = RungeKutta(parameters, 'Diff_eq_X') ;
    Xmat(ii,(ii-1)*part_x+1 : ii*part_x) = Xmat_temp(1,:);
    % Store the updated row vector segment into  X

    % Now the same procedure for W1 and X1
    X1mat_temp(1,:) = X1mat(ii,(ii-1)*part_x1+1 : ii*part_x1); % take
    the row vector segment from X1
    W1mat_temp(:,1) = W1mat((ii-1)*part_x1+1 : ii*part_x1 , ii); % take the subcolumn
    W1^i of W1, that has to be updated
    W1mat_temp_old = W1mat_temp; % keep the not updated Wmat_temp in a separate matrix
    % now update subvector W1^i of W1
    parameters = struct( 't',{i}, 'X1_signal',{X1mat_temp},'update',
    {W1mat_temp}, 'e_signal',{e(ii)}, 'u',{u(ii)}, 'matrixS1',
    {sigm_W1_diag(ii,ii)}, 'Pmat',{P(ii,ii)} ) ;
    W1mat_temp = RungeKutta(parameters, 'Diff_eq_W1') ;
```

```
        W1mat((ii-1)*part_x1+1 : ii*part_x1 , ii) = W1mat_temp(:,1); % Store the
        updated
         subcolumn W1^i into W1
        %now update the i-th subvector of X1
        parameters = struct( 't',{i}, 'W1_signal',{W1mat_temp_old},'update',
        {X1mat_temp}, 'e_signal',{e(ii)}, 'u',{u(ii)}, 'matrixS1',
        {sigm_W1_diag(ii,ii)}, 'Pmat',{P(ii,ii)} ) ;
        X1mat_temp = RungeKutta(parameters, 'Diff_eq_X1') ;
        X1mat(ii,(ii-1)*part_x1+1 : ii*part_x1) = X1mat_temp(1,:); % Store the
        updated row vector segment into X1

    end % {for ii loop}


end % end of the iterations {for i loop}.

%Construction of the plots x1 and x2.
figure
plot(time_plot,xdes(1,:),'b',time_plot,x_plot(:,1),'r');
title('Construction of the 1st state of the DC Motor')
xlabel('Time')
ylabel('Values of the desired and the actual state')
legend('Desired State','Actual State')

figure
plot(time_plot,xdes(2,:),'b',time_plot,x_plot(:,2),'r');
title('Construction of the 2nd state of the DC Motor')
xlabel('Time')
ylabel('Values of the desired and the actual state')
legend('Desired State','Actual State')

figure
plot(time_plot,x_plot(:,3),'r');
title('Construction of the 3rd state of the DC Motor')
xlabel('Time')
ylabel('Values of the desired and the actual state')
legend('Desired State','Actual State')

%Construction of the control input u.
figure
plot(time_plot,u_plot);
title('Construction of the control input of the Neuro-Fuzzy Model')
xlabel('Time')
ylabel('Values of the control input')
legend('1st input','2nd input')
```

```
%Construction of the error signal between the F-RHONN approximator and
the states of the system.
figure
plot(time_plot,plot_e)
title('Error between the F-RHONN approximator and the states of the system')
xlabel('Time')
ylabel('Values of the estimated error')
```

In the sequel we present all the appropriate .m files which we used in the main program.

It can be seen below, the .m files for the differential equations of the update laws for the matrices W, X, W_1, X_1, as long as the construction of the neural form of the RHONN model, the dynamical equations of the DC motor the calculated sigmoidals terms, and last the Runge-Kutta solution of the differential equations we solved in the main program.

```
%This m-file contains all differential equations of W.

function xdot = Diff_eq_W(params);

X = params.x_signal ;
P_mat = params.Pmat ;
S_mat = params.matrixS ;
error_signal = params.e_signal ;

xdot = [sign(X')] * (S_mat') * (P_mat) * (error_signal)  ;

%This m-file contains all differential equations of X.

function xdot = Diff_eq_X(params);

W = params.W_signal ;
P_mat = params.Pmat ;
S_mat = params.matrixS ;
error_signal = params.e_signal ;

xdot = S_mat' * W' * P_mat * error_signal ;

%This m-file contains all differential equations of W1.

function xdot = Diff_eq_W1(params);

X1 = params.X1_signal ;
```

```
P_mat = params.Pmat ;
S1_mat = params.matrixS1 ;
error_signal = params.e_signal ;
u_signal = params.u ;

xdot = [sign(X1')] * (P_mat) * (error_signal) * u_signal' * S1_mat;

%This m-file contains all differential equations of X1.

function xdot = Diff_eq_X1(params);

W1 = params.W1_signal ;
P_mat = params.Pmat ;
S1_mat = params.matrixS1 ;
error_signal = params.e_signal ;
u_signal = params.u ;

xdot = [sign(W1')] * (P_mat) * (error_signal) * u_signal * S1_mat ;

%This m-file contains the neural form that the actual system(DC MOTOR) is
%modeled by.

function xdot = Neural_Form(params);

x_signal = params.update ;
X = params.Xmat_sign ;
W = params.Wmat_sign ;
Sx = params.sigm_W_sign ;
X1 = params.X1mat_sign ;
W1 = params.W1mat_sign ;
S1x = params.sigm_W1_diag_sign ;
u_signal = params.u;
A = params.Amat_sign;

xdot = A*x_signal + X*W*Sx + X1*W1*S1x*u_signal ;

function xdot = DC_MOT(parameters);

x = parameters.update;
u = parameters.u;

xdot(1,1) = -148.88*x(1,1)-148.88*x(2,1)*x(3,1)+148.88*u(1) ;
xdot(2,1) = 42.91*x(1,1)*x(3,1)-0.0129*x(2,1) ;
xdot(3,1) = -1/31.88*(x(3,1)/(2.6-1.6*x(3,1)))+1/31.88*u(2) ;

%Function of sigmoidals that we use in order to compute the high order
```

```
%combination of sigmoid functions.


function sig=Sigmoidals(x,a,b);

%We produce the sigmoidal functions as follows:

sigt = a/(1+exp(-b*x(1,1)));

sig1 = sigt^2 ;

sigt2 = a/(1+exp(-b*x(2,1)));

sig2 = sigt2^2 ;


%sigmoid vectors

sig = [ sig1 ; sig2 ];

function xnew = RungeKutta(parameters,fname);

% Fourth order Runge-Kutta integration subroutine (fixed time step)
% File RungeKutta.m
% SP Sampling period
% x the state vector
% xdot the derivative of the state vector

global SP;


xdot = feval(fname, parameters);

x = parameters.update;
t = parameters.t;

K1 = SP*xdot;
x1 = x + 0.5*K1;  parameters.update = x1;


t = t + 0.5*SP;  parameters.t = t;
x1dot = feval(fname, parameters);

K2 = SP*x1dot;
x1 = x + 0.5*K2;   parameters.update = x1;
```

```
x1dot = feval(fname, parameters);

K3 = SP*x1dot;
x1 = x + K3;  parameters.update = x1;

t = t + 0.5*SP;   parameters.t = t;
x1dot = feval(fname, parameters);
K4 = SP*x1dot;


Dx = 1/6*(K1+2*K2+2*K3+K4);
xnew = x + Dx;
```

I hereby affirm that I composed this work independently and used no other than the specified sources and tools and that I marked all quotes as such.