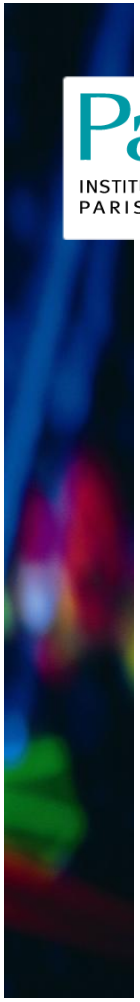
Identification de systèmes utilisant les réseaux de neurones : un compromis entre précision, complexité et charge de calculs.

Héctor Manuel Romero Ugalde



ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

ARTS ET MÉTIERS ParisTech

2013-ENAM-0001

École doctorale n° 432 : Sciences des Métiers de l'Ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure d'Arts et Métiers

Spécialité " Automatique "

présentée et soutenue publiquement par

Héctor Manuel ROMERO UGALDE

le 16 Janvier 2013

Identification de systèmes utilisant les réseaux de neurones: un compromis entre précision, complexité et charge de calculs.

Directeur de thèse : **Jean-Claude CARMONA**
Directeur de thèse : **Víctor ALVARADO-MARTINEZ**

Jury

Pr. Germain GARCIA, Professeur des Universités, Unité de recherche *LAAS,* (France) — Rapporteur
Dr. Dionisio SUÁREZ, Chercheur, Gestion de Surveillance des processus, IIE (Mexique) — Rapporteur
Dr. Manuel ADAM MEDINA, Professeur Associé, Département Electronique, CENIDET (Mexique) — Examinateur
Dr. Víctor ALVARADO-MARTINEZ, Professeur Associé, Département Electronique, CENIDET (Mexique) — Examinateur
Pr. Jean-Claude CARMONA, Professeur des Universités, LSIS-INSM, Arts et Métiers Paris Tech (France) — Examinateur

T H È S E

Arts et Métiers ParisTech - Centre d'Aix-en-Provence
Nom de l'Unité de recherche

“System identification using neural networks: a balanced accuracy, complexity and computational cost approach.”

presented by:

Héctor Manuel Romero Ugalde

M. of Sc. in Electronic engineering

A thesis submitted to

the **CENTRO NACIONAL DE INVESTIGACIÓN y DESARROLLO TECNOLÓGICO**

and

the **ECOLE NATIONALE SUPÉRIEURE D’ARTS ET MÉTIERS**

in a total fulfillment of the requirements for the degree of

Doctor of Engineering
in
Automatic Control

Advisors:

Dr Víctor ALVARADO MARTINEZ..... (CENIDET, Morelos, Mexique)

Pr Jean-Claude CARMONA..... (ENSAM, Aix en Provence, France)

January 2013

.

Dedicated to my family: Héctor Romero Cuervo, Martha Beatriz Ugalde Baca, Sagrario C. Romero Ugalde (connie), Jesús Javier Romero Ugalde (javi) and Concepción Baca Vélez.

A mis padres Martha Beatriz Ugalde Baca y Héctor Romero Cuervo que me dieron la vida, por ser los mejores padres del mundo y un ejemplo a seguir, por haberme apoyado e inculcado los valores que me regirán toda mi existencia.

A mis hermanos Sagrario C. Romero Ugalde (connie) y Jesús Javier Romero Ugalde (javi), por los grandes momentos que hemos disfrutado juntos, por orientarme cuando he estado indeciso y principalmente porque siempre han demostrado ser mis mejores amigos.

A mi abuelita Concepción Baca Vélez quien cuidó de mí y me oriento en los primeros años de mi infancia.

Por ser las personas más importantes que existen en mi vida.

Especialmente a ti señor Jesús que estás a mi lado en todos los momentos de mi vida.

GRACIAS
Héctor Manuel Romero Ugalde

Acknowledgement

This work is done in the framework of a co-direction between the Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), Cuernavaca, Morelos, Mexico and the Ecole Nationale Supérieure d'Arts et Métiers (ENSAM), Aix en Provence, France.

Foremost, I would like to express my sincere gratitude to my advisors Prof. Jean-Claude Carmona and Dr. Víctor Manuel Alvarado Martínez for the continuous support of my PhD study and research, for their patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

Besides my advisors, I would like to thank the rest of my thesis committee: Dr. Manuel Adam Medina, Dr. Carlos Manuel Astorga Zaragoza, Dr. Carlos Daniel García Beltrán, Dr. Juan Reyes Reyes, Dr. Germain Garcia, y Dr. Dionisio Suárez, for their encouragement, insightful comments and hard questions.

My sincere thanks also go to my fellow's lab mates and professors of the two institutions CENIDET and ENSAM, for offering me their knowledge and friendship.

I thank the financial support from CONACYT and EGIDE.

Thank to my friends: Mario Sotomayor, Mario J., Gary, Sandra, Gabriel C., Marinné, Vero, Iván H., Juan A., Miguel, Jhoel, Antoni, Aksel, Xavier, Adeline, Jose, Fany, Cecilia, Mehdi, Amir, Zeineb, Wendole, Ruding, Christophe, Cinda, Maud, Frederic, Alek, Zongcheng, Benjamin, Matthieu, Julien, Mikael, Aymen and Nouha for offering me very good moments.

A would like especially to thank to my girlfriend Anne-Lise Sampieri and her family, for giving me love and a family when I was far of my country.

I do not forget all my family members, uncles, aunts, cousins and godparents. Thanks for all your support! Thank to my grandmothers: Concha and Eva for giving me the best parents in the world. And to my aunt Reyna, thank for help to my father during his childhood.

Finally and more important, I would like to thank my family: Héctor Romero, Martha Ugalde, Connie, Javi and Concha for giving me their support and love throughout my life.

Contents

List of Figures

List of Tables

List of symbols

θ	Parameter of a model.
$\hat{\theta}$	Estimation of a parameter.
θ^*	Optimal estimation of a parameter.
w_i	Synaptic weights.
n	Number of inputs of the neuron.
Z_h	Threshold or bias.
$\hat{y}$	Model output, Neural network output.
n_a	Number of pass outputs.
n_b	Number of pass inputs.
n_k	Dead time in the system.
$y(k)$	Measured or real output.
$e(k)$	Prediction error.
k	Instant of time or iteration
N	Total number of data for the estimation.
R	Modifies the search direction.
η	Step size or leaning rate.
E	Objective function or criterion function.
γ	Number of epochs.
γ^*	Number of epochs required for the estimation.
η_0	Initial value of η.
z_p	Synaptic weight.
φ	Activation function.
T	Internal signal of the neural network.
r_p	Internal signal of the neural network.
J	Regressors vector.
nn	Number of neurons.
J_u	Input regressors vector.
$J_{\hat{y}}$	Output regressors vector.
X	Synaptic weight.
Z_b	Synaptic weight.
Z_a	Synaptic weight.
Z_h	Synaptic weight.
V_{b_i}	Synaptic weights.
V_{a_i}	Synaptic weights.
W_{b_i}	Synaptic weights.
W_{a_i}	Synaptic weights.

W_B	Synaptic weight.
W_A	Synaptic weight.
V_B	Synaptic weight.
V_A	Synaptic weight.
H	Synaptic weight.
r_b	Internal signal of the neural network.
r_a	Internal signal of the neural network.
O	Number of parameters of a NN times the number data required for its estimation.
W_{bh_i}	Synaptic weights.
W_{ah_i}	Synaptic weights.
V_{bh}	Synaptic weight.
V_{ah}	Synaptic weight.
Q	Synaptic weight.
$bMat$	Synaptic weight.
$aVec$	Synaptic weight.
$cVec$	Synaptic weight.
P	Synaptic weight.
L	Synaptic weight.
d	Synaptic weight.
y_{NL}	Internal signal of the neural network.
y_{NL1}	Internal signal of the neural network.
y_{NL2}	Internal signal of the neural network.
y_L	Internal signal of the neural network.
y_{L1}	Internal signal of the neural network.
Q_1	Synaptic weight.
P_1	Synaptic weight.
$aVec_1$	Synaptic weight.
$cVec_1$	Synaptic weight.
Q_2	Synaptic weight.
μ_t	The mean value of the simulation error.
s_t	The standard deviation of the error.
e_{RMSt}	The root mean square (RMS) of the error for the validation data.
e_{RMSe}	The root mean square (RMS) of the error for the estimation data.
n_p	Number of parameters.
TT	Time in minutes required to obtain each model.
$FITF$	Percentage of the measured output that was explained by the model.
E_{x1}	Frequency response of the measured data.
E_{x2}	Frequency response of the estimated data.

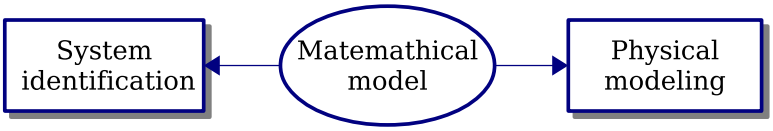
Abbreviations

RBF	Radial basis networks.
RNN	recurrent neural networks.
MLP	Multilayer Perceptron.
MSE	Mean square error.
LSAD	Least Sum Absolute Deviation.
GA	Genetic algorithms.
PSO	Particle swarm optimization.
GP	Genetic programming.
SVAR	Singular value architectural recombination.
LM	Levenberg-Marquardt.
MA	Memetic Algorithms.
DE	Differential evolution.
BP	Backpropagation.
ODE	opposition based differential evolution.
FFT	Fast Fourier Transform.
FRF	Frequency Response Function.

General introduction

In engineering fields a model is a mathematic representation of a real system used for analysis, supervision, fault detection, prediction, estimation of unmeasurable variables, optimization and model-based control process,... [1, 2, 3, 4, 5, 6, 7, 8, 9].

Basically, a model can be constructed according two routes or a combination of them [10] (see the figure below). One route is called physical modeling, based on the physical mechanisms that govern the system's behavior. The models thus obtained are adequate approximations of the real process [11]. But, in many cases involving complex nonlinear systems, it is very difficult or almost impossible to derive dynamic models based on all the physical phenomena involved [12, 13, 14].



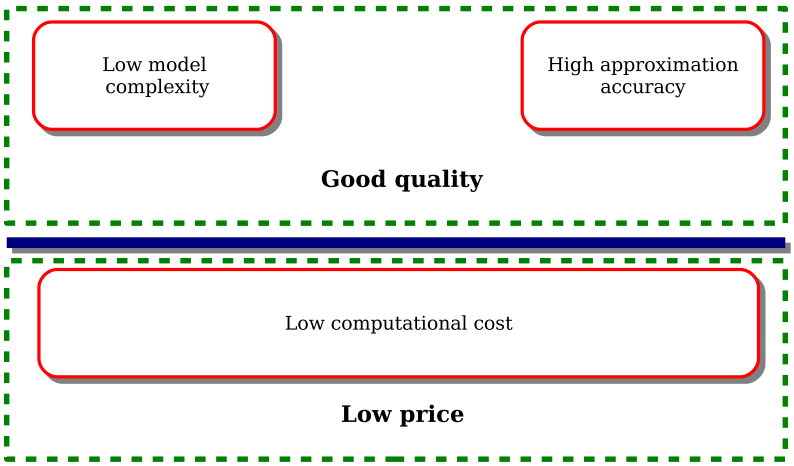
System identification/Physical modeling.

On the contrary, the other route called "System identification" provides an alternative way to build mathematical models to capture the system dynamics. The models are constructed based on the input-output data obtained by the experimentation with the real system [11, 12, 13, 14]. The main advantage of this technique is the fact that no extensive knowledge on the physical behavior of the real system is required [11, 12]. In the sequel, we shall focus on this approach.

System identification

The earliest works on system identification were mainly conducted to find the "true system", i.e. the model which provides exactly the output data when excited by the input data [11]. The researchers were not concerned by the complexity of the models and the handling of the model parameters computation ([15, 16]).

More recently, due to the critical role of system modeling in engineering fields, in particular model-based control, the goal in system identification tends to generate models with good "quality" at a low "price" [11] (see the figure below). In the sequel, for greatest convenience we shall denote, "quality" the balance between accuracy and complexity of the model, and "price" the computational cost to generate it [11].



New challenge.

Several methods for system identification have been widely discussed ([5, 10, 11, 17, 18]). But all of them are based on the following steps that define a general system identification methodology (see figure bellow). Hereafter are the "golden steps" for system identification.

1.- Three basis entities

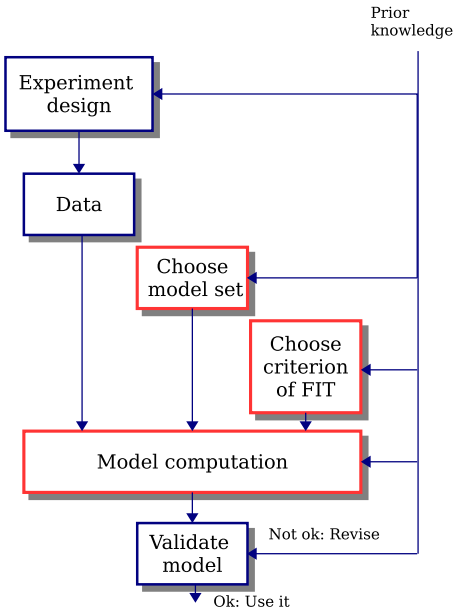
- The data record. It all starts with the necessary to have a very good dataset at our disposal. i.e. a set of data which contains the maximum information about the system dynamics. The objective of the experiment design is thus to generate data maximally informative.
- The set of models or the model structure. A set of candidate models is obtained by specifying within which collection of models we are going to look for a suitable candidate. Ideally, the "true system" should belong to this model set.
- Model estimation. Suppose a set of candidate models has been selected, and supposed they are parametrized using a parameter vector θ. The search for the best suitable model within the set, then becomes a problem of determining, or estimating θ, according to a given criterion. The better we choose both a criterion and an optimization algorithm, the better result estimated model is.

2.- Model validation. After having settled up the preceding three choices, we have at our disposal a candidate model, the one in the parametrized model set which describes the best the data according to a given criterion. It is then necessary to test whether this model is "good enough" or valid for its purpose. Such tests are known as *Model validation*.

3.- The system identification loop. The system identification procedure has a natural logical flow: first collect the data, then choose a model set and estimate the "best" model in this set. If the model candidate does not pass the model validation, we must go back to revise the various steps of the procedure. The model may be deficient for several reasons:

- The numerical procedure fails to find the best model candidate according to our criterion.
- The criterion is not well chosen.
- The model set is not appropriate, in that it does not contain any "good enough" description of the system.

– The data set is not informative enough to provide guidance in selecting good models.



General system identification method.

The choice of the model structure and the model estimation (criterion and optimization algorithm) [19] certainly have a considerable effect on both the quality of the resulting estimated model and the price to generate it [11, 20]. Hence, these parts of the system identification procedure will be addressed in this thesis.

The choice of the model structure is often based on the intended use of the model. For example, for applications as model-based control, in particular inverse control, accurate process representations with a reasonable low complexity are required [3, 21]. If some physical insight is available, it is recommended to used it in order to select a model structure with the maximum chance of containing the "true system". But, in the industrial fields, different process systems show non linearities and uncertainties which can be considered as partially or totally black-box [22]. In that cases where no physical insight is available or used, a model structure which is known to have good flexibility and which has been successfully used in the past, should be used. These cases, known as black box system identification, will be addressed in this thesis.

Once a set of candidate models, parametrized by the parameter vector θ, has been selected, the model estimation consists in finding the best estimation $\hat{\theta}$. Ideally this estimation should yield the $\hat{\theta} = \theta^*$ corresponding to the "true system". Moreover, computational cost should be taken into consideration. In practice, the goal is to determine $\hat{\theta}$ which leads to a model which represents the system as accurately as possible with a low computational cost.

More precisely, this estimation consists in searching the $\hat{\theta}$ which minimizes an objective function (criterion function). The Mean Square Error (MSE) criterion is the most commonly used in system identification. Generally, the MSE estimation leads to significant performances under classical assumptions. Although, sometimes the system data set contains corrupted data, called outliers which greatly damage the MSE performances. In this case, robust estimators using robust norms are used. See the Huber's function, Talwars function [23].... for example.

Once the objective function is chosen, the search for $\hat{\theta}$ which minimizes the objective function is achieved by an optimization algorithm. The most commonly encountered in system identification, in particular in neural networks, are the steepest descent algorithm, the Gauss-Newton algorithm and the Levenberg-Marquardt algorithm. As a consequence, both the computational cost and the accuracy of the obtained model depend on the learning algorithm chosen to estimate the model parameters, for the same reason as the objective function and the structure of the model.

Remember that, in order to cover the cases when no reliable physical insight is available, black box nonlinear system identification methods will be addressed in this thesis. Recent research results have shown, that the nice properties of universal approximation of neural networks make them suitable for modeling complex nonlinear systems when we consider the plant as a black-box [4, 8, 17, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31], especially those which are hard to describe mathematically [29, 30, 32, 33]. Consequently, let us now remind the main features of this approach.

System identification using neural networks

As presented above, two important steps in a general system identification method are: the model structure selection and the model estimation. In neural network theory, these two steps are addressed to as: neural network architecture selection and neural network training, respectively.

Let us remember that artificial neural networks consist in a large number of interconnected processing elements known as neurons [32]. Depending on the way the neurons are structured, neural networks can be classified as feedforward, recurrent or cellular neural networks [17, 23, 24, 25, 31, 34, 35].

For system identification and control, the most used architectures are feedforward and recurrent neural networks. In these type of configurations, the neurons are usually arranged in layers. The number of layers and the number of neurons in each layer play an important role in the neural networks design. While the number of layers and neurons increases, the approximation capabilities of the model increases too. Unfortunately, the model complexity and the computational cost correspondingly increase as well.

Within the feedforward and recurrent neural networks, the most encountered in system identification [24] are: the Single layer feedforward networks, the Multilayer feedforward networks, the Radial basis networks (RBF) and the Dynamics or recurrent neural networks.

The Feedforward networks are suitable for the approximation of complex static functions [31, 35]. However, the major drawback of such type of neural networks in describing dynamic functions, is that the weights updating does not utilize information on the local date structure. Moreover, the function approximation is sensitive to the quality of training data. Since recurrent neural networks (RNN) incorporate feedback, they can successfully overcome these disadvantages. Moreover, recent results show that RNN can outperform feedforward neural networks such as Multilayer Perceptron (MLP) or RBF networks [36]. Even more, they can yield smaller structures [37]. Due to its dynamic characteristics, RNN has a great potential for nonlinear dynamic system identification and control [27, 31, 34, 37, 38, 39]. Therefore, RNN will be used in this thesis for black box nonlinear system identification.

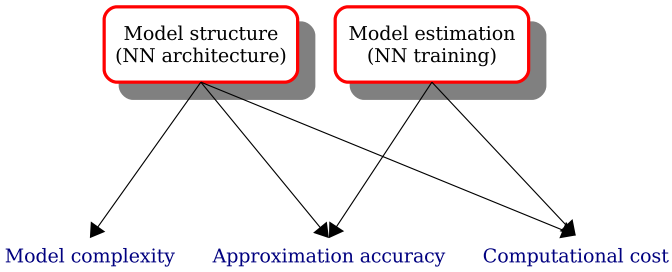
Once the model structure is chosen, the next step is the estimation of the synaptic weights, i.e. the parameter vector $\hat{\theta}$. This is achieved during the training phase or model estimation phase. Here, the idea is, as already explained, to minimize an objective function (MSE, LSAD, Huber's function, etc) by means of a learning algorithm. The computational bulk consequently depends on the learning algorithm, the structure and the criterion function chosen.

Problem statement: A convenient trade-off

Lest us remember that, one of the most important advantages of black box modeling approaches is the limited physical insight required to develop the model [40], but as a trade-off, these techniques imply the use of model structures which are as flexible as possible. Generally, this property leads to a high number of parameters of the structure [41].

Neural network identification techniques have the same problem: a large number of neurons (parameters) is usually necessary to correctly describe complex nonlinear systems [28]. As mentioned above, the number of neurons directly affects the quality (i.e. model complexity and approximation accuracy) and the computational cost.

In order to improve the preceding characteristics, we must conveniently choose the model structure and the model estimator (i.e. the learning algorithm). The figure below, illustrate the different influences of the different choices we have to do.



These choices are somewhat contradictory and necessarily a good compromise should be proposed [42, 43]. The different works in neural networks system identification dealing with this compromise can be divided in two types. The first part is dedicated to improve the quality of the models by means of finding a good compromise between the model complexity and the approximation accuracy. To achieve this, the authors directly deal with the neural network architecture (see the figure above). The second part tends to estimate accurate models with low computational cost. In the literature, this is achieved by the choice of the learning algorithm or by a convenient choice of the model structure.

State of art

Let us first present the works which tend to the best trade-off between the model complexity and approximation accuracy. Trial and error is one of this techniques [44]. In this approach, the user test different number of neurons. When the "best compromise" between the number of neurons and the accuracy of the architecture is found, the tests are stooped. Different works based on this approach [21, 40, 45, 46, 47, 48, 49, 50] propose models with a good quality level. Although, this procedure is laborious and it mays not lead to the "best compromise" between the model complexity and the approximation accuracy [40, 51]. Other techniques trying to improve the

same approach are known as pruning. Pruning based techniques have been successfully used for structural optimization [5, 44, 52, 53, 54, 55, 56, 57, 58, 59]. Pruning methods attempt to find a quick solution by starting with a large network architecture then reducing it according to a subjective criterion. In this approach [60, 61], besides optimizing the number of neurons, the connections between the neurons are also optimized. In fact, the synaptic weights which are "not important" are eliminated, leading to models with less number of parameter. More recently, other evolutionary techniques has been employed in order to derive "optimal" structures. For example, genetic algorithms (GAs) in [28, 30, 51, 62], dissimilation particle swarm optimization (PSO) in [33], genetic programming (GP) in [3], a combination of GA and singular value architectural recombination (SVAR) in [63]. As in the pruning techniques, each evolution of the neural network is based on a criterion chosen by the designer. The previously outlined techniques, based on the evolution of the neural network, have been successfully applied for structural optimization, but its main disadvantage is the excessive requirement of time to find the most convenient number of neurons, since the neural network is trained every time the model is modified or restructured [29]. Moreover, to solve the problem of finding the best trade-off between model complexity and model accuracy, rather subjective criterion is always used to decide whether the evolution of the neural network is appropriate and sufficient.

Now, let us present the works devoted to improve the balance between the accuracy and the price of the model. As mentioned above, this can be achieved by the implementation of an efficient learning algorithm or by a convenient choice of the model structure. Different learning algorithms can be applied in order to improve the accuracy and reduce the computational burden. In [34] a Kalman filter-based algorithm with faster convergence is proposed. Although this algorithm is more complex than the gradient based algorithms, a decoupling technique is used to decrease the computational burden. In [20] a combination of clustering, gradient and Kalman filter algorithms leads to a quick and efficient approach for modeling. In [31] a Bounding Ellipsoid algorithm for high computational efficiency and fast convergence is proposed. In [64] an improved simultaneous perturbation stochastic approximation (SPSA) algorithm yields an improved model in terms of time of convergence and a smaller identification error. In [19] a combination of GAs and LM algorithms take advantage of the global search of GA and the estimation ability of LM to improve the accuracy and reduce the computation time. In [65] two Memetic Algorithms (MA), combining evolutionary algorithms (i.e. GA and Differential evolution (DE), which are global search methods)) with a backpropagation (BP) learning algorithm are proposed. This algorithm has faster convergence in comparison with only evolutionary computation and avoids the possibility of local minima normally existing in the gradient algorithms. This result is extended in [66] where the PSO is combined to with BP. Following the same idea, in [29] an opposition based differential evolution (ODE) algorithm combined with LM is used for training the feed-forward neural network. These results are interesting owing to its convergences properties. In the previous works the improvements are based on the algorithms. Even if, the accuracy of the model, the price to generate it or both of them are improved, the computational cost is still affected by the complexity of the model structure. In [37, 39] the improvements of the computational cost and/or the approximation accuracy are based on the choice of a convenient model structure, that is, by the design of the neural network.

With the conviction that the improvement of the model quality and the reduction of the price of the model, by convenience denoted in the sequel "constraints", are directly linked to a suitable neural network design, we decided to face to this neural network "constraints" by proposing first a neural network (inspired in [67]) adapted to system identification purposes, secondly a

model complexity reduction procedure and thirdly a computational cost reduction approach. These are the main lines of our research contribution. We shall see that using this original methodology, we provide ready-to-use accurate models with a small number of parameters at a low price. More precisely, the main technicals issues of this thesis depend on the particular selection of two factors: the activation functions in each layer and the initial conditions of the synaptic weights. The model complexity reduction approach is developed in two steps: the first step consists in training a neural network with a large enough number of neurons to provide the intended accuracy. In the second step, an appropriate procedure significantly reduces the number of neurons without loss of accuracy. Moreover, we show that the proposed architecture nevertheless remains sufficiently general to provide a wide range of useful model types with good quality at a low price. These model types are currently used in particular for model-based control techniques [4, 5, 6, 7, 71]. The learning algorithm used to optimize the synaptic weights is the classical steepest descent algorithm which is one of the simplest algorithms, since, in a first time, we deliberately decide not to treat this point

In the sequel, this manuscript is organized as follows.

Chapter 1 summarizes the background on neural network which may be helpful to understand the main components and steps for their application in system identification.

In Chapter 2, we focus on the particular design of the different architectures, which allow us to investigate the balance complexity/accuracy and the reduction of the computational cost.

In Chapter 3, the main contributions of this thesis are presented. The theorems on which the model reduction approach and the computational cost reduction approach are based, are presented in details.

In Chapter 4, a methodology based on the previous approaches is described in order to yield models with good accuracy, low complexity and low price. Moreover, the proposed method is applied to different neural network architectures in order to demonstrate the interest of these approaches.

Results of the identification of different systems are discussed in Chapter 5. These systems consist in simulations ones, as the classical Wiener-Hammerstein benchmark, but also experimental setups like an acoustic system, a robot arm, etc.

In Chapter 6 conclusions are given and some perspectives of the present works are drawn.

.

Chapter 1

System identification using neural networks: Background

This chapter deals with neural networks and its application for system identification. The principle on which neural networks are based and its interesting characteristics which make them applicable for black box system identification are presented here. Moreover, the concept related to the structure selection such as the number of layers, the number of neurons and the activation functions which define the structure of an identification model and its accuracy are introduced, as well as, the concepts of learning algorithm, number of epochs, learning coefficient, gradient and backpropagation algorithms which are related to the neural network training. Both, the structure selection and the neural networks training are the main steps in the system identification procedure as we will see.

1.1 Artificial neurons

Artificial neural networks are originally motivated by biological structures in the brain of humans and animals. Functionally, the biological neurons are simple information processors containing three main components: dendrites, axon and cell body.

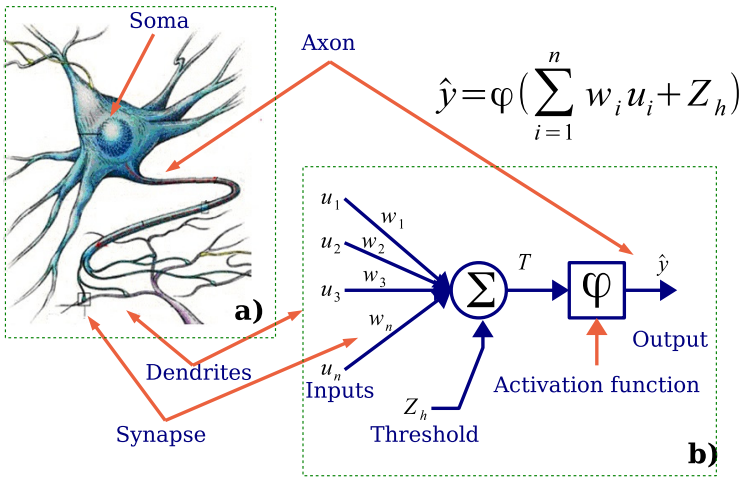


Figure 1.1: a) Biological neuron. b) Artificial neuron.

Fig. 1.1 a) shows a biological neuron. The dendrites are the input connections which conduct the nerve impulses toward the cell body soma. The axon extends away from the cell body and provides the path over which the information travels to other neurons. The dendrites are tube-like extensions that branch repeatedly and form a bushy tree around the cell body (Soma). They provide the main path on which the neurons receive coming information and the impulse sweeps along the axon until it reaches the end. The junction point of an axon with a dendrite of another neuron is called a synapse [17, 23, 24].

In that sense, Fig. 1.1 b) shows an artificial neuron, the input (dendrite) receives external information or information from other neurons. This information is pondered by the synaptic weights (synapses). Then the sum of the weighted inputs and the threshold is processed by a function called activation function or output function (linear, sigmoid, gaussian or others, see Appendix A), finally, the output (axon) conduces the information to other neurons or to the outside.

The mathematical representation of the artificial neuron shown in Fig. 1.1 b) is given by (1.1).

$$y = \varphi\left(\sum_{i=1}^{n} w_i u_i + \theta\right) \tag{1.1}$$

where, φ is the activation function, u_i are the inputs, w_i are the synaptic weights, n is the number of inputs of the neuron, θ is the threshold or bias and y is the output.

1.2 Neural network architectures and system identification

Artificial neurons are the fundamental units for the operation of a neural network. In order to reach its real potential, these neurons should be associated to others neurons, developing the main existent neural networks architectures.

Depending on the structure in which the artificial neurons are arranged in a neural network architecture, neural networks can be classified as feedforward, recurrent or cellular neural networks [17, 23, 24, 25, 31, 34, 35].

In the feedforward or static neural networks (see Fig. 1.2), the connections between neurons do not create feedback loops. In this kind of architectures, usually a quick response is produced.

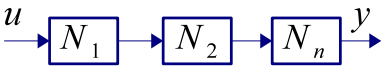


Figure 1.2: Feedforward neural networks.

In the recurrent neural networks (RNN) (see Fig. 1.3), there are feedback loops between neurons. In some recurrent neural networks, every input is presented in the time t, and the response is produced after some iterations.

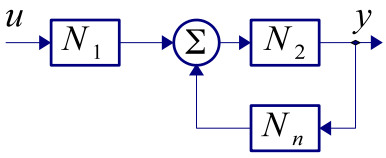


Figure 1.3: Recurrent neural networks.

Cellular neural networks (see Fig. 1.4) consist in the connection of many neurons specially located, each cell is connected only to the neighboring cells, although, one cell affects to the others indirectly, by the signal propagation during the dynamical behavior of the cellular neural network.

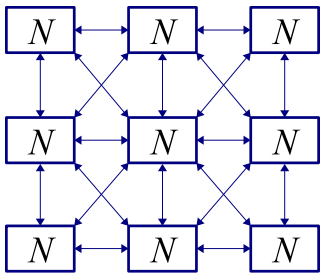


Figure 1.4: Cellular neural networks.

Thanks to the following features neural networks are suitable for black-box system identification [17, 23, 24, 25, 26, 27, 28]: approximation capability of nonlinear functions via activation functions, ability to process many inputs and outputs and the synaptic weights are automatically adjusted by a learning algorithm during the training.

For system identification and control, the architectures the most used are feedforward and recurrent neural networks. In these type of configurations, the neurons are usually arranged in layers. For example, Fig. 1.5 shows a three layers perceptron where the neurons are located in layers sequentially connected. Each layer is numbered (0, 1, 2 or 3). The layer 0, commonly known like input layer, feeds the layer 1 with the input signals, the layer 1 and the layer 2 are called hidden layers, the output of each neuron in the layer 1 are the inputs of the neurons in the layer 2. Finally, the outputs of the neurons of the layer 2 are the inputs of the neurons in the layer 3 commonly known as output layer. The neurons in the hidden layer are called hidden neurons, and the neurons in the input and output layers are called input and output neurons respectively. In the sequel, in order to refer to the number of neurons in each layer, the following notation is used n_1-n_2-$\cdots$-n_i, where $n_i - 1$ is the number of hidden layers, n_1 is the number of neurons in the first hidden layer, n_2 is the number of neurons in the second hidden layer and n_i is the number of neurons in the output layer.

As already mentioned, the number of layers plays an important role in neural networks design. While the number of layers grow up, the approximation capabilities are increased, but the model complexity and the computational cost increase as well. The number of neurons in each layer has the same influence as the number of layers on the quality and the price of the model. Therefore, these two aspects are considered in this work.

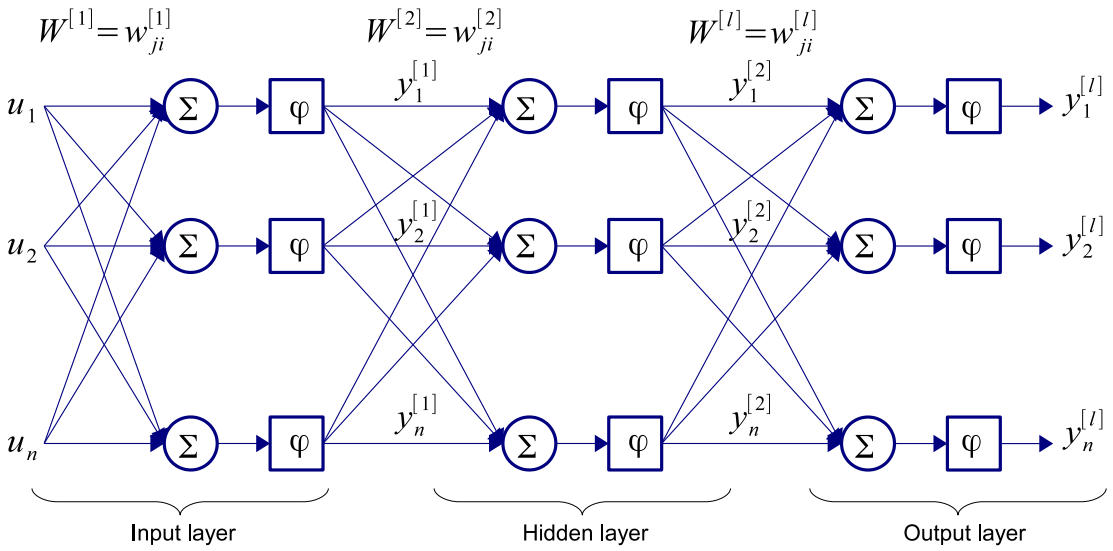


Figure 1.5: Three layers perceptron.

1.3 System identification using neural network: a general procedure

A general system identification procedure using neural networks is introduced in this section. Two important steps in a general system identification method using neural networks are: the neural network structure selection and the neural network training.

1.3.1 Neural networks structure selection

The structure selection is one of the most important step in the system identification procedure. An important choice is the type of neural network architecture. As already mentioned, feedforward and recurrent neural networks are the types of architectures the most used for system identification. Due to its advantages over the feedforward architectures, RNN are used in the sequel for modeling dynamics systems.

Once the type of architecture is selected, important characteristics of a neural network architecture are the number of layers, the number of neurons and the activation functions. Usually these choices are based on a previous knowledge of the system. Since we are focused in a black box approach, these choices depends on some criterion defined by the designer.

Intuitively, the number of layers and the number of neurons should be chosen in order to find a good balance between the accuracy and the complexity of the obtained model. It was formally confirmed that neural networks with at least one hidden layer, and whenever they are defined with a sufficient number of neurons, are able to approximate a large class of systems within a small error margin [72, 73]. Therefore, the most common practice is to manipulate a three layers neural network [21, 74, 75]. Even though, in order to improve the accuracy of their models, some authors use more than three layers [76] regardless the complexity of the model. However, when the simplicity is preferred, two layers are acceptable. In the sequel, a method which provides ready to use models, combining the simplicity of a two layer architecture and the accuracy of

a three layers neural network is proposed. Nevertheless, different works have been achieved in order to find the "optimal" number of neurons [3, 5, 21, 28, 29, 30, 33, 40, 44, 46, 50, 51, 52, 53, 54, 57, 60, 61, 62, 63]. The main constraint of these works, is the excessive requirement of time to find the most convenient number of neurons. Moreover a rather subjective criterion is always used to decide when the trade-off between the model accuracy and the model complexity is appropriate and sufficient. Here we propose a method where the number of neurons chosen does not affect the complexity of final models.

The activation function selection plays an important role in the approximation capabilities and the complexity of a model. By different combinations of activations function in a neural network architecture, several models can be derived [67, 77, 78, 79]. The combination of activation functions, the most encountered in the literature is linear functions in the input and output neurons and nonlinear functions in the hidden neurons [80, 81, 82, 83]. The nonlinear activation functions are classically chosen as $\tanh(\bullet)$ [84] because it is a saturation type and its derivative can be expressed as a simple function of its output $(1-\tanh^2(\bullet))$ [17, 23, 24, 25]. As it will be presented in the following section, this derivative is required for any gradient-based optimization technique (see Appendix B).

Once the model structure is chosen, the next step is the adaptation of the synaptic weights (parameters of the model). This is achieved during the training phase or model computation phase.

1.3.2 Neural network training

The neural network training is realized in accordance to the procedure illustrated in Fig. 1.6. The chosen neural network receives the pass inputs $[u(k-1), ..., u(k-n_b)]$ and the pass outputs $[y(k-1), ..., y(k-n_a)]$ of the real system and generates the actual estimated output $(\hat{y}(k))$. This output value is subtracted from the actual measured output $(y(k))$, to compute the prediction error $(e(k))$ in the k_{th} iteration. The synaptic weights are adapted by a learning algorithm depending on this prediction error.

The learning algorithm is an optimization algorithm adapting the synaptic weights of a defined neural network architecture. The adaptation is conducted each iteration (k) in order to minimize an objective function (MSE, LSAD, Huber's function, etc). In effect, if the data set used for the identification of a system is corrupted, robust functions such as the Huber's function is recommended regardless the computational cost required. If the data set is not really contaminated, a quadratic criterion mays lead to a good accuracy, i.e. estimation with no or minor bias.

More precisely, let us suppose a neuro-model parametrized by the synaptic weights w. The special case of the quadratic criterion is given by (1.2).

$$E(w, u^N, y^N) = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{2}e^2(k, \hat{w}) \tag{1.2}$$

where N is the total number of data used for the parameters estimation and the prediction error $e(k, \hat{w})$ is:

$$e(k, \hat{w}) = y(k) - \hat{y}(k, \hat{w}) \tag{1.3}$$

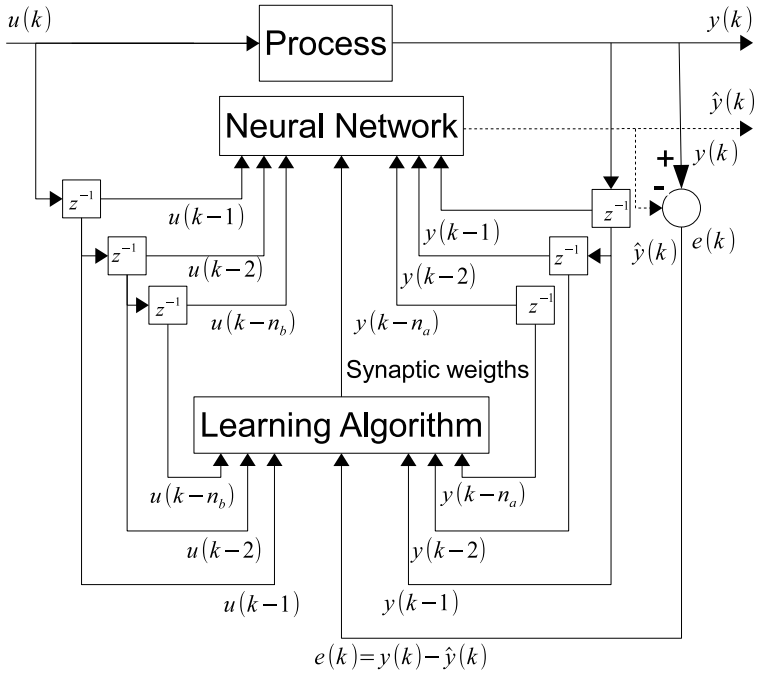


Figure 1.6: Neural network training.

A general family of algorithms to adapt the synaptic weights (w) is given by (1.4).

$$\hat{w}(k+1) = \hat{w}(k) - \eta[R]^{-1}\frac{\partial E}{\partial \hat{w}} \tag{1.4}$$

where R modifies the search direction, η is the step size known as leaning rate, $\frac{\partial E}{\partial \hat{w}}$ is the gradient of the objective function and $\hat{w}(k)$ denotes the value of $\hat{w}$ in the k_{th} iteration.

Different algorithms (Steepest descent, Gauss-Newton, Levenberg-Marquardt, etc) can be derived from (1.4) by conveniently choosing R (the search direction) (See Appendix B). The simplest case where $R = 1$ yields one of the simplest algorithm used in neural network theory, namely steepest descent algorithm (1.5):

$$\hat{w}(k+1) = \hat{w}(k) - \eta\frac{\partial E}{\partial \hat{w}} \tag{1.5}$$

where k increases (from 1 to N) each time a pair of data (from the set used for the training phase "training dataset") is processed. In order to provide more time to the learning algorithm to compute the optimal values of the synaptic weights w, the neural network training is realized in epochs γ. One epoch is elapsed (that is, $\gamma = \gamma + 1$) each time $k = N$. Once one epoch is elapsed, the training data set is used again, that is $k = 1, 2, \cdots, N$.

The number of epochs γ^* required for the estimation of the synaptic weights is chosen on the base of the mean square error (1.2) computed each epoch is elapsed.

For example, from Fig. 1.7, $\gamma = 40$ is a good choice, because after 40 epochs the quadratic error has not further significant reduction.

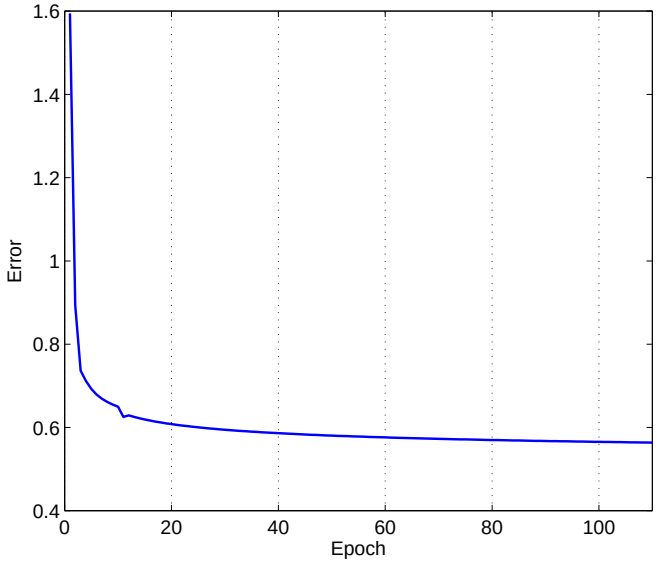


Figure 1.7: Choice of number of epochs.

In (1.5), the parameter η, commonly referred to as the learning rate or integration step size, plays an important role in the parameters adaptation in particular the convergence speed of the algorithm. It should be pointed out that for the discrete time steepest descent algorithm, η should be bounded in a small range, to nevertheless ensure the stability of the algorithm. A small η means that the convergence to a solution is slow while a large η means a faster convergence where oscillations may occur and stability may be lost.

In order to improve the learning algorithm, different solutions for the adaptation of η have been implemented in this work. For example, a modification of the "search then convergence algorithm" used in [23] is given by (1.6).

$$\eta(n) = \eta_0 \frac{1}{1 + \frac{n}{k_0 \gamma}} \tag{1.6}$$

where:

η_0 is the initial value of η proposed by the user.
$k_0 = \frac{10N}{3}$.
$n = 1, 2, \cdots, (N \times \gamma)$ is the iteration increasing during all the training phase (all the epochs).
$\gamma = 1, 2, \cdots, \gamma*$ is a the number of epochs elapsed.
N is the number of data used for the neural network training.
γ^* is a "optimal" number of epochs required for the adaptation of the model parameters.

Every time that one epoch has elapsed $\eta_0 = \eta(\gamma)$ and $\gamma = \gamma + 1$.

Fig. 1.8 illustrates, the dynamic of η during the complete training phase (that is, γ^* epochs have elapsed).

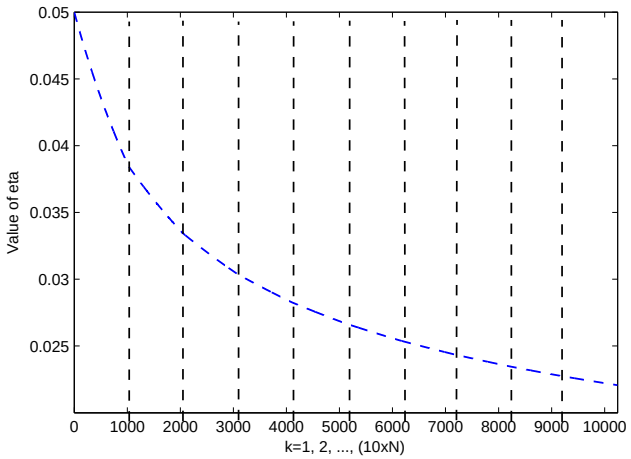


Figure 1.8: Learning coefficient during all the epochs ($\eta(n)$).

From (1.6) we can deduce that when n increases the value of the learning coefficient $\eta(k)$ decreases (see Fig. 1.8). It is desirable because at the beginning of the training the synaptic weights are supposed far of their optimal values and they should be adapted as fast as possible ($\eta = \eta_0$). When a period of time is over ($k \rightarrow (N \times \gamma)$) the synaptic weights are supposed near their optimal values. Then, the parameters should be adapted slowly ($\eta \rightarrow 0$) in order to improve the convergence.

Once all the elements required to understand the different procedures for nonlinear system identification using neural networks have been discussed, let us now present an illustrative example.

1.3.3 Illustrative example

The first step in system identification using neural networks is to choose and define the structure of the model. In order to explain the development of the adaptation algorithms of the synaptic weights, let us introduce a neural network architecture as an example.

Fig. 1.9 shows a two layers neural network with nn neurons in the first layer and one neuron in the output layer. It is interesting to notice that the number of neurons in the first layer (nn neurons used to process the regressors input-output vectors) is chosen by the user. This offers to the user a possibility for choosing a balanced simplicity-accuracy structure.

The mathematical representation of such architecture is given by (1.7).

$$\begin{aligned} \hat{y}(k) &= \varphi_3(T) \\ T &= \sum_{p=1}^{nn} z_p \varphi_1(r_p) \\ r_p &= \sum_{i=1}^{n_b} (w_{p,i} u(k-i)) + \sum_{j=1}^{na} (w_{p,i+j} y(k-j)) \end{aligned} \tag{1.7}$$

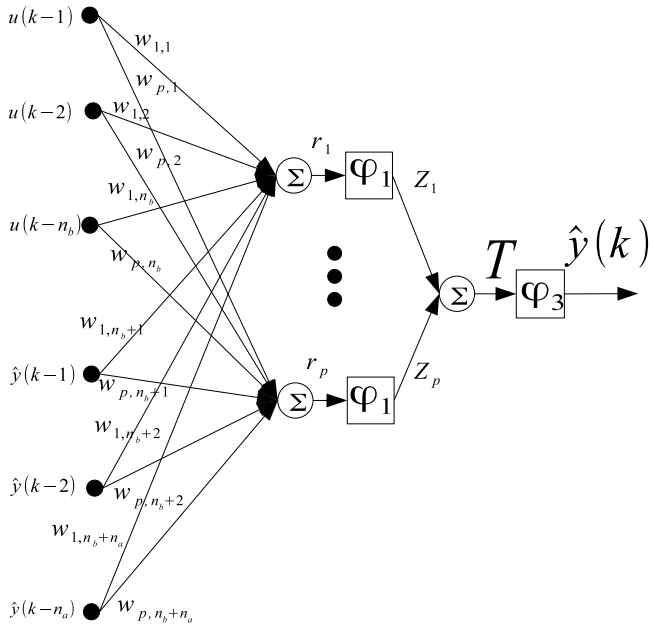


Figure 1.9: Recurrent nn-1 neural network.

where:

$u(k-i)$, $y(k-j)$ ϵR^1
$w_{p,q}$, z_p ϵR^1 are the synaptic weights.
$u(k-i)$ is the input delay i times.
$y(k-j)$ is the input delay j times.

As already mentioned, different models could be generated by different combination of activations functions. Let us start with the simplest of the models, that is, all the activation functions linear $\varphi_1(x) = \varphi_3(x) = x$, then the model given by (1.7) becomes:

$$\begin{aligned} \hat{y}(k) &= T \\ T &= \sum_{p=1}^{nn}(z_p r_p) \\ r_p &= \sum_{i=1}^{n_b}(w_{p,i}u(k-i)) + \sum_{j=1}^{na}(w_{p,i+j}y(k-j)) \end{aligned} \tag{1.8}$$

With linear activation functions in the first layer ($\varphi_1(x) = x$) and the activation function in the output layer nonlinear $\varphi_3(x) = nonlinear$, the model given by (1.7) becomes:

$$\begin{aligned} \hat{y}(k) &= \varphi_3(T) \\ T &= \sum_{p=1}^{nn} z_p(r_p) \\ r_p &= \sum_{i=1}^{n_b}(w_{p,i}u(k-i)) + \sum_{j=1}^{na}(w_{p,i+j}y(k-j)) \end{aligned} \tag{1.9}$$

A different model can be contained by choosing the activation functions in the first layer nolinear ($\varphi_1(x) = nonlinear$) and the activation function in the output layer linear $\varphi_3(x) = x$, the model given by (1.7) becomes:

$$\begin{aligned} \hat{y}(k) &= T \\ T &= \sum_{p=1}^{nn} z_p \varphi_1(r_p) \\ r_p &= \sum_{i=1}^{n_b} (w_{p,i} u(k-i)) + \sum_{j=1}^{na} (w_{p,i+j} y(k-j)) \end{aligned} \tag{1.10}$$

To be more demonstrative, let us choose the model given by (1.10). With the nonlinear activation functions classically chosen as $\varphi_1(x) = \tanh(x)$ and $n_b = 2$ delays in the input layer, $n_a = 1$ delay in the output layer and $nn = 2$ neurons in the first layer, this model becomes:

$$\begin{aligned} &\hat{y}(k) = T \\ &T = z_1 \tanh(r_1) + z_2 \tanh(r_2) \\ &r_p = w_{1,1} u(k-1) + w_{1,2} u(k-1) + w_{1,3} y(k-1) + w_{2,1} u(k-1) + w_{2,2} u(k-1) + w_{2,3} y(k-1) \end{aligned} \tag{1.11}$$

or

$$\begin{aligned} \hat{y}(k) &= T \\ T &= \sum_{p=1}^{2} z_p \tanh(r_p) \\ r_p &= \sum_{q=1}^{n_a+n_p} (w_{p,q} J(q)) \end{aligned} \tag{1.12}$$

where q is the q_{th} element of the vector $J = [u(k-1) \quad u(k-2) \quad y(k-1)]$.

Notice that in the preceding model, by increasing the number of neurons nn or the number of delays in the input n_b or the output n_a the number of parameters increase as well, so as the model complexity.

Once the neuro-model is totally defined, the following step is the synaptic weights adaptation during the training phase. Remember that in this step, an objective function has to be minimized. For simplicity, in this example the objective function used is the MSE given by (1.13).

$$E(w_{p,q}, z_p) = \frac{1}{N} \sum_{k=1}^{N} \frac{1}{2} e^2(k, w_{p,q}, z_p) \tag{1.13}$$

where the prediction error in the k_{th} iteration is $e(k) = y(k) - \hat{y}(k)$.

The rules for the adaptation of the synaptic weights $w_{p,q}$ and z_p, based on the steepest descent algorithm, which is the simplest of the gradient-based algorithms are:

$$\hat{w}_{p,q}(k+1) = \hat{w}_{p,q}(k) - \eta \frac{\partial E}{\partial \hat{w}_{p,q}} \tag{1.14}$$

$$\hat{z}_p(k+1) = \hat{z}_p(k) - \eta \frac{\partial E}{\partial \hat{z}_p} \tag{1.15}$$

where η is adapted by the algorithm (1.6) and the partial derivatives $\frac{\partial E}{\partial \hat{w}_{p,q}}$ and $\frac{\partial E}{\partial \hat{z}_p}$ are computed according the chain rule as follows.

$$\frac{\partial E}{\partial \hat{w}_{p,q}} = \frac{\partial E}{\partial \hat{e}} \frac{\partial e}{\partial \hat{\hat{y}}} \frac{\partial \hat{y}}{\partial T} \frac{\partial T}{\partial r_p} \frac{\partial r_p}{\partial \hat{w}_{p,q}} \tag{1.16}$$

$$\frac{\partial E}{\partial \hat{z}_p} = \frac{\partial E}{\partial \hat{e}} \frac{\partial e}{\partial \hat{\hat{y}}} \frac{\partial \hat{y}}{\partial T} \frac{\partial T}{\partial z_p} \tag{1.17}$$

The partial derivatives for the chosen model (1.12) are:

$$\begin{aligned}
&\frac{\partial E}{\partial \hat{e}} = e(k) \\
&\frac{\partial e}{\partial \hat{\hat{y}}} = -1 \\
&\frac{\partial \hat{y}}{\partial T} = 1 \\
&\frac{\partial T}{\partial r_p} = z_p sech^2(r_p) \\
&\frac{\partial r_p}{\partial \hat{w}_{p,q}} = J(q) \\
&\frac{\partial T}{\partial z_p} = \tanh(r_p)
\end{aligned}$$

By substitution of the partial derivatives in (1.14) and (1.15) we obtain:

$$\hat{w}_{p,q}(k+1) = \hat{w}_{p,q}(k) + \eta e(k) z_p sech^2(r_p) J(q) \tag{1.18}$$

$$\hat{z}_p(k+1) = \hat{z}_p(k) + \eta e(k) \tanh(r_p) \tag{1.19}$$

It is interesting here to notice that the computational bulk of the model estimation depends on the choice of the learning algorithm, the neural network architecture (number of neurons, number of layers, delays in the input and delays in the output) and the criterion function.

We have presented and discussed the main issues for an efficient and complete neural network identification methodology for complex and nonlinear systems. Now, we have to present the particular neural networks design which allows us the reduction of the number of parameters and the reduction in the computational cost. In this sense we shall contribute to investigate the crux problem of the good compromise between complexity, quality and cost of the neural network estimation.

Chapter 2

A new identification oriented neural network design

In this chapter, one of the proposed neural networks which allows us the main contributions of this work is presented. This architecture, presented in the Section 2.1, is based on a three layers recurrent neural network with a variable number of neurons in the first layer (2nn-2-1 recurrent neural network). The particular configuration of such neural network, allow us to transform (after the training phase) its structure into the simpler 2-1 architecture (see Section 2.2) preserving the accuracy of the former 2nn-2-1 architecture. The computational cost required to train the proposed neural network is reduced into the one required to train the 2-2-1 architecture presented in Section 2.3. Let us now present the different structures neural networks an their main goal.

2.1 Recurrent 2nn-2-1 neural network: accuracy

Fig. 2.1 shows a three layers neural network with $2 * nn$ neurons in the input layer, two neurons in the hidden layer and one neuron in the output layer. This architecture allows us to define the desired level of the model accuracy.

It is interesting to notice that the number of neurons in the hidden layer is fixed and the number of neurons in the input layer (nn neurons used to process the regressors input vector and nn neurons used to process the regressors output vector) is chosen by the user. This special configuration allows us to reduce the 2nn-2-1 neural network into the 2-1 architecture shown in Fig. 2.2 keeping the same approximation accuracy of the original non reduced model. Even if this structure is somehow particular, by different combination of activation functions, the proposed architecture permits us to generate easily the classical models presented in ([6]). Therefore, at a user point of view, this architecture remains sufficiently general to cover almost all its practical needs. The mathematical representation of the proposed architecture is given by (2.1).

$$\begin{aligned} \hat{y}(k) &= X\varphi_3(T) \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) + Z_h \\ r_b &= \sum_{i=1}^{nn} V_{b_i}\varphi_1(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}\varphi_1(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{2.1}$$

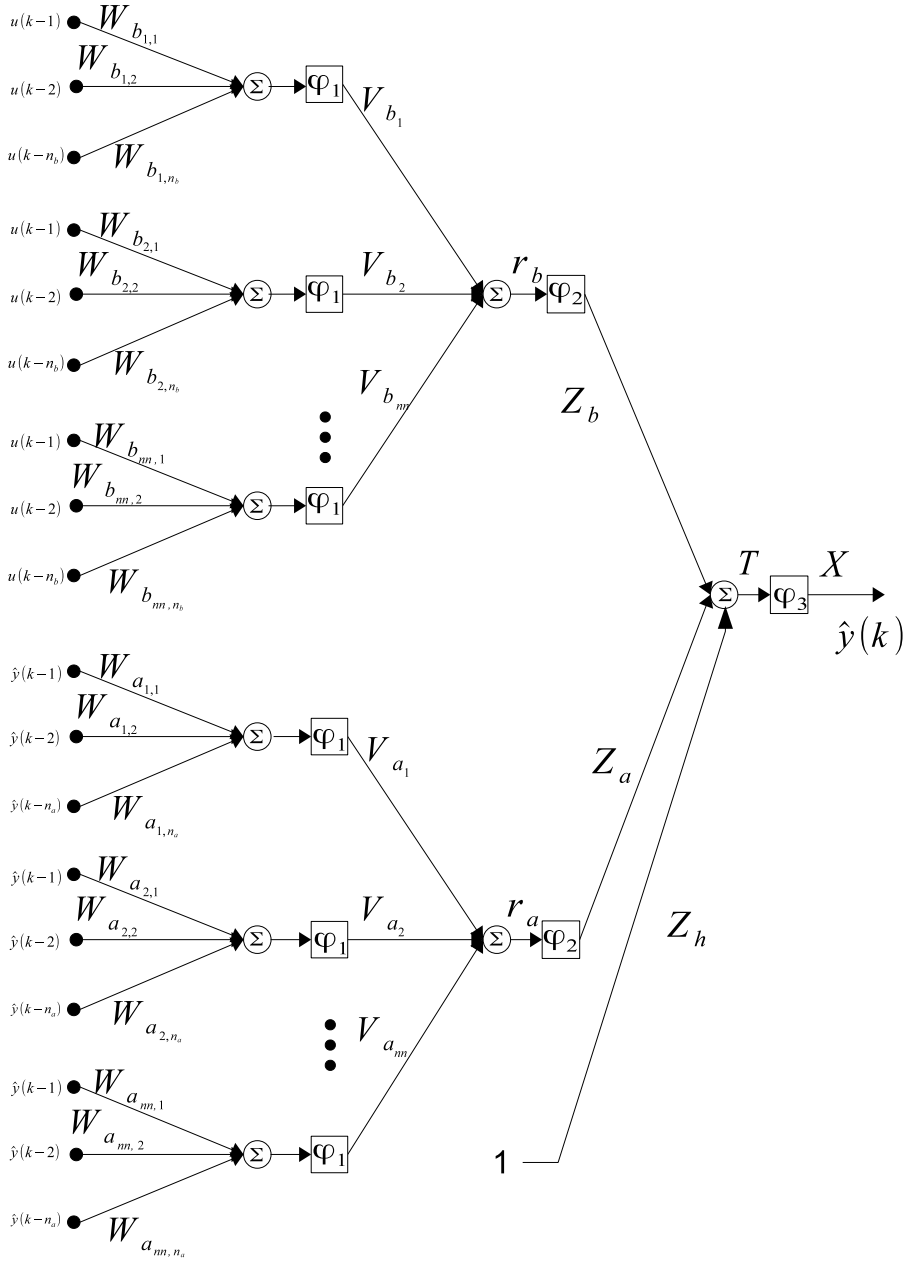


Figure 2.1: Recurrent 2nn-2-1 neural network.

where:

$J_u = [u(k-1) \quad u(k-2) \quad ... \quad u(k-n_b)] \quad \epsilon \quad R^{1\times n_b}$
$J_{\hat{y}} = [\hat{y}(k-1) \quad \hat{y}(k-2) \quad ... \quad \hat{y}(k-n_a)] \quad \epsilon \quad R^{1\times n_a}$
$W_{b_i} = [W_{b_{i,1}} \quad W_{b_{i,2}} \quad ... \quad W_{b_{i,n_b}}]^\top \quad \epsilon \quad R^{n_b\times 1}$
$W_{a_i} = [W_{a_{i,1}} \quad W_{a_{i,2}} \quad ... \quad W_{a_{i,n_a}}]^\top \quad \epsilon \quad R^{n_a\times 1}$
$X,\ Z_b,\ Z_a,\ V_{b_i},\ V_{a_i},\ Z_h\ \epsilon R^1$
$X,\ Z_b,\ Z_a,\ V_{b_i},\ V_{a_i},\ W_{b_i},\ W_{a_i}$, and Z_h are synaptic weights.
where $i = 1, 2, \cdots, nn$ and nn is the number of neurons.

Different models can be derived from this architecture by different combinations of activation functions as it was presented in the illustrative example in Chapter 1. Let us denote four different families of models developed in this thesis.

1. FF1H model:

By selecting $\varphi_3(z) = \varphi_2(z) = z$ in (2.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b r_b + Z_a r_a + Z_h \\ r_b &= \sum_{i=1}^{nn} V_{b_i} \varphi_1(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i} \varphi_1(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{2.2}$$

Moreover, from this model we can derive models where the dependence on the past values of the input is linear and dependence on the past values of the output is nonlinear. These kind of models is particularly suited for nonlinear control problem [6, 65, 29, 66].

2. FF2H model:

By selecting $\varphi_3(z) = \varphi_1(z) = z$ in (2.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b \varphi_2(r_b) + Z_a \varphi_2(r_a) + Z_h \\ r_b &= \sum_{i=1}^{nn} V_{b_i} (J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i} (J_{\hat{y}} W_{a_i}) \end{aligned} \tag{2.3}$$

In the same way as for the model (2.2), from (2.3) we can derive models where the dependence on the past values of the input is linear and dependence on the past values of the output is nonlinear.

3. ARXH model:

By selecting $\varphi_1(z) = \varphi_2(z) = \varphi_3(z) = z$ in (2.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b r_b + Z_a r_a + Z_h \\ r_b &= \sum_{i=1}^{nn} V_{b_i} (J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i} (J_{\hat{y}} W_{a_i}) \end{aligned} \tag{2.4}$$

The interest on these kind of models is that is a classical linear structure.

4. NLARXH model:

By selecting $\varphi_1(z) = \varphi_2(z) = z$ in (2.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= X\varphi_3(T) \\ T &= Z_b r_b + Z_a r_a + Z_h \\ r_b &= \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{2.5}$$

The reader shall notice that, these four models found in the literature [6, 85, 65, 39, 29, 66], represent a fairly large class of systems [85].

2.2 Recurrent 2-1 neural network: simplicity

Now we shall present the final architecture that the reduction procedure provides. Fig. 2.2 shows a two layers neural network with 2 neurons in the input layer (*one* neuron used to process the regressors input vector and *one* neuron used to process the regressors output vector). In fact, this architecture defines the desired complexity of the final model that the proposed system identification method must provide.

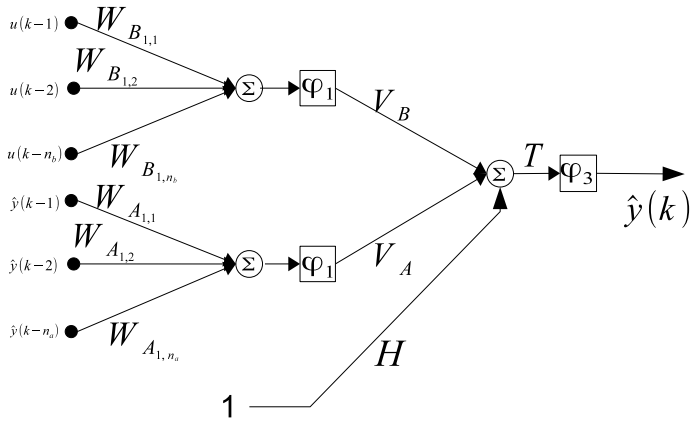


Figure 2.2: Recurrent 2-1 neural network.

The mathematical representation of such neural network architecture is given by (2.6).

$$\begin{aligned} \hat{y}(k) &= \varphi_3(T) \\ T &= V_B\varphi_1(J_u W_B) + V_A\varphi_1(J_{\hat{y}} W_A) + H \end{aligned} \tag{2.6}$$

where:

$$\begin{aligned} &W_B = [W_{B_{1,1}} \quad W_{B_{1,2}} \quad ... \quad W_{B_{1,n_b}}]^\top \quad \epsilon \quad R^{n_b \times 1} \\ &W_A = [W_{A_{1,1}} \quad W_{A_{1,2}} \quad ... \quad W_{A_{1,n_a}}]^\top \quad \epsilon \quad R^{n_a \times 1} \\ &V_B,\ V_A \text{ and } H\ \epsilon R^1 \end{aligned}$$

It is interesting to notice that this simple 2-1 architecture defining the complexity of the final model, keeps the same accuracy than the 2nn-2-1 complex neural network.

2.3 Recurrent 2-2-1 neural network: computational cost

Finally, let us present the architecture that helps us to define the desired computational cost our approach has to satisfy, generating ready to use models.

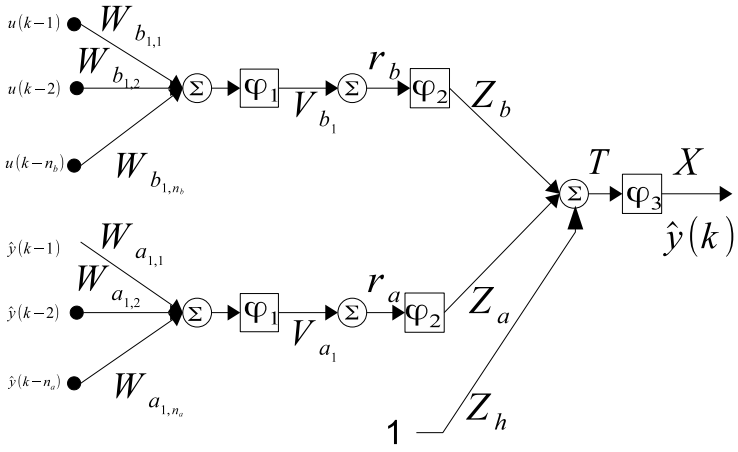


Figure 2.3: Recurrent 2-2-1 neural network.

Fig. 2.3 shows a three layers recurrent neural network with two neurons in the first layer, 2 neurons in the second layer and one neuron in the output layer. The mathematical representation of such neural network is given by (2.7).

$$\begin{aligned} \hat{y}(k) &= X\varphi_3(T) \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) + Z_h \\ r_b &= nn \times V_{b_1}\varphi_1(J_u W_{b_1}) \\ r_a &= nn \times V_{a_1}\varphi_1(J_{\hat{y}} W_{a_1}) \end{aligned} \tag{2.7}$$

where:

$$\begin{aligned} &W_{b_1} = [W_{b_1} \quad W_{b_2} \quad ... \quad W_{b_{n_b}}]^\top \quad \epsilon \quad R^{n_b \times 1} \\ &W_{a_1} = [W_{a_1} \quad W_{a_2} \quad ... \quad W_{a_{n_a}}]^\top \quad \epsilon \quad R^{n_a \times 1} \\ &X,\ Z_b,\ Z_a,\ V_{b_1},\ V_{a_1},\ Z_h\ \epsilon R^1 \end{aligned}$$

It is interesting to notice that the 2nn-2-1 model, defining the accuracy of the final model (Eq. 2.1), contain more parameters than the 2-2-1 architecture defining the computational cost (Eq. 2.7). This ensure the interest on training the simplest model which required a lower computational cost. Moreover, thanks to the design conditions presented in the following chapter, both models, (2.7) and (2.1) are equivalents.

Above, we have define the main performances in terms of accuracy, complexity and computational cost which the proposed nonlinear system identification methodology aims to satisfy. More precisely, this results constitute the main contribution of this work: generate a model

combining the simplicity of the neural network of Fig. 2.2 (Eq. (2.6)), the approximation capabilities of the architecture shown in Fig. 2.1 (Eq. (2.1)) and the computational cost of the 2-2-1 architecture shown in Fig. 2.3. This is achieved by two reduction approaches, one reducing the number of parameters and the second reducing the computational cost of the 2nn-2-1 architecture. Both reduction approaches are achieved under two reasonable assumptions as we will see in the following chapter.

Chapter 3

Formal issues: The reduction procedures

The main contributions of this thesis, already published in [68, 69, 70], are presented in this chapter. The proposed model complexity reduction approach allows us to reduce the number of parameters of a 2nn-2-1 architecture into the number of parameters of a 2-1 neural network. A computational cost reduction approach, allows us to reduce the computational bulk required to train a 2nn-2-1 architecture into the one required to train a 2-2-1 neural network. In fact, we avoid the cost corresponding to the training of the complex architecture without loss of accuracy.

3.1 Validity assumptions

These two reduction procedures provide models of relevant quality and low price, under the following two simple assumptions whose purpose is to achieve two design conditions. The first one is a neural architecture design condition and the second one is a training design condition. The reader shall notice that Assumption 2, represents the originality of this work.

Assumption 1: At least one layer should have all its activation functions chosen as linear, that is, $\varphi_1(T) = T$ or $\varphi_2(T) = T$ or $\varphi_3(T) = T$ in Fig. 2.1.

The reader shall notice that Assumption 1 is not very restrictive, since it is a classical way to choose the activation functions in neural networks.

Assumption 2: The designer should select the initial condition of the synaptic weights equals group by group, i. e., $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{b_1}(0) = W_{b_j}(0)$ and $W_{a_1}(0) = W_{a_j}(0)$ with $j = 2, 3, \cdots, nn$.

Even if this is not a classical way to choose the initial conditions of the synaptic weights, full experiments, detailed in Chapter 5 and presented in [68], will definitely convince the user of their validity.

3.2 Model complexity reduction approach

Theorem 1: Consider the neural network whose architecture 2nn-2-1 is expressed in (2.1) and depicted in Fig. 2.1, if the assumption 1 and the assumption 2 are fulfilled, then such neural network can be reduced into a 2-1 equivalent architecture (see Fig. 2.2).

Proof. Let us consider the architecture of Fig. 2.1 corresponding to a three layers neural network, with the input-output mapping given by (2.1). For better understanding we decide to

divide the model transformation in the two following steps.

Step 1.- Neural network training under the proposed assumptions.

In order to satisfy the assumption 1 in the theorem, let us select $\varphi_3(z) = \varphi_2(z) = z$ and $\varphi_1(z) = \tanh(z)$ in (2.1). It comes:

$$\begin{aligned} \hat{y}(k) &= X(T) \\ T &= Z_b(r_b) + Z_a(r_a) + Z_h \\ r_b &= \sum_{i=1}^{nn} V_{b_i} \tanh(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i} \tanh(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{3.1}$$

According to assumption 2, $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{b_1}(0) = W_{b_j}(0)$ and $W_{a_1}(0) = W_{a_j}(0)$ with $j = 2, 3, \cdots, nn$.

The adaptation laws of the synaptic weights, derived according to the steepest descent algorithm (see Appendix B) are:

$$X(k+1) = X(k) + \eta e(k) T \tag{3.2}$$
$$Z_h(k+1) = Z_h(k) + \eta e(k) X \tag{3.3}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \tag{3.4}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \tag{3.5}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X Z_b \tanh(J_u W_{b_i}) \tag{3.6}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X Z_a \tanh(J_{\hat{y}} W_{a_i}) \tag{3.7}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X Z_b V_{b_i} sech^2(J_u W_{b_i}) J_u \tag{3.8}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X Z_a V_{a_i} sech^2(J_{\hat{y}} W_{a_i}) J_{\hat{y}} \tag{3.9}$$

where $i = 1, 2, \cdots, nn$.

Once the neural network model given by (3.1) is trained under these 2 assumptions, we obtain:

$$\begin{aligned} \hat{y}(k) &= X^* T \\ T &= Z_b^* r_b + Z_a^* r_a + Z_h^* \\ r_b &= \sum_{i=1}^{nn} V_{b_i}^* \tanh(J_u W_{b_i}^*) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}^* \tanh(J_{\hat{y}} W_{a_i}^*) \end{aligned} \tag{3.10}$$

Since the initial condition of the synaptic weights are chosen equals group by group (see Assumption 2) and each group (V_{b_i}, V_{a_i}, W_{b_i} and W_{a_i}) is trained by the same adaptation rule (see (3.6), (3.7), (3.8) and (3.9) respectively), the final values of the synaptic weights are: $V_{b_1}^* = V_{b_j}^*$, $V_{a_1}^* = V_{a_j}^*$, $W_{b_1}^* = W_{b_j}^*$ and $W_{a_1}^* = W_{a_j}^*$ with $j = 2, 3, \cdots, nn$.

Step 2.- Model transformation.

Let us remember that the synaptic weights are computed during the neural network training. We can now develop the model transformation.

Layers reduction. In (3.10), it is indeed possible to make the following algebraic operations:

$$\begin{aligned} H^* &= X^* \times Z_h^* \\ V_{B_i}^* &= X^* \times Z_b^* \times V_{b_i}^* \\ V_{A_i}^* &= X^* \times Z_a^* \times V_{a_i}^* \end{aligned}$$

where $V_{B_1}^* = V_{B_j}^*$ and $V_{A_1}^* = V_{A_j}^*$ (with $j = 2, 3, \cdots, nn$) due to Assumption 2 (see Step 1).

Then, the three layers model (Fig. 2.1, Eq. (3.10)) is redefined as a two layers neural network (Fig. 3.1, Eq. (3.11)). It is important to remark that the model is always the same (we only change its notations in order to reduce the number of parameters), thus the neural network keeps its approximation accuracy.

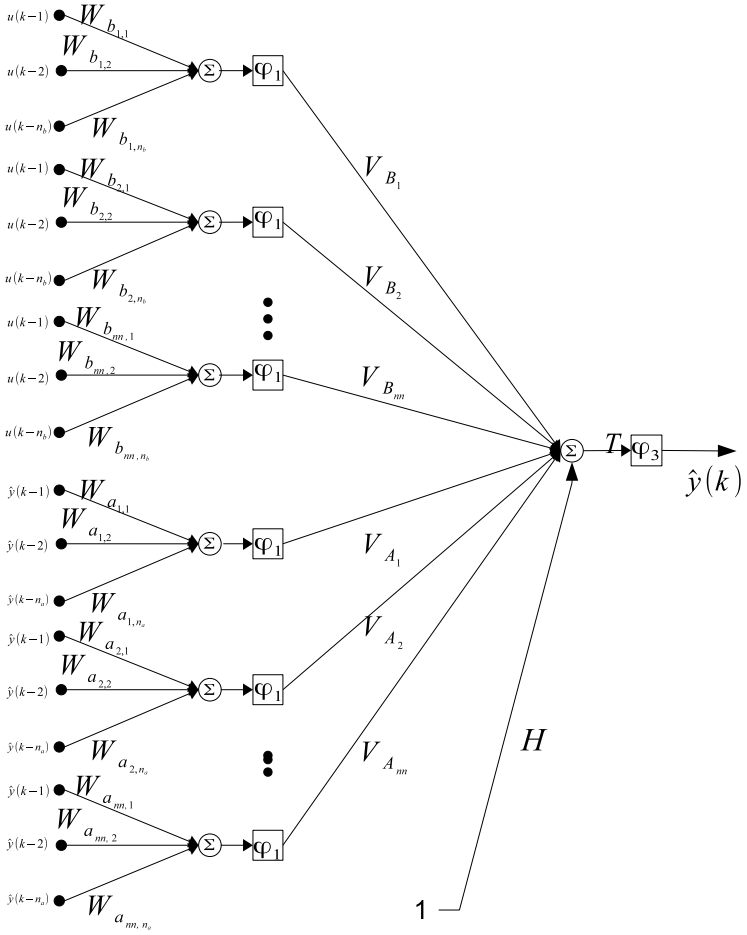


Figure 3.1: Recurrent 2nn-1 neural network.

$$\begin{aligned}
&\hat{y}(k) = T \\
&T = r_b + r_a + H^* \\
&r_b = \sum_{i=1}^{nn} V_{B_i}^* \tanh(J_u W_{b_i}^*) \\
&r_a = \sum_{i=1}^{nn} V_{A_i}^* \tanh(J_{\hat{y}} W_{a_i}^*)
\end{aligned} \tag{3.11}$$

Neurons reduction. A supplementary transformation is achieved in order to change the 2nn-1 neural network, containing $2 \times nn$ neurons in the first layer (Fig. 3.1), into a model of 2 neurons in the first layer (Fig. 2.2).

From Assumption 2 and after "Layers reduction" we have $V_{B_1}^* = V_{B_j}^*$, $V_{A_1}^* = V_{A_j}^*$, $W_{b_1}^* = W_{b_j}^*$ and $W_{a_1}^* = W_{a_j}^*$ (with $j = 2, 3, \cdots, nn$) in (3.11).

The following algebraic operations can be done:

$$\begin{aligned}
&\sum_{i=1}^{nn} V_{B_i}^* \tanh(J_u W_{b_i}^*) = V_B^* \tanh(J_u W_B^*) \\
&\sum_{i=1}^{nn} V_{A_i}^* \tanh(J_{\hat{y}} W_{a_i}^*) = V_A^* \tanh(J_{y_n} W_A^*)
\end{aligned}$$

where:

$$\begin{aligned}
&V_B^* = nn \times V_{B_i}^* \\
&V_A^* = nn \times V_{A_i}^* \\
&W_B^* = W_{b_1}^* \\
&W_A^* = W_{a_1}^*
\end{aligned}$$

The resulting model after the "neurons reduction" has the following mathematical form:

$$\hat{y}(k) = V_B^* \tanh(J_u W_B^*) + V_A^* \tanh(J_{\hat{y}} W_A^*) + H^* \tag{3.12}$$

Finally, this shows that, by applying the step 1 and the step 2, the complex 2nn-2-1 model (Fig. 2.1) is reduced to a 2-1 model (Fig. 2.2) with the same accuracy as the complex model. This completes the proof. ■

Remark 3: It is important to insist in the fact that (3.12) is entirely equivalent to the complex model (3.10). Consequently, these two equivalent models have the same accuracy. And no computational cost is carried out during the step 2, that represents the main interest of our solution.

Remark 4: Notice that the 2nn-2-1 model (3.1) has $(nn \times (n_b)) + (nn \times (n_a))$ synaptic weights in the first layer, the number of synaptic weights in the second layer is $(2 \times nn)$ and the third

layer has only one neuron with 4 synaptic weights. As a result, the neural network has $(((nn \times (n_b)) + (nn \times (n_a))) + (2 \times (nn)) + (4))$ synaptic weights. However the 2-1 reduced neuro model (3.12) which yields from theorem 1, has $(n_b + n_a)$ synaptic weights in the first layer and 3 synaptic weights in the second layer. Consequently, the reduced 2-1 neural network has $((n_a + n_b) + 3)$ synaptic weights. In order to illustrate the interest of the application of this theorem, let us consider the experiment 1 that will be developed in Section 5.1.2 where we train the 2nn-2-1 model (3.1) where $nn = 40$, $n_a = 4$ and $n_b = 7$. This complex model has $((40 \times (7)) + (40 \times (4)) + 2 \times (40) + 4 = 524)$ synaptic weights. Theorem 1 yields a reduced 2-1 neuro model (3.12) with just $((4 + 7) + 3 = 14)$ synaptic weights.

Even if the final model achieved by the proposed model reduction approach has a reduced number of parameters, the computational complexity depends on the training time required to adapt the complex model. More precisely, $(((nn \times n_b) + (nn \times n_a)) + (2 \times nn) + (4))$ synaptic weights should be adapted each iteration. Then, if N iterations are required to adapt the synaptic weights, the computational complexity to generate the model is given by $(O(2nn - 2 - 1) = (((nn \times n_b) + (nn \times n_a)) + (2 \times nn) + (4)) \times N)$. In order to improve the computational cost, the following theorem is proposed.

3.3 Computational cost reduction approach

Theorem 2: Consider the neural network whose architecture 2nn-2-1 is expressed in (2.1).

$$\begin{aligned}
\hat{y}(k) &= X\varphi_3(T) \\
T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) + Z_h \\
r_b &= \sum_{i=1}^{nn} V_{b_i}\varphi_1(J_u W_{b_i}) \\
r_a &= \sum_{i=1}^{nn} V_{a_i}\varphi_1(J_{\hat{y}} W_{a_i})
\end{aligned}$$

If the assumption 1 and the assumption 2 are fulfilled, then the computational cost required to adapt the parameters of such neural network $(O(2nn - 2 - 1) = (((nn \times n_b) + (nn \times n_a)) + (2 \times nn) + (4)) \times N)$ can be reduced into the one required to adapt the parameters of a 2-2-1 equivalent architecture $(O(2 - 2 - 1) = (n_b + n_a + 4) \times N)$.

Proof. Let us consider the 2nn-2-1 neural network with the input-output mapping given by (2.1).

Remember that the synaptic weights ($V_{b_i}^*$, $V_{a_i}^*$, $W_{b_i}^*$ and $W_{a_i}^*$) with $i = 1, 2, \cdots, nn$ of this architecture are adapted by (3.6), (3.7), (3.8) and (3.9) respectively, that is, for each synaptic weight, the computational complexity is given by $O(W_{b_i}) = N \times (nn \times n_b)$, $O(W_{a_i}) = N \times (nn \times n_a)$, $O(V_{b_i}) = N \times (nn)$, $O(V_{a_i}) = N \times (nn)$.

As it has been demonstrated in the proof of the theorem 1, once the neural network model given by (2.1) is trained under Assumption 1 and Assumption 2, we obtain (3.10) where the final values of the synaptic weights are: $V_{b_1}^* = V_{b_j}^*$, $V_{a_1}^* = V_{a_j}^*$, $W_{b_1}^* = W_{b_j}^*$ and $W_{a_1}^* = W_{a_j}^*$ with $j = 2, 3, \cdots, nn$. Since, $V_{b_1}^* = V_{b_j}^*$, $V_{a_1}^* = V_{a_j}^*$, $W_{b_1}^* = W_{b_j}^*$ and $W_{a_1}^* = W_{a_j}^*$. Then we can change the model (3.10) into the following model:

$$\begin{aligned}
&\hat{y}(k) = X^* T \\
&T = Z_b^* r_b + Z_a^* r_a + Z_h^* \\
&r_b = nn \times V_{b_1}^* \tanh(J_u W_{b_1}^*) \\
&r_a = nn \times V_{a_1}^* \tanh(J_{\hat{y}} W_{a_1}^*)
\end{aligned} \tag{3.13}$$

with the synaptic weights (V_{b_1}, V_{a_1}, W_{b_1} and W_{a_1}) adapted as follows:

$$V_{b_1}(k+1) = V_{b_1}(k) + \eta e(k) X Z_b \tanh(J_u W_{b_1}) \tag{3.14}$$

$$V_{a_1}(k+1) = V_{a_1}(k) + \eta e(k) X Z_a \tanh(J_{\hat{y}} W_{a_1}) \tag{3.15}$$

$$W_{b_1}(k+1) = W_{b_1}(k) + \eta e(k) X Z_b V_{b_1} sech^2(J_u W_{b_1}) J_u \tag{3.16}$$

$$W_{a_1}(k+1) = W_{a_1}(k) + \eta e(k) X Z_a V_{a_1} sech^2(J_{\hat{y}} W_{a_1}) J_{\hat{y}} \tag{3.17}$$

Let us notice that the model 2-2-1 given by (3.13) with the synaptic weights X^*, Z_h^*, Z_a^*, Z_b^*, $V_{b_1}^*$, $V_{a_1}^*$, $W_{b_1}^*$ and $W_{a_1}^*$ adapted by (3.2), (3.3), (3.4), (3.5), (3.14), (3.15), (3.16) and (3.17) respectively, is totally equivalent to the 2nn-2-1 model given by (3.10) with the synaptic weights (X^*, Z_h^*, Z_a^*, Z_b^*, $V_{b_i}^*$, $V_{a_i}^*$, $W_{b_i}^*$ and $W_{a_i}^*$) adapted by (3.2), (3.3), (3.4), (3.5), (3.6), (3.7), (3.8) and (3.9), respectively. The computational cost to adapt the synaptic weights X^*, Z_h^*, Z_a^*, Z_b^* is $O(X) = N$, $O(Z_h) = N$, $O(Z_a) = N$ and $O(Z_b) = N$ for both architectures. The only difference is that, to adapt the synaptic weights V_{b_1}, V_{a_1}, W_{b_1} and W_{a_1} we only require $O(W_{b_1}) = N \times n_b$, $O(W_{a_1}) = N \times n_a$, $O(V_{b_1}) = N$, $O(V_{a_1}) = N$ computations instead of the $O(W_{b_i}) = N \times (nn \times n_b)$, $O(W_{a_i}) = N \times (nn \times n_a)$, $O(V_{b_i}) = N \times nn$, $O(V_{a_i}) = N \times nn$ computations required to adapt the synaptic weights V_{b_i}, V_{a_i}, W_{b_i} and W_{a_i}.

Then, the computational cost required to train the 2nn-2-1 model ($O(2nn-2-1) = (((nn \times n_b) + (nn \times n_a)) + (2 \times nn) + (4)) \times N$) is reduced to ($O(2-2-1) = (n_b + n_a + 4) \times N$). This completes the proof. ■

The reader shall notice that $O(2nn-2-1)$ and $O(2-2-1)$ are a measured of *the number of parameters* times *the number data required for its estimation* ($n_p \times N$).

In application, a comparison in terms of the time required to compute all the parameters of a model estimated by both, a classical way and the proposed approach (Theorem 2) is done in Chapter 5 (see Section 5.1.2, Table 5.1), where the classical training takes $12.78min$ and Theorem 2 leads $2.85min$, it shows, the interest of the proposed approach.

To conclude, following the proof of the theorems, we have a method to generate mathematic models with the simplicity of the 2-1 architecture presented in Fig. 2.2, the accuracy of the 2nn-2-1 neural network depicted in Fig. 2.1 and the computational complexity of the 2-2-1 neural network shown in Fig. 2.3, this methodology in presented in the following chapter.

Chapter 4

A new efficient system identification methodology

In this chapter Theorem 1 and Theorem 2 previously established are used to propose an efficient system identification method. Since the proposed procedure follows the same steps as the proof of the theorems, this approach yields to accurate mathematical models with a reduced number of parameters and a reduced computational cost: the so-called quality/cost balance. In order to provide to the reader different examples, this original procedure is applied to some of the proposed neural network architectures in Chapter 2. Moreover, we shall take the opportunity to apply this new methodology to a neural network corresponding to the sigmoid estimator of the toolbox for nonlinear system identification in Matlab.

4.1 The proposed system identification procedure

Let us first introduce the proposed methodology which naturally resume the fundamental steps classically defined by Fig. 4.1, where we partially focus on the points referred to as "Choose model set" and "Model computation".

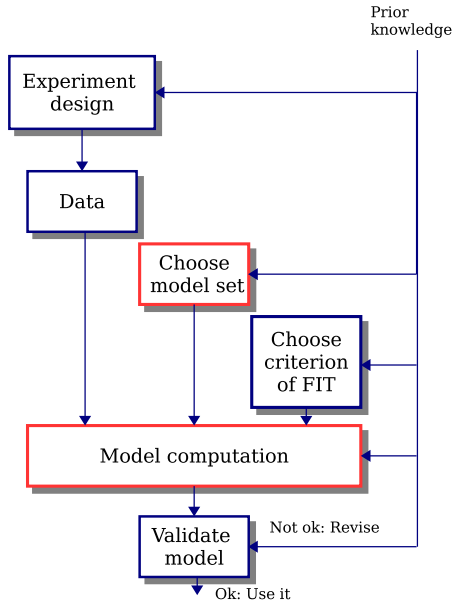


Figure 4.1: System identification procedure.

- **Choice model set:** As mentioned in the general introduction, a set of candidate models is defined by specifying within which collection of models we are going to look for a suitable candidate. In order to achieve accurate models, with a reduced number of parameters and at a low computational cost (the so-called quality/cost balance), the neural network architecture (structure of the model set) should be chosen according to Assumption 1, i.e., to choose at least all the activation functions of one layer as linear, that is, $\varphi_1(T) = T$ or $\varphi_2(T) = T$ or $\varphi_3(T) = T$ in the neural network structure presented in Chapter 2.

- **Improved model computation:** As already presented in the general introduction, once the set of candidate models has been established, the search for the best suitable model within the set becomes a problem of determining, or estimating the model parameters (synaptic weights) according to a given criterion.

The reader shall notice that, we are proposing an improved procedure based on the demonstration of the theorems presented in Chapter 3:

Step 1.- Neural network training under particular assumptions

Once the neural network structure is established according to Assumption 1, the designer should satisfy Assumption 2, that is, to select the initial conditions of the synaptic weights equals group by group, i. e., $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{b_1}(0) = W_{b_j}(0)$ and $W_{a_1}(0) = W_{a_j}(0)$ with $j = 2, 3, \cdots, nn$.

Then, the neural network must be trained following the proof of Theorem 2, that is, to apply the proposed "computational cost reduction approach" presented in Section 3.3.

Step 2.- Model transformation

Once the 2nn-2-1 neural network is trained under the specified assumptions, its representation is transformed into the 2-1 architecture as presented in the proof of Theorem 1, that is, to apply the proposed "model complexity reduction approach" presented in Section 3.2.

Since the model transformation is done by algebraic operations (change in notation) and after the training phase (see Section 1.3.2), no computational cost and no loss of accuracy are carried out during Step 2.

It is interesting to notice that, by following these improved system identification procedure, i.e. choice of model structure according to Assumption 1 and model estimation by following the step 1 and the step 2 presented above, we derive accurate models with a reduced number of parameters and at a low computational cost. In fact, we are improving the classical model computation by applying the theorems proposed in Chapter 3.

Once the two main steps of the proposed system identification procedure are established, let us now present three examples of the application of the proposed method to different neural network architectures. These architectures will be used for the identification of different systems in Chapter 5. Let us now see that concretely it works.

4.2 Example 1: Recurrent 2nn-2-1 without thresholds

Let us take as a first example the recurrent 2nn-2-1 neural network of Fig. 4.2, the reader shall notice that this architecture has the same structure of the neural network shown in Fig. 2.1, but this architecture has not thresholds in any of its layers.

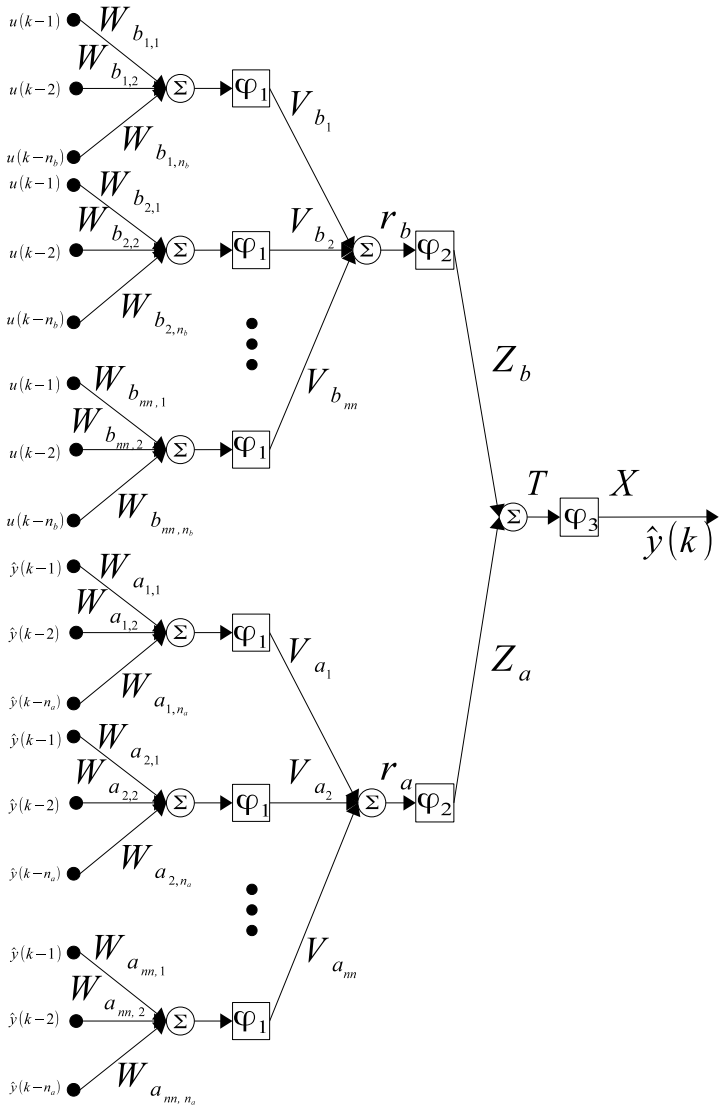


Figure 4.2: Recurrent 2nn-2-1 neural network.

The mathematical representation of the such 2nn-2-1 neural network is given by (4.1).

$$\begin{aligned} \hat{y}(k) &= X\varphi_3(T) \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) \\ r_b &= \sum_{i=1}^{nn} V_{b_i}\varphi_1(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}\varphi_1(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{4.1}$$

where:

$J_u = [u(k-1) \quad u(k-2) \quad ... \quad u(k-n_b)] \quad \epsilon \quad R^{1\times n_b}$
$J_{\hat{y}} = [\hat{y}(k-1) \quad \hat{y}(k-2) \quad ... \quad \hat{y}(k-n_a)] \quad \epsilon \quad R^{1\times n_a}$
$W_{b_i} = [W_{b_{i,1}} \quad W_{b_{i,2}} \quad ... \quad W_{b_{i,n_b}}]^\top \quad \epsilon \quad R^{n_b\times 1}$
$W_{a_i} = [W_{a_{i,1}} \quad W_{a_{i,2}} \quad ... \quad W_{a_{i,n_a}}]^\top \quad \epsilon \quad R^{n_a\times 1}$
$X,\ Z_b,\ Z_a,\ V_{b_i}, V_{a_i} \epsilon R^1$
$X,\ Z_b,\ Z_a,\ V_{b_i},\ V_{a_i},\ W_{b_i}$ and W_{a_i} are the synaptic weights.
with $i = 1, 2, \cdots, nn$ and nn is the number of neurons.

Let us follow the proposed system identification procedure:

- **Choice of model structure:** The first step in the proposed system identification method is to choose the activation functions according to Assumption 1. Different models can be derived in order to satisfy this assumption. Let us name the following model families.

  Model FF1: By selecting $\varphi_3(z) = \varphi_2(z) = z$ and $\varphi_1(z) = nonlinear$ in (4.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b r_b + Z_a r_a \\ r_b &= \sum_{i=1}^{nn} V_{b_i}\varphi_1(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}\varphi_1(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{4.2}$$

  Model FF2: By selecting $\varphi_3(z) = \varphi_1(z) = z$ and $\varphi_2(z) = nonlinear$ in (4.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) \\ r_b &= \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{4.3}$$

Model ARX: By selecting $\varphi_3(z) = \varphi_2(z) = \varphi_1(z) = z$ in (4.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b^*(r_b) + Z_a(r_a) \\ r_b &= \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{4.4}$$

Model NLARX: By selecting $\varphi_2(z) = \varphi_1(z) = z$ and $\varphi_3(z) = nonlinear$ in (4.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= X\varphi_3(T) \\ T &= Z_b^*(r_b) + Z_a(r_a) \\ r_b &= \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{4.5}$$

In the sequel, without loss of generality we choose to develop the computation step only for the model FF1 given by (4.2). The application of Theorem 1 and Theorem 2 to the rest of the models is presented in Appendix G.

- **Model computation:** Let us follow the improved procedure:

Step 1.- Neural network training under particular assumptions

According to Assumption 2, the initial conditions of the synaptic weights are chosen equals group by group as follows $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{b_1}(0) = W_{b_j}(0)$ and $W_{a_1}(0) = W_{a_j}(0)$ with $j = 2, 3, \cdots, nn$.

Then, the proposed neuro-model is trained following the computational cost reduction approach presented in the proof of Theorem 2, that is:

To train the equivalent model given by (4.6):

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b r_b + Z_a r_a \\ r_b &= nn \times V_{b_1}\varphi_1(J_u W_{b_1}) \\ r_a &= nn \times V_{a_1}\varphi_1(J_{\hat{y}} W_{a_1}) \end{aligned} \tag{4.6}$$

with the synaptic weights computed as follows:

$$
\begin{aligned}
&X(k+1) = X(k) + \eta e(k) T \\
&Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \\
&Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \\
&V_{b_1}(k+1) = V_{b_1}(k) + \eta e(k) X Z_b \tanh(J_u W_{b_1}) \\
&V_{a_1}(k+1) = V_{a_1}(k) + \eta e(k) X Z_a \tanh(J_{\hat{y}} W_{a_1}) \\
&W_{b_1}(k+1) = W_{b_1}(k) + \eta e(k) X Z_b V_{b_1} sech^2(J_u W_{b_1}) J_u \\
&W_{a_1}(k+1) = W_{a_1}(k) + \eta e(k) X Z_a V_{a_1} sech^2(J_{\hat{y}} W_{a_1}) J_{\hat{y}}
\end{aligned}
$$

Once the neural network model given by (4.2) is trained under these two assumptions, we obtain:

$$
\begin{aligned}
&\hat{y}(k) = X^* T \\
&T = Z_b^* r_b + Z_a^* r_a \\
&r_b = \sum_{i=1}^{nn} V_{b_i}^* \varphi_1(J_u W_{b_i}^*) \\
&r_a = \sum_{i=1}^{nn} V_{a_i}^* \varphi_1(J_{\hat{y}} W_{a_i}^*)
\end{aligned}
\tag{4.7}
$$

Since the initial conditions of the synaptic weights are chosen equals group by group (see Assumption 2), and each group is trained by the same adaptation rule, the final values of the synaptic weights are: $V_{b_1}^* = V_{b_j}^*$, $V_{a_1}^* = V_{a_j}^*$, $W_{b_1}^* = W_{b_j}^*$ and $W_{a_1}^* = W_{a_j}^*$ with $j = 2, 3, \cdots, nn$.

Step 2.- Model transformation

According to Theorem 1, the final values of the synaptic weights are: $W_{b_1}^* = W_{b_j}^*$, $V_{b_1}^* = V_{b_j}^*$, $W_{bh_1}^* = W_{bh_j}^*$, $W_{a_1}^* = W_{a_j}^*$, $V_{a_1}^* = V_{a_j}^*$ and $W_{ah_1}^* = W_{ah_j}^*$ with $j = 2, 3, \cdots, nn$. Then, in (4.7) it is indeed possible to make the following algebraic operations:

$$
\begin{aligned}
V_{B_i}^* &= X^* \times Z_b^* \times V_{b_i}^* \\
V_{A_i}^* &= X^* \times Z_a^* \times V_{a_i}^*
\end{aligned}
$$

where $V_{B_1}^* = V_{B_j}^*$, $V_{A_1}^* = V_{A_j}^*$ (with $j = 2, 3, \cdots, nn$) due to Assumption 2 (Step 1).

Then, the 2nn-2-1 neuro-model given by (4.7) becomes:

$$
\begin{aligned}
&\hat{y}(k) = T \\
&T = r_b + r_a \\
&r_b = \sum_{i=1}^{nn} V_{B_i}^* \varphi_1(J_u W_{b_i}^*) \\
&r_a = \sum_{i=1}^{nn} V_{A_i}^* \varphi_1(J_{\hat{y}} W_{a_i}^*)
\end{aligned}
\tag{4.8}
$$

A supplementary transformation is achieved in order to change the redefined 2nn-1 model (4.8) into a 2-1 representation by the following algebraic operations:

$$\sum_{i=1}^{nn} V_{B_i}^* \varphi_1(J_u W_{b_i}^*) = V_B^* \varphi_1(J_u W_B^*)$$
$$\sum_{i=1}^{nn} V_{A_i}^* \varphi_1(J_{\hat{y}} W_{a_i}^* = V_A^* \varphi_1(J_{y_n} W_A^*)$$

where:

$$V_B^* = nn \times V_{B_i}^*$$
$$V_A^* = nn \times V_{A_i}^*$$
$$W_B^* = W_{b_1}^*$$
$$W_A^* = W_{a_1}^*$$

The resulting model after the model transformation has the following mathematical form:

$$\hat{y}(k) = V_B^* \varphi_1(J_u W_B^*) + V_A^* \varphi_1(J_{\hat{y}} W_A^*) \tag{4.9}$$

In this example, both the model complexity reduction approach and the computational cost reduction approach were applied to one of the four possible models, although, since the four models FF1, FF2, ARX and NARX (see (4.2), (4.3), (4.4) and (4.5) respectively) satisfy Assumption 1 and can be trained as proposed in Theorem 2 under Assumption 2, we can apply the proposed model reduction approach (see Appendix G), in order to generate the following ready-to-use models:

FF1 after model reduction:

$$\hat{y}(k) = V_B^* \varphi_1(J_u W_B^*) + V_A^* \varphi_1(J_{\hat{y}} W_A^*) \tag{4.10}$$

FF2 after model reduction:

$$\hat{y}(k) = V_B^* \varphi_2(J_u W_B^*) + V_A^* \varphi_2(J_{\hat{y}} W_A^*) \tag{4.11}$$

ARX after model reduction:

$$\hat{y}(k) = J_u W_B^* + J_{\hat{y}} W_A^* \tag{4.12}$$

NLARX after model reduction:

$$\hat{y}(k) = X^* \varphi_3(J_u W_B^* + J_{\hat{y}} W_A^*) \tag{4.13}$$

It is interesting to notice that, the reduced models keep the same accuracy as their original non reduced models, by this way, we yield balanced accuracy/complexity/cost models.

4.3 Example 2: Recurrent 2nn-2-1 with thresholds in all the layers

It seem necessary to proof that our approach also work when using model structures with thresholds. Therefore, let us take as an example the 2nn-2-1 recurrent neural network of Fig. 4.3, notice that this architecture has the same structure of the neural networks shown in Fig. 2.1 and Fig. 4.2, but this architecture have thresholds in all its layers.

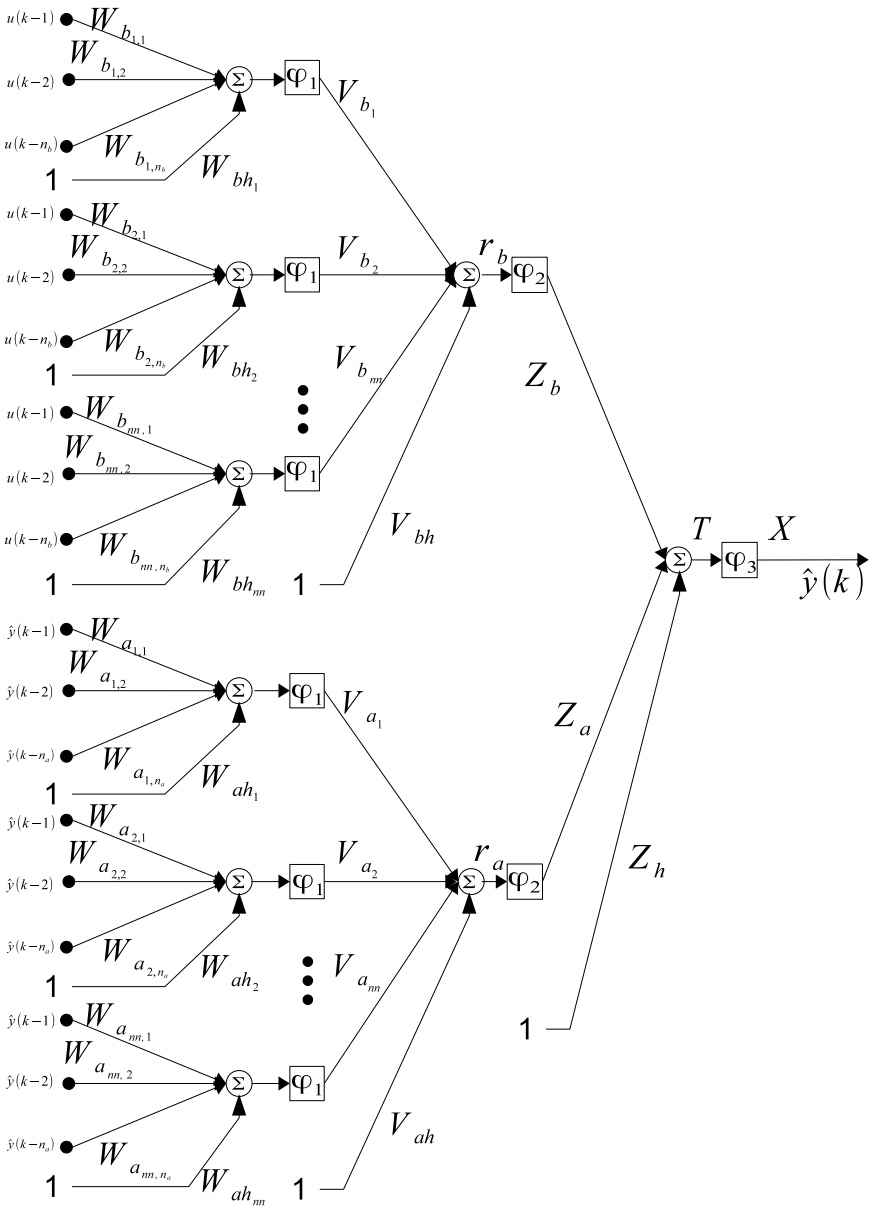


Figure 4.3: Recurrent 2nn-2-1 neural network.

The mathematical representation of the proposed recurrent architecture is given by (4.14).

$$\begin{aligned} \hat{y}(k) &= X\varphi_3(T) \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) + Z_h \\ r_b &= V_{bh} + \sum_{i=1}^{nn} V_{b_i}\varphi_1(J_u W_{b_i} + W_{bh_i}) \\ r_a &= V_{ah} + \sum_{i=1}^{nn} V_{a_i}\varphi_1(J_{\hat{y}} W_{a_i} + W_{ah_i}) \end{aligned} \tag{4.14}$$

where:

$J_u = [u(k-1) \quad u(k-2) \quad ... \quad u(k-n_b)] \quad \epsilon \quad R^{1\times n_b}$
$J_{\hat{y}} = [\hat{y}(k-1) \quad \hat{y}(k-2) \quad ... \quad \hat{y}(k-n_a)] \quad \epsilon \quad R^{1\times n_a}$
$W_{b_i} = [W_{b_{i,1}} \quad W_{b_{i,2}} \quad ... \quad W_{b_{i,n_b}}]^\top \quad \epsilon \quad R^{n_b\times 1}$
$W_{a_i} = [W_{a_{i,1}} \quad W_{a_{i,2}} \quad ... \quad W_{a_{i,n_a}}]^\top \quad \epsilon \quad R^{n_a\times 1}$
$X,\ Z_b,\ Z_a,\ V_{b_i},\ V_{a_i},\ W_{bh_i},\ W_{ah_i},\ V_{bh},\ V_{ah},\ Z_h\ \epsilon R^1$
$X,\ Z_b,\ Z_a,\ V_{b_i},\ V_{a_i},\ W_{b_i},\ W_{a_i},\ W_{bh_i},\ W_{ah_i},\ V_{bh},\ V_{ah}$ and Z_h are synaptic weights.

with $i = 1, 2, \cdots, nn$ and nn is the number of neurons.

Let us follow the proposed system identification procedure:

- **Choice of model structure:** The first step in the proposed system identification method is to choose the activation functions according to Assumption 1. Different models can be derived in order to satisfy this assumption. Let us name the following model families.

  FF1HH model: By selecting $\varphi_3(z) = \varphi_2(z) = z$ in (4.14) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b r_b + Z_a r_a + Z_h \\ r_b &= V_{bh} + \sum_{i=1}^{nn} V_{b_i}\varphi_1(J_u W_{b_i} + W_{bh_i}) \\ r_a &= V_{ah} + \sum_{i=1}^{nn} V_{a_i}\varphi_1(J_{\hat{y}} W_{a_i} + W_{ah_i}) \end{aligned} \tag{4.15}$$

  FF2HH model: By selecting $\varphi_3(z) = \varphi_1(z) = z$ in (4.14) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) + Z_h \\ r_b &= V_{bh} + \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i} + W_{bh_i}) \\ r_a &= V_{ah} + \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i} + W_{ah_i}) \end{aligned} \tag{4.16}$$

ARXHH model: By selecting $\varphi_1(z) = \varphi_2(z) = \varphi_3(z) = z$ in (4.14) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b r_b + Z_a r_a + Z_h \\ r_b &= V_{bh} + \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i} + W_{bh_i}) \\ r_a &= V_{ah} + \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i} + W_{ah_i}) \end{aligned} \tag{4.17}$$

NLARXHH model: By selecting $\varphi_1(z) = \varphi_2(z) = z$ in (4.14) we obtain:

$$\begin{aligned} \hat{y}(k) &= X\varphi_3(T) \\ T &= Z_b r_b + Z_a r_a + Z_h \\ r_b &= V_{bh} + \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i} + W_{bh_i}) \\ r_a &= V_{ah} + \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i} + W_{ah_i}) \end{aligned} \tag{4.18}$$

In the sequel, without loss of generality we choose to develop the computation step only for the model FFHH1 given by (4.15). The application of Theorem 1 and Theorem 2 to the rest of the models is presented in Appendix G.

- **Model computation:** Let us follow the improved procedure:

Step 1.- Neural network training under particular assumptions

According to Assumption 2, the initial conditions of the synaptic weights are chosen equals group by group, for this architecture that is: $W_{b_1}(0) = W_{b_j}(0)$, $W_{a_1}(0) = W_{a_j}(0)$, $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{bh_1}(0) = W_{bh_j}(0)$ and $W_{ah_1}(0) = W_{ah_j}(0)$ with $j = 2, 3, \cdots, nn$.

Then, we have to train the proposed neuro-model by the computational cost reduction approach as presented in Theorem 2, that is, to train the equivalent model given by (4.19).

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b r_b + Z_a r_a + Z_h \\ r_b &= V_{bh} + nn \times V_{b_1}\varphi_1(J_u W_{b_1} + W_{bh_1}) \\ r_a &= V_{ah} + nn \times V_{a_1}\varphi_1(J_{\hat{y}} W_{a_1} + W_{ah_1}) \end{aligned} \tag{4.19}$$

with the synaptic weights adapted as follows:

$$
\begin{aligned}
&X(k+1) = X(k) + \eta e(k) T \\
&Z_h(k+1) = Z_h(k) + \eta e(k) X \\
&Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \\
&Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \\
&V_{bh}(k+1) = V_{bh}(k) + \eta e(k) X Z_b \\
&V_{ah}(k+1) = V_{ah}(k) + \eta e(k) X Z_a \\
&V_{b_1}(k+1) = V_{b_1}(k) + \eta e(k) X Z_b \tanh(J_u W_{b_1} + W_{bh_1}) \\
&V_{a_1}(k+1) = V_{a_1}(k) + \eta e(k) X Z_a \tanh(J_{\hat{y}} W_{a_1} + W_{ah_1}) \\
&W_{bh_1}(k+1) = W_{bh_1}(k) + \eta e(k) X Z_b V_{b_1} sech^2(J_u W_{b_1} + W_{bh_1}) \\
&W_{ah_1}(k+1) = W_{ah_1}(k) + \eta e(k) X Z_a V_{a_1} sech^2(J_{\hat{y}} W_{a_1} + W_{ah_1}) \\
&W_{b_1}(k+1) = W_{b_1}(k) + \eta e(k) X Z_b V_{b_1} sech^2(J_u W_{b_1} + W_{bh_1}) J_u \\
&W_{a_1}(k+1) = W_{a_1}(k) + \eta e(k) X Z_a V_{a_1} sech^2(J_{\hat{y}} W_{a_1} + W_{ah_1}) J_{\hat{y}}
\end{aligned}
$$

Once the training process is completed, we have the trained model given by:

$$
\begin{aligned}
&\hat{y}(k) = X^* T \\
&T = Z_b^* r_b + Z_a^* r_a + Z_h^* \\
&r_b = V_{bh}^* + \sum_{i=1}^{nn} V_{b_i}^* \varphi_1(J_u W_{b_i}^* + W_{bh_i}^*) \\
&r_a = V_{ah}^* + \sum_{i=1}^{nn} V_{a_i}^* \varphi_1(J_{\hat{y}} W_{a_i}^* + W_{ah_i}^*)
\end{aligned}
\tag{4.20}
$$

Step 2.- Model transformation

According to Theorem 1, the final values of the synaptic weights are: $W_{b_1}^* = W_{b_j}^*$, $V_{b_1}^* = V_{b_j}^*$, $W_{bh_1}^* = W_{bh_j}^*$, $W_{a_1}^* = W_{a_j}^*$, $V_{a_1}^* = V_{a_j}^*$ and $W_{ah_1}^* = W_{ah_j}^*$ with $j = 2, 3, \cdots, nn$. Then, in (4.20) it is indeed possible to make the following algebraic operations:

$$
\begin{aligned}
&Z_H^* = X^* \times Z_h^* \\
&V_{BH}^* = X^* \times Z_b^* \times V_{bh}^* \\
&V_{AH}^* = X^* \times Z_a^* \times V_{ah}^* \\
&V_{B_i}^* = X^* \times Z_b^* \times V_{b_i}^* \\
&V_{A_i}^* = X^* \times Z_a^* \times V_{a_i}^*
\end{aligned}
$$

where $V_{B_1}^* = V_{B_j}^*$, $V_{A_1}^* = V_{A_j}^*$ (with $j = 2, 3, \cdots, nn$) due to Assumption 2 (see Step 1). Then, the 2nn-2-1 neuro-model given by (4.20) becomes:

$$
\begin{aligned}
&\hat{y}(k) = T \\
&T = r_b + r_a + Z_H^* \\
&r_b = V_{BH}^* + \sum_{i=1}^{nn} V_{B_i}^* \varphi_1(J_u W_{b_i}^* + W_{bh_i}^*) \\
&r_a = V_{AH}^* + \sum_{i=1}^{nn} V_{A_i}^* \varphi_1(J_{\hat{y}} W_{a_i}^* + W_{ah_i}^*)
\end{aligned}
\tag{4.21}
$$

A supplementary transformation is achieved in order to change the redefined 2nn-1 model (4.21) by the following algebraic operations:

$$\sum_{i=1}^{nn} V_{B_i}^* \varphi_1(J_u W_{b_i}^* + W_{b_h}^*) = V_B^* \varphi_1(J_u W_B^* + W_{BH}^*)$$
$$\sum_{i=1}^{nn} V_{A_i}^* \varphi_1(J_{\hat{y}} W_{a_i}^* + W_{a_h}^*) = V_A^* \varphi_1(J_{y_n} W_A^* + W_{AH}^*)$$

where:

$$\begin{aligned} V_B^* &= nn \times V_{B_i}^* \\ V_A^* &= nn \times V_{A_i}^* \\ W_B^* &= W_{b_1}^* \\ W_A^* &= W_{a_1}^* \\ W_{BH}^* &= W_{bh_1}^* \\ W_{AH}^* &= W_{ah_1}^* \\ H^* &= Z_H^* + V_{BH}^* + V_{AH}^* \end{aligned}$$

The resulting model after the "neurons reduction" has the following mathematical form:

$$\hat{y}(k) = V_B^* \varphi_1(J_u W_B^* + W_{BH}^*) + V_A^* \varphi_1(J_{\hat{y}} W_A^* + W_{AH}^*) + H^* \tag{4.22}$$

Since the four models FFHH1, FFHH2, ARXHH and NARXHH (see (4.15), (4.16), (4.17) and (4.18) respectively) satisfy Assumption 1 and can be trained as proposed in Theorem 2 under Assumption 2, we can apply the proposed model reduction approaches (see Appendix G), in order to generate the following ready-to-use models:

FFHH1 after model reduction:

$$\hat{y}(k) = V_B^* \varphi_1(J_u W_B^* + W_{BH}^*) + V_A^* \varphi_1(J_{\hat{y}} W_A^* + W_{AH}^*) + H^* \tag{4.23}$$

FFHH2 after model reduction:

$$\hat{y}(k) = Z_B^* \varphi_2(J_u W_B^* + W_{BH}^*) + Z_A^* \varphi_2(J_{\hat{y}} W_A^* + W_{AH}^*) + H^* \tag{4.24}$$

ARXHH after model reduction:

$$\hat{y}(k) = (J_u W_B^*) + (J_{\hat{y}} W_A^*) + H^* \tag{4.25}$$

NLARXHH after model reduction:

$$\hat{y}(k) = X^* \varphi_3((J_u W_B^*) + (J_{\hat{y}} W_A^*) + H^*) \tag{4.26}$$

We shall conclude that our approach treats with the same equivalent manner neural network structures with or without thresholds.

Two examples of the application of the proposed reduction procedures have been presented, in both cases the proposed reduction approaches are applied to the particular designs developed in this thesis (one architecture without thresholds and one architecture with thresholds in all the neurons of the neural network). In a third example, the proposed model reduction approach is applied to a classical architecture, in order to show that the proposed model reduction approach is relevant to all the neural networks which accomplish the two assumptions.

4.4 Example 3: Sigmoid network of the toolbox in Matlab

In this example, the proposed model reduction approach is applied to other neural network architecture. Here, we will answer to the following question: For any reason, I choose a different type of neuro-model set. Can I nevertheless use the model reduction approach?.

Let us take the sigmoid neural network used for the estimation of the Nonlinear ARX model shown in Fig. 4.4. This architecture, was reproduced in the thesis ([86]) and correspond to the sigmoid estimator of the non linear system identification toolbox in Matlab.

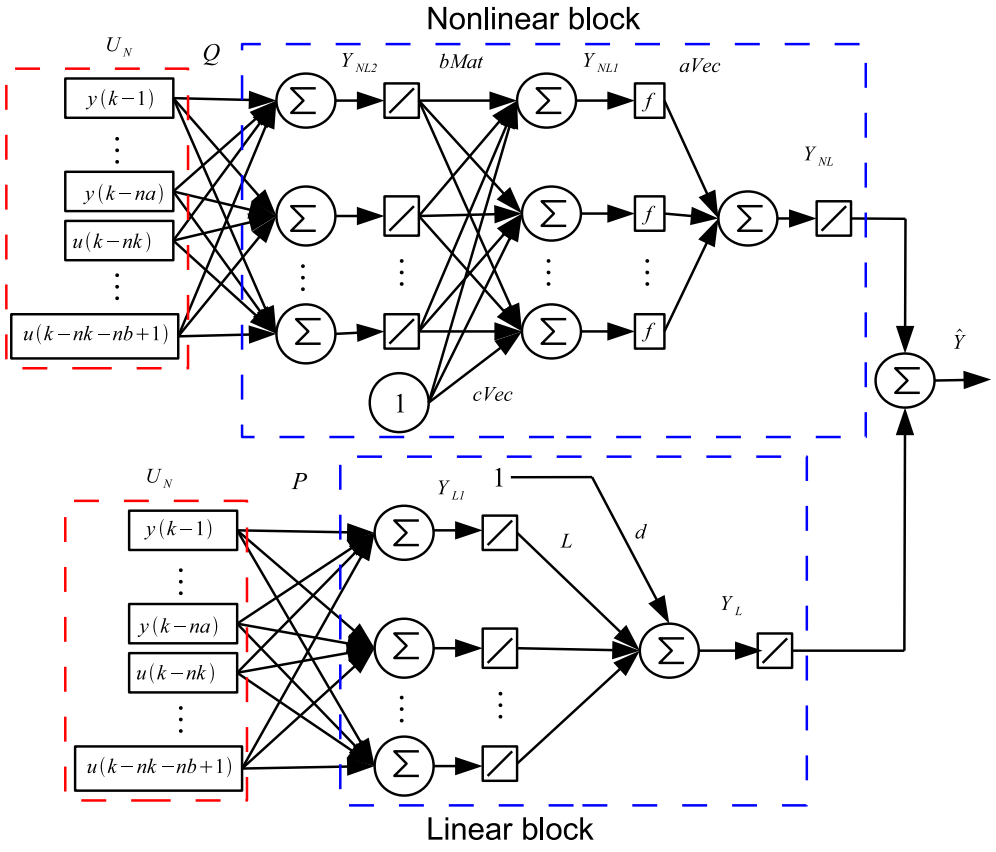


Figure 4.4: Non linear ARX.

Let us follow the proposed system identification procedure:

- **Choice of model structure:** The first step in the proposed system identification method is to choose the activation functions according to Assumption 1.

  The mathematical representation of the sigmoid network used in this example (Fig. 4.4) is given by (4.27)).

$$\begin{aligned} \hat{y}(k) &= y_{NL} + y_L \\ y_{NL} &= tanh(y_{NL1})aVec \\ y_{NL1} &= y_{NL2}bMat + cVec \\ y_{NL2} &= U_N Q \\ y_L &= y_{L1}L + d \\ y_{L1} &= U_N P \end{aligned} \tag{4.27}$$

  where:

$$U_N \in \mathbb{R}^{1\times(na+nb)} \text{ is the regression vector.}$$

  and the parameters of the model (synaptic weights) are:

$$\begin{aligned} &Q \in \mathbb{R}^{(na+nb)\times(na+nb)} \\ &bMat \in \mathbb{R}^{(na+nb)\times nn} \\ &aVec \in \mathbb{R}^{nn\times 1} \\ &cVec \in \mathbb{R}^{1\times nn} \\ &P \in \mathbb{R}^{(na+nb)\times(na+nb)} \\ &L \in \mathbb{R}^{(na+nb)\times 1} \\ &d \in \mathbb{R} \end{aligned}$$

  Notice that, this model (4.27) consists on two neural networks (one linear and the other one nonlinear) and each architecture directly satisfies Assumption 1. Therefore, it is not necessary to change the activation functions of this model. We can apply the system identification procedure proposed in this thesis, since we consider the complete architecture satisfies Assumption 1.

- **Model computation:** Let us follow the improved procedure:

  Step 1.- Neural network training under particular assumptions

  According to Assumption 2, the initial condition of the synaptic weights are chosen equals group by group, i.e. $Q_{1,:} = Q_{(j:,1)}$, $P_{1,:} = P_{(j:,1)}$, with $j = 2, \cdots, n_a + n_b$ and $bMat_{1,:} = bMat_{i,:}$, $L_{1,:} = L_{i,:}$ with $i = 2, \cdots, nn$.

  Once Assumption 1 and Assumption 2 are fulfilled, the neural is trained. For this architecture, the adaptation of the synaptic weights is achieved according to the

original algorithms presented in the thesis ([86]), therefore the computational cost reduction approach is not applied for this example.

$$\begin{aligned}
&\hat{y}(k) = y_{NL} + y_L \\
&y_{NL} = tanh(y_{NL1})aVec^* \\
&y_{NL1} = y_{NL2}bMat^* + cVec^* \\
&y_{NL2} = U_N Q^* \\
&y_L = y_{L1}L^* + d^* \\
&y_{L1} = U_N P^*
\end{aligned} \tag{4.28}$$

It is interesting to remark that, even if the adaptation of the synaptic weights is achieved by the original algorithms used in the thesis ([86]), the final values of the synaptic weights of the trained model given by (4.28) are equals group by group: $Q^*_{1,:} = Q^*_{(j:,1)}$, $P^*_{1,:} = P^*_{(j:,1)}$, with $j = 2, \cdots, n_a + n_b$ and $bMat^*_{1,:} = bMat^*_{i,:}$, $L^*_{1,:} = L^*_{i,:}$ with $i = 2, \cdots, nn$.

Step 2.- Model transformation

Once the neural network is trained under Assumption 1 and Assumption 2, in (4.28) it is indeed possible to make the following algebraic operations.

$$\begin{aligned}
&Q_1 = Q * bMat \\
&P_1 = P * L
\end{aligned}$$

Then, the model is redefined as:

$$\begin{aligned}
&\hat{y}(k) = y_{NL} + y_L \\
&y_{NL} = tanh(y_{NL1})aVec \\
&y_{NL1} = y_{NL2} + cVec \\
&y_{NL2} = U_N Q_1 \\
&y_L = y_{L1} + d \\
&y_{L1} = U_N P_1
\end{aligned} \tag{4.29}$$

where:

$$\begin{aligned}
&P_1 \in \mathbb{R}^{(n_a+n_b)\times 1} \\
&Q_1 \in \mathbb{R}^{(n_a+n_b)\times nn} \\
&\text{with } Q1^*_{1,:} = Q1^*_{(j:,1)} \text{ with } j = 2, \cdots, nn.
\end{aligned}$$

Finally, in (4.29) it is indeed possible to make the following operations:

$$\begin{aligned}
&Q_2 = Q_1(:, 1) \\
&cVec_1 = cVec(1) \\
&aVec_1 = nn * aVec(1)
\end{aligned}$$

Then, the model given by (4.29) becomes (4.30):

$$\hat{y}(k) = y_{NL} + y_L \tag{4.30}$$
$$y_{NL} = tanh(JQ_2 + cVec_1)aVec_1$$
$$y_L = J * P_1 + d$$

(4.31)

where:

$$Q_2 \in \mathbb{R}^{(n_a+n_b)\times 1}$$
$$cVec_1 \in \mathbb{R}^1$$
$$aVec_1 \in \mathbb{R}^1$$

Then, by using the proposed model reduction approach, a model with $((na + nb) \times (na+nb))+((na+nb)\times nn)+(nn\times 1)+(1\times nn)+((na+nb)\times(na+nb))+((na+nb)\times 1)+(1)$ parameters is reduced to a model with $((n_a+n_b)\times 1)+(1)+(1)+((n_a+n_b)\times 1)+(1)$ parameters.

Interesting results have been obtained by the application of the reduction procedures to the different neural network presented in the preceding examples. Our goal clearly was to show that our model reduction procedures preserving the estimation accuracy was applicable to the widest model rage possible.

In the following chapter, these architectures are applied to the identification of different systems.

Chapter 5

Application to complex system identification

In the sequel, we applied the previous methodology to both simulation examples and real experiment setups, in order to validate the preceding results. Simulation is run on the unavoidable Wiener-Hammerstein benchmark. On the other hand, our methodology is also applied for data coming from real experimental set ups like flexible robot arm and an acoustic duct.

5.1 Wiener-Hammerstein benchmark

Let us start with the Wiener-Hammerstein benchmark proposed in [87]. The DUT is an electronic nonlinear circuit with a Wiener-Hammerstein structure (see Fig. 5.1).

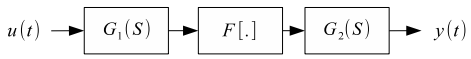


Figure 5.1: Wiener-Hammerstein system.

The first filter $G_1(s)$ is designed as a third order Chebyshev filter (pass-band ripple of $0.5dB$ and cut off frequency of $4.4kHz$). The second filter $G_2(s)$ is designed as a third order inverse Chebyshev filter (stop-band attenuation of $40dB$ starting at $5kHz$). The static nonlinearity $F[.]$ is built using a diode circuit (Fig. 5.2).

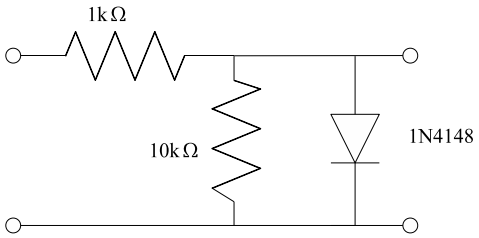


Figure 5.2: Circuit used to built the static nonlinear system.

The system is excited with a band-limited filtered Gaussian excitation signal and a dataset of 188000 data is generated, corresponding to $3.6719s$ at the sampling frequency of $51200Hz$. Further details are available in ([87]).

5.1.1 Validation test

The data set available for the benchmark is gathered in two records: the system is identified using the "estimation data" $(u(t), y(t)$ for $t = 1, 2, \cdots, 100000)$. Then, the model is simulated to estimate the output $\hat{y}(t)$ of the system, using no more than the input data at the span of time of the "test data" set $(t = 100001, \cdots 188000)$.

The benchmark established the basis to enable the comparison of the models quality. The quality assessment comprises four performance statistical indicators:

- The mean value of the simulation error:

$$\mu_t = \frac{1}{87000} \sum_{t=101001}^{188000} e_{sim}(t). \tag{5.1}$$

- The standard deviation of the error:

$$s_t^2 = \frac{1}{87000} \sum_{t=101001}^{188000} (e_{sim}(t) - \mu_t)^2 \tag{5.2}$$

- The root mean square (RMS) of the error:

$$e_{RMSt}^2 = \frac{1}{87000} \sum_{t=101001}^{188000} (e_{sim}(t))^2 \tag{5.3}$$

  In Eqs. 5.1-5.3, the sum is started at $t = 101001$ instead of $t = 188000$ to eliminate the influence of transient errors at the beginning of the simulation.

- The root mean square (RMS) value of the error for the estimation data ($t \in [1001, 100000]$):

$$e_{RMSe}^2 = \frac{1}{99000} \sum_{t=1001}^{100000} (e_{sim}(t))^2 \tag{5.4}$$

5.1.2 Experiments

In order to validate the proposed system identification approach, the following experiments are conducted:

Experiment 1.-The system is identified with the complex 2nn-2-1 model given by (3.1) with the nonlinear function classically chosen as $\varphi_1(z) = \tanh(z)$ ([84]), $nn = 40$, $n_a = 4$, $n_b = 7$ and $n_k = 4$. One thousand training runs are conducted using the estimation data $(u(t), y(t)$ for $t = 1, 2, \cdots, 100000)$ each time with different initialization values, but satisfying Assumption 2 (Initial weights equals group by group). Once the 1000 models are obtained, we choose the best one according to the validation error μ_t (model FFH1) and the best one according to the validation error e_{RMSt} (model FFH2). For the model FFH1 the validation results are $\mu_t = 0.00471 \times 10^{-3}$, $s_t = 56.71 \times 10^{-3}$, $e_{RMSt} = 56.71 \times 10^{-3}$ and $e_{RMSe} = 56.03 \times 10^{-3}$ and for the model FFH2 the validation results are $\mu_t = -5.635 \times 10^{-3}$, $s_t = 50.20 \times 10^{-3}$, $e_{RMSt} = 50.51 \times 10^{-3}$ and $e_{RMSe} = 49.74 \times 10^{-3}$. Notice that these two non reduced models have

524 parameters and the computational complexity is $O(FFH1) = O(FFH2) = (((40\times7)+(40\times4))+(2\times40)+(4))\times100,000 = 52.4\times10^6$. In order to show the interest of our model complexity reduction approach, we apply this reduction approach (theorem 1) to both models FFH1 and FFH2, lets us name these models as "FFH1a" and "FFH2a" respectively. As demonstrated in the proof of the theorem 1, the complex models FFH1 and FFH2 are equivalent to the reduced models "FFH1a" and "FFH2a" respectively, and consequently the validation results are the same. The main advantage with this reduction approach is that, the reduced models have only 14 parameters instead of 524. But the computational complexity for the reduced models, is the same as for the complex models, that is $O(FFH1a) = O(FFH2a) = 52.4 \times 10^6$.

Experiment 2.-In order to validate the improvements on the computation time, the model given by (3.1) is trained with the same initial conditions which conduct to the models $FFH1a$ and $FFH2a$, this time the training is developed as presented in the proof of Theorem 2. Then, we apply Theorem 1 to each model in order to obtain two reduced models, let us name these two reduced models as $FFH1b$ and $FFH2b$ respectively. As demonstrated in the proof of Theorem 2, the final values of the parameters of the reduced models when we train the complex 2nn-2-1 architecture by the classical way ($FFH1a$ and $FFH2a$) and the final values of the parameters of the reduced models when we train the complex 2nn-2-1 architecture as presented in the proof of Theorem 2 ($FFH1b$ and $FFH2b$) are equals respectively. The advantage of the models $FFH1b$ and $FFH2b$ is that the computational complexity ($O(FFH1b) = O(FFH2b) = (((7)+(4))+(2)+(4))\times100,000 = 1.7\times10^6$), is lower ($1.7\times10^6$ vs. 52.4×10^6 operations).

After applying Theorem 1 and Theorem 2, we have two models of the form of (5.5).

$$\hat{y}(k) = V_B^* \tanh(J_u W_B^* + W_{BH}^*) + V_A^* \tanh(J_{\hat{y}} W_A^* + W_{AH}^*) + H^* \tag{5.5}$$

where:

$$J_u = [u(k-1) \quad u(k-2) \quad ... \quad u(k-7)]$$
$$J_{\hat{y}} = [\hat{y}(k-1) \quad \hat{y}(k-2) \quad ... \quad \hat{y}(k-4)]$$

and the 14 parameters characterizing the system are:

FFH1a=FFH1b:

$W_A^* = [0.10752 \quad 0.0548 \quad 0.0017 \quad -0.0495]$
$W_B^* = [0.0142 \quad -0.0035 \quad 0.0140 \quad 0.0245 \quad 0.0033 \quad -0.0152 \quad 0.0435]$
$V_A^* = 8.1996$, $V_B^* = 0.4842$
$H^* = -0.0022$

FFH2a=FFH2b:

$W_A^* = [0.164176 \quad 0.0932 \quad 0.00979 \quad -0.08208]$
$W_B^* = [0.01011 \quad -0.0054 \quad -0.0096 \quad 0.0176 \quad 0.0158 \quad -0.0326 \quad 0.02411]$
$V_A^* = 5.04290$, $V_B^* = 2.5643$
$H^* = -0.0020$

For comprehensive reasons, we can not present the 524 parameters of the non reduced models FFH1 and FFH2 described above.

Experiment 3.-In a third experiment, we investigate the possibility to generate models with the same architecture and accuracy of the reduced models FFH1b and FFH2b by directly training the 2-1 model given by (3.12). The nonlinear function is classically chosen as $\varphi_1(z) = \tanh(z)$ ([84]), $n_a = 4$, $n_b = 7$ and $n_k = 4$. One thousand training runs are conducted using the estimation data $(u(t), y(t)$ for $t = 1, 2, \cdots, 100000)$ each time with different initialization values. Once the 1000 models are obtained we choose the best one according to the validation error μ_t (model FFH3) and the best one according to the validation error e_{RMSt} (model FFH4). These two models have the same architecture of the models FFH1a and FFH2a (see (3.12)) where the 14 parameters characterizing the system are:

FFH3:

$W_A^* = [0.90585 \quad 0.1782 \quad 0.0715 \quad -0.34546]$
$W_B^* = [0.1166 \quad 0.2003 \quad 0.19789 \quad 0.18499 \quad -0.06139 \quad 0.2379 \quad 0.4065]$
$V_A^* = 1.1957,\ V_B^* = 0.0279,\ H^* = -0.0020$

FFH4:

$W_A^* = [0.5259 \quad 0.6859 \quad -0.0497 \quad -0.4054]$
$W_B^* = [0.06179 \quad 0.0847 \quad 0.2830 \quad -0.0110 \quad 0.1595 \quad -0.0633 \quad 0.4174]$
$V_A^* = 1.2582,\ V_B^* = 0.0466,\ H^* = -0.0024$

For the model FFH3 the validation results are $\mu_t = 0.07221 \times 10^{-3}$, $s_t = 91.62 \times 10^{-3}$, $e_{RMSt} = 91.62 \times 10^{-3}$ and $e_{RMSe} = 93.01 \times 10^{-3}$ and for the model FFH4 the validation results are $\mu_t = -0.9493 \times 10^{-3}$, $s_t = 80.29 \times 10^{-3}$, $e_{RMSt} = 80.30 \times 10^{-3}$ and $e_{RMSe} = 81.41 \times 10^{-3}$. The computational complexity for the models FFH3 and FFH4 is $O(FFH3) = O(FFH4) = (n_a + n_b + 4) \times N = (4 + 7 + 4) \times 100000 = 1.4 \times 10^6$.

Experiment 4.-In this experiment, in order to measure the impact of Assumption 2, the system is identified with the complex 2nn-2-1 model given by (3.1) with same characteristics as in the first experiment (the nonlinear function classically chosen as $\varphi_1(z) = \tanh(z)$ ([84]), $nn = 40$, $n_a = 4$, $n_b = 7$ and $n_k = 7$). One thousand training runs are conducted using the same data as in the first experiment (estimation data $(u(t), y(t)$ for $t = 1, 2, \cdots, 100000)$ each time with different initialization values but this time without satisfying Assumption 2 (that is, all the initial weights random). Once the 1000 models are obtained, we choose the best one according to the validation error μ_t (model FFH5) and the best one according to the validation error e_{RMSt} (model FFH6). Notice that, since the models FFH5 and FFH6 do not satisfy Assumption 2, it is not possible to apply the proposed model reduction approach. Consequently, these models have the same number of parameters (524) as the non reduced models FFH1 and FFH2 but with different validation results. For the model FFH5 the validation results are $\mu_t = -0.0103 \times 10^{-3}$, $s_t = 55.49 \times 10^{-3}$, $e_{RMSt} = 55.49 \times 10^{-3}$ and $e_{RMSe} = 54.70 \times 10^{-3}$ and for the model FFH6 the validation results are $\mu_t = -4.05 \times 10^{-3}$, $s_t = 53.23 \times 10^{-3}$, $e_{RMSt} = 53.39 \times 10^{-3}$ and $e_{RMSe} = 52.61 \times 10^{-3}$. The computational complexity for the models FFH5 and FFH6 is the same as for the model FFH1, FFH2, FFH1a and FFH2a, that is, $O(FFH5) = O(FFH6) = 52.4 \times 10^6$.

In order to follow the validation tests proposed in ([87]), we arranged the validation results of the proposed models in Table 5.1 where n_p is the number of parameters, O is the computational complexity or number of adaptations required to obtain each model, measured as presented in Chapter 3 and TT is the time in minutes required to obtain each model.

Table 5.1: Validation results.

Model	O $(\times 10^6)$	TT	np	μ_t $(\times 10^{-3})$	s_t $(\times 10^{-3})$	e_{RMSt} $(\times 10^{-3})$	e_{RMSe} $(\times 10^{-3})$
FFH1	52.4	12.78	524	0.00471	56.71	56.71	56.03
FFH1a	52.4	12.78	14	0.00471	56.71	56.71	56.03
FFH1b	1.7	3.85	14	0.00471	56.71	56.71	56.03
FFH2	52.4	12.78	524	−5.635	50.20	50.51	49.74
FFH2a	52.4	12.78	14	−5.635	50.20	50.51	49.74
FFH2b	1.7	3.85	14	−5.635	50.20	50.51	49.74
FFH3	1.4	3.55	14	0.07221	91.62	91.62	93.01
FFH4	1.4	3.55	14	−0.9493	80.29	80.30	81.41
FFH5	52.4	12.78	524	−0.0103	55.49	55.49	54.70
FFH6	52.4	12.78	524	−4.05	53.23	53.39	52.61

To complete the exposition of results for best model in terms of μ_t (FFH1b) in the required form, we also present the estimated output in the time domain (Fig. 5.3), as well as the fast Fourier transform (FFT) of the estimated output signal (Fig. 5.4). Further information is given through the frequency response function (FRF) of the nonparametric best linear approximation obtained from the test data and the estimated output (see. Fig. 5.5) which is mainly appreciated by the users in practical applications.

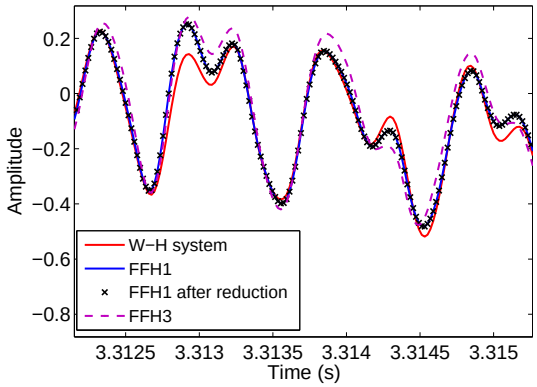


Figure 5.3: Measured output vs Estimated output.

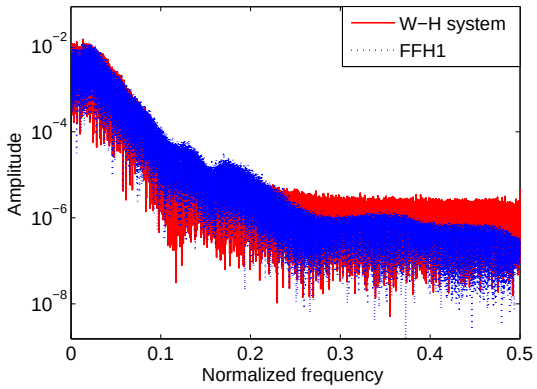


Figure 5.4: Output Fourier Transform.

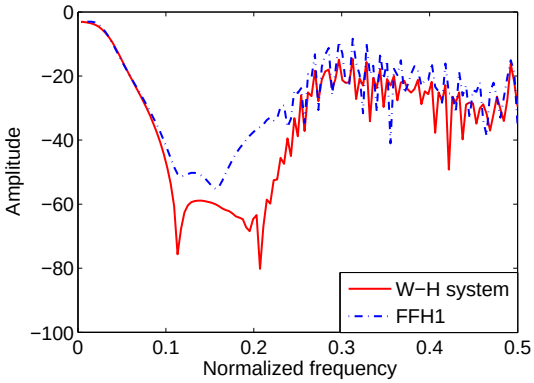


Figure 5.5: Frequency Response Function (FRF) of the nonparametric best linear approximation obtained from the test data and the estimated output.

Naturally we have to compare the our approach with other black box system identification techniques. For example, the same system is identified with the nonlinear system identification toolbox in Matlab 2011. Here, the identification model used is the NLARX model, the nonlinearity estimator is the sigmoid network and the parameters of the model are estimated by using the Levenberg-Marquardt algorithm, let us name this algorithm "MATLAB". The comparison, realized in terms of the number of parameters np, the time in minutes required to obtain each model TT and the validation results proposed in [87], are summarized in Table 5.2.

Table 5.2: Validation results: Proposed approach vs toolbox of Matlab.

Ref.	TT	n_p	μ_t $(\times 10^{-3})$	s_t $(\times 10^{-3})$	e_{RMSt} $(\times 10^{-3})$	e_{RMSe} $(\times 10^{-3})$
FFH1b	3.85	14	0.00471	56.71	56.71	56.03
MATLAB	1.52	691	-1.44	19.28	19.34	19.26

As already mentioned, the learning algorithms used to adapt the synaptic weights of the proposed neuro-models are based on the steepest descent algorithm. Although, the Levenberg-Marquardt algorithm was implemented in one of the proposed neural networks in order to analyze the compromise accuracy-computational cost. This is tested in the following experiment.

Experiment 5.-The system is identified with the complex 2nn-2-1 model (FFHH1) given by (4.15) presented in the example 2 in Chapter 4. The nonlinear function is classically chosen as $\varphi_1(z) = \tanh(z)$ ([84]), $nn = 40$, $n_a = 4$, $n_b = 7$ and $n_k = 4$. Two training runs are conducted using the estimation data $(u(t), y(t)$ for $t = 1, 2, \cdots, 100000)$ in both cases satisfying Assumption 2 (Initial weights equals group by group). In a first training, the parameters are estimated by using the steepest descent algorithm, let us name this model "FFHH1-GRA". In a second training, the parameters are estimated by using the Levenberg-Marquardt algorithm, let us name this model "FFH1-LM". In order to follow the validation tests proposed in ([87]), the validation results of the proposed "FFHH1-GRA" and "FFH1-LM" models are arranged in Table 5.3 where a comparison between the two neuro-models is done according the time in minutes required to obtain each model (TT) and the accuracy of each model.

Table 5.3: Performance measures: Gradient vs. Levenberg-Marquardt.

Model	TT	np	μ_t $(\times 10^{-3})$	s_t $(\times 10^{-3})$	e_{RMSt} $(\times 10^{-3})$	e_{RMSe} $(\times 10^{-3})$
FFHH1-GRA	3.69	16	-2.9	46.25	46.35	45.61
FFHH1-LM	21.82	16	-15.32	43.26	45.89	45.16

5.1.3 Comments

Fig. 5.3 exposes the relevant performance of the proposed neuro-model FFH1b with respect to the real system behavior, particularly if the number of model parameters are taking into consideration. Here it seems remarkable that with only 14 parameters the proposed model provides a significant level of accuracy.

From Table 5.1, comparing the best models according to the validation error μ_t (FFH1, FFH3 and FFH5), the validation results s_t, e_{RMSt}, e_{RMSe} of the complex models FFH1 and FFH5 given by (3.1) are almost the same. In the same way, comparing the best models according to the validation error e_{RMSt} (FFH2, FFH4 and FFH6), the validation results of the complex models FFH2 and FFH6 given by (3.1) are almost the same too. On the contrary, if we train directly the simpler 2-1 neural network models (FFH3 and FFH4), the accuracy of the obtained models is degraded. Since the 2nn-2-1 models given by (3.1) with $nn = 40$ have more neurons and one extra layer than the 2-1 model given by (3.12), these results are somehow natural. As mentioned above more neurons presume a better approximation.

Comparing in terms of the number of parameters n_p (see Table 5.1), for the models FFH1, FFH2, FFH5 and FFH6 it is substantially larger, 524 vs the 14 parameters of the models FFH3 and FFH4. But, after applying the proposed model reduction approach to the complex models FFH1 and FFH2 (524 parameters), we have the models FFH1b and FFH2b (14 parameters only) with the same accuracy as the previous models FFH1 and FFH2. These results confirm the theorem 1. Since models FFH5 and FFH6 do not satisfy Assumption 2, it is impossible to apply the model complexity reduction approach.

The reader shall notice that, as already announced, there is no loss of accuracy during the reduction approaches: see performances of FFH1, FFH1a and FFH1b (respectively FFH2, FFH2a and FFH2b).

From Table 5.2, confronting the two black box models, the deviation errors (s_t, e_{RMSt} and e_{RMSe}) and the time TT of the Matlab model are two or three times better, but the number of parameter is substantially larger (14 vs 691). Conversely, the proposed model FFH1a leads to a μ_t much better despite the use of a simple gradient training algorithm. Remember that we do not pretend to give to the user the reduced order architecture with best performance possible. A lot of high level works exist in the literature. We only propose to the user a rather simple and efficient way to find a good balance between accuracy and complexity. The users must ponder accuracy against complexity, according to their own modeling needs.

From Table 5.3, comparing both models, the one obtained by the steepest descent algorithm (FFHH1-GRA) and the one obtained by the Levenberg-Marquardt algorithm (FFHH1-LM), the validation results are almost the same, but comparing in terms of the time required to obtain each model, the FFHH1-LM model requires seven times more time than the FFHH1-GRA model.

5.2 Acoustic duct identification

This experimental device is an acoustic waves guide made of Plexiglas, used to develop an active noise control (see Fig. 5.6). One end of the duct is almost anechoic and the other end is opened. The identification input signal is a pseudo random binary sequence (PRBS) with a length $L = 2^{10} - 1$ and level $\pm 3V$ sufficiently exciting applied to the control loudspeaker. The sampling period is $TS = 500\mu s$. In order to model, the first propagative modes of the waves guide (also called the secondary path) which lye in the frequency range $[0; 1000Hz]$. We shall use several data set of the same length namely 1024, measured by the output microphone. A prior measurement of the propagation delay confirms the analytical value $\tau \approx 7TS$.

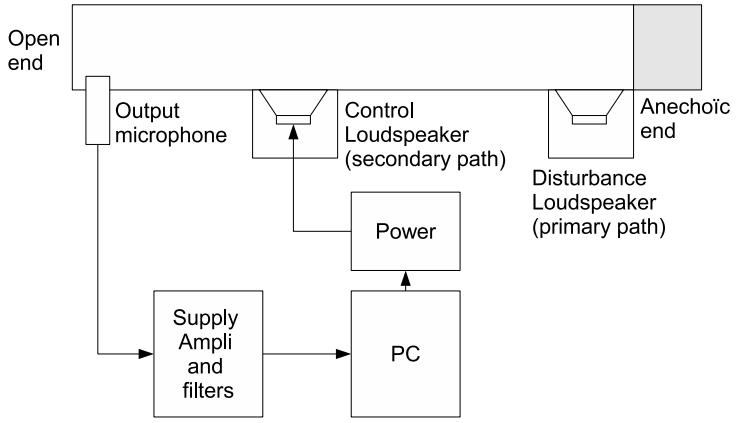


Figure 5.6: Schematic of semi finite acoustic waves guide.

This system is identified with the complex 2nn-2-1 FF1 model given by (4.2) with the nonlinear function classically chosen as $\varphi_1(z) = \tanh(z)$ ([84]), $nn = 10$, $n_a = 13$, $n_b = 11$ and $n_k = 7$. In order to fulfill Assumption 2, the initial conditions of the synaptic weights are chosen equals group by group, since the proposed model satisfies the two assumptions, the proposed model reduction approach is applied in order to obtain a model of the form of (5.6), let us name this model "FF1r".

$$\hat{y}(k) = V_B^* \tanh(J_u W_B^*) + V_A^* \tanh(J_{\hat{y}} W_A^*) \tag{5.6}$$

where:

$$\begin{aligned} J_u &= [u(k-1) \quad u(k-2) \quad ... \quad u(k-11)] \\ J_{\hat{y}} &= [\hat{y}(k-1) \quad \hat{y}(k-2) \quad ... \quad \hat{y}(k-13)] \end{aligned}$$

and the 26 parameters characterizing the system are:

$W_A^* = [0.202 \quad -0.192 \quad 0.157 \quad 0.261 \quad -0.038 \quad 0.041 \quad 0.089 \quad 0.011 \quad -0.029 \quad 0.072 \quad 0.074 \quad 0.029 \quad -0.008]$
$W_B^* = [-0.158 \quad 0.077 \quad 0.117 \quad 0.0373 \quad 0.087 \quad -0.017 \quad 0.038 \quad -0.024 \quad -0.023 \quad -0.007 \quad -0.088]$
$V_A^* = -2.7027$ and $V_B^* = 1.4493$

Naturally the proposed approach have to be compared with other black box system identification technique. Therefore, the same acoustic system is identified with the nonlinear system identification toolbox in Matlab 2011. Here, the identification model used is the NLARX model, the nonlinearity estimator is the sigmoid network with 20 units. The 1385 parameters of the

Matlab model are estimated by using the default learning algorithm. The comparison, realized in terms of the number of parameters np and the frequency Response function (FRF) (see. Fig. 5.7) which is mainly appreciated by the users in practical applications.

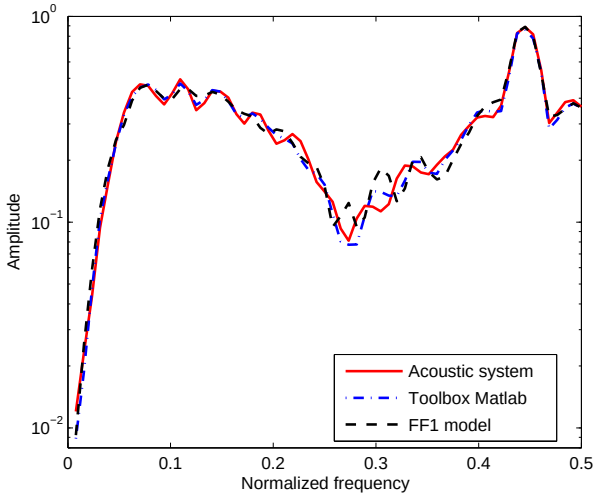


Figure 5.7: Frequency Response Function (FRF).

In order to establish a numerical comparison, the following measured (FITF) is computed from Fig. 5.7:

$$FITF = 100\left(1 - \frac{\|E_{x1} - E_{x2}\|}{\|E_{x1} - \bar{E_{x1}}\|}\right) \tag{5.7}$$

where:

$FITF$ is the percentage of the measured output that was explained by the model.
E_{x1} is the frequency response of the measured data.
E_{x2} is the frequency response of the estimated data.

The FITF is computed for both, the proposed model (FF1) and the model obtained by the nonlinear system identification toolbox of Matlab (Matlab), these results are arranged in Table 5.4.

Table 5.4: Proposed approach vs Matlab

Model	$FITF$	n_p
Matlab	93.95	1385
FF1	90.9	528
FF1r	90.9	26

5.2.1 Comments

Fig. 5.7 illustrates the relevant performance of both, the proposed neuro-model FFH1r and the Matlab model, with respect to the real system behavior. But, if the number of model parameters is taking into consideration, it seems remarkable that with only 26 parameters the proposed model provides a significant level of accuracy.

From Table 5.4, comparing the proposed model (FF1r) and the model obtained by using the toolbox of Matlab both models almost have the same accuracy, but once again, the number of parameters of the Matlab model is substantially larger (1385 vs 26). Then, we are proposing to the user a system identification approach with results comparable to the powerful toolbox of Matlab, but with a smaller number of parameters.

5.3 Robot arm identification

The idea here is to test the efficiency of our method in the case, frequently encountered, of identification with corrupted data. In fact, we take the opportunity to check the robustness of our estimation procedure even if we do not pretend to exhaustively study this nevertheless interest configuration.

As similarly, in this work both the classical MSE criterion and a robust criterion known as Huber's function are implemented for the adaptation of the synaptic weights, these criteria are tested in this experiment.

The data comes from a flexible robot arm, the arm is installed on an electrical motor. The applied persisting exciting input, corresponding to the reaction torque of the structure on the ground is a periodic sine sweep. The output of such system, is the acceleration of the flexible arm. This system identification case is issued from a example through DaISy (Database for the Identification of Systems) *ftp://ftp.esat.kuleuven.be/pub/SISTA/data/mechanical*.

Figure 5.8: Flexible robot arm.

In order to test the performances of the proposed objective functions, i.e. MSE and Huber functions, two outliers with random levels and time steps are deliberately inserted in the dataset as presented in Fig. 5.9.

Once the outliers were inserted in the dataset, the system is identified with the proposed neuro-model NARX given by (4.5) with the nonlinear function classically chosen as $\varphi_1(z) = \tanh(z)$ ([84]), $nn = 40$, $n_a = 8$, $n_b = 7$ and $n_k = 1$. In a first test, the system is identified by using the classical MSE criterion, let us call this model "NLARX-MSE". In a second test, the Huber function is used as objective function, let us name this model "NLARX-HUBER".

Once the models are trained under Assumption 1 and Assumption 2, Theorem 1 (reduction approach) and Theorem 2 (computational cost reduction approach) are applied in order to generate two models of the form of (5.8).

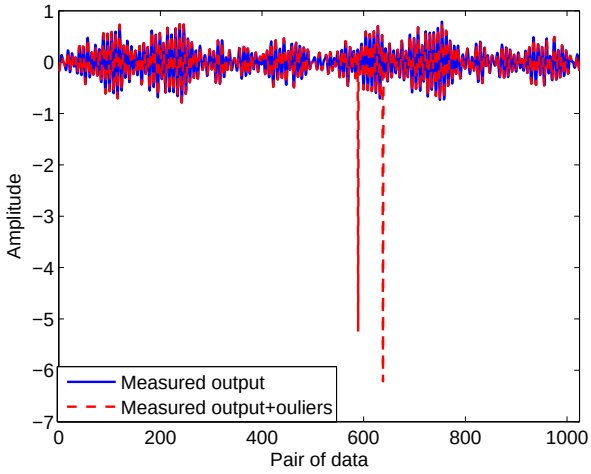


Figure 5.9: Measured output vs. Output corrupted by outliers.

$$\hat{y}(k) = X^* \tanh(J_u W_B^* + J_{\hat{y}} W_A^*) \tag{5.8}$$

where the 16 parameters characterizing the system are:

NLARX-HUBER model:

$$W_B^* = [-0.2955 \quad 0.4292 \quad -0.1253 \quad -0.1296 \quad 0.1434 \quad 0.1353 \quad -0.1988]$$
$$W_A^* = [1.0192 \quad -0.2565 \quad -0.1988 \quad -0.2512 \quad 0.1184 \quad 0.1909 \quad 0.0515 \quad -0.3213]$$
$$X^* = 1.3320$$

NLARX-MSE model:

$$W_B^* = [0.3025 \quad -0.4034 \quad 0.0652 \quad 0.1589 \quad -0.1102 \quad -0.2060 \quad 0.2388]$$
$$W_A^* = [-0.6727 \quad -0.2074 \quad 0.3380 \quad 0.4301 \quad -0.1481 \quad -0.4000 \quad 0.1048 \quad 0.2999]$$
$$X^* = -1.4704$$

In order to compare the performances of both models, the frequency Response function (FRF) which is mainly appreciated by the users in practical applications is computed from the measured data and the estimated output of both, the NLARX-MSE model and the NLARX-HUBER model (see Fig. 5.10).

Comments

Fig. 5.10 exposes the interest of the implementation of the Huber function, since in the presence of outliers, the performance of the model estimated by using the steepest descent algorithm (NLARX-MSE) is degraded. On the contrary the model NLARX-HUBER, accurately represent the system behavior, in particular in low frequency where generally the control objectives are attended.

In this chapter, we have shown how practically the user may apply our new identification methodology, both on simulation application and real experiment one. Moreover, we have shown the interest of applying this method to various configurations, covering nonlinear complex system identification as well as robust system identification.

The quality of the results presented validates the performance of our new complex system identification methodology in terms of our non announced motivation: the search of the best compromise between accuracy, complexity and computational cost.

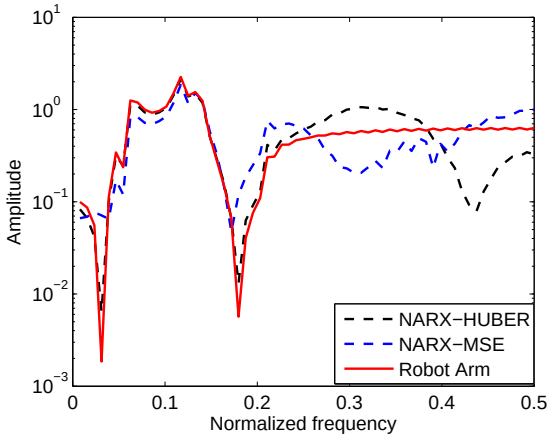


Figure 5.10: Frequency Response Function (FRF) of the nonparametric best linear approximation obtained from the test data and the estimated output.

Chapter 6

Conclusions and perspectives

6.1 Conclusions

This thesis is based on a work done in the framework of a co-direction between the Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), Cuernavaca, Morelos, Mexico and the Ecole Nationale Suprieure d'Arts et Mtiers (ENSAM), Centre d'Aix en Provence, France. The report concerns the research topic of black box nonlinear system identification entering into the academic collaboration between these two institutions.

In effect, among all the various and numerous techniques developed in this field of research these last decades, it seems still interesting to investigate the neural network approach in complex system model estimation. Obviously neural networks represents a powerful tool since numerous years for black box nonlinear system identification. Even if accurate models have been derived, the main drawbacks of these techniques remain the large number of parameters required and, as a consequence, the important computational cost necessary to obtain the convenient level of the model accuracy desired. Therefore, neural networks are still often considered as complex and hard to implement in real applications.

Hence, motivated to address these drawbacks, we achieved a complete and efficient system identification methodology providing balanced accuracy/complexity/cost models. In effect, it is well known that trying to improve one of the three preceding items implies a significant deterioration of the two other ones. In this work, we demonstrated that it is possible to obtain such a good compromise by the development of, firstly, new neural network structures particularly adapted to a very wide use in practical nonlinear system modeling, secondly, a simple and efficient model reduction technique, and, thirdly, a computational cost reduction procedure. It is important to notice that these last two reduction techniques can be applied to a very large range of neural network architectures under two simple specific assumptions which are not at all restricting. In effect, the first assumption which is related to the choice of the activation functions is classical in the neural network theory. Concerning the second assumption, this is an original idea which allows to achieved the two reduction procedures proposed in this thesis, as it has been shown by experiments, this assumption does not affect the accuracy of the neural network.

Moreover, the proposed model complexity reduction procedure does not carry along a loss of accuracy during the model transformation, as it classically happens when other traditional techniques are used. Even more, the accuracy thus obtained remains significantly better than the one obtained by directly estimating, in the same conditions, a model with a comparable low complexity.

In particular, due to the originality of this approach, the number of neurons initially chosen to identify a system, does not affect the complexity of the obtained reduced models. In effect, the first training determines the parameters of the model and the model accuracy as a consequence, so that no additional criterion is required to stop the model simplification and no further loss of accuracy occurs.

Moreover, the proposed computational cost reduction approach leads to a very significant decrease of the calculation bulk classically needed. Since, in this approach, the computational cost does not depend on the number of neurons chosen by the designer as it happens when classical techniques are used.

Concerning the optimization techniques used for the parameters estimation, the synaptic weights adaptation rules are based on the steepest descent algorithm. Even if the Levenberg-Marquardt algorithm has been tried for at least one architecture, the application examples demonstrate that the compromise between accuracy and the computational cost remains better when the steepest descent algorithm is used.

Concerning the choice of the optimization criterion, the classical MSE and the robust Huber M-estimate criteria were checked. The main reason was that in applications where data are significantly corrupted, the performances of the MSE estimator are classically degraded. In these cases, the resort of the Huber's approach is always recommended regardless the slight extra computational cost required.

Furthermore, application examples driven in simulation or on a real process, satisfactorily validate all the preceding results, confirming all the interest of such a new nonlinear identification methodology.

To summarize, we proposed a new complete identification methodology using a neural network approach for complex systems guaranteeing a good compromise between model accuracy, model complexity and model computational cost. The main contributions of this work is firstly an original choice of the parameterized model set within which the estimation of the candidate model is performed. The second important contribution is the improved model computation, where the neural network training is achieved with an original and efficient reduction procedure. Finally, the last important contribution of this work is to have shown that this estimation phase can be achieved in a robust framework if the quality of identification data compels it.

6.2 Perspectives

During our investigations, we noticed an interesting reduction of the number of epochs required to train the neural network, mainly due to a particular selection of the initial conditions of the synaptic weights. This choice depends on the estimation dataset and on the neural network architecture chosen. It seems interesting to investigate this point more formally.

Another research direction to be more finally investigated is the choice of the scaling factor of the robust M-estimator used in case of corrupted data. In particular, the influence of this tuning constant defining the ratio of estimation errors treated by the L1-norm w.r. the estimation errors treated by the classical L2-norm, on the global identification performances such as model accuracy, model complexity and model computational cost might be developed in a next future.

Bibliography

[1] K. Hangos, J. Bokor and G. Szederknyi. *Analysis and control of nonlinear process systems.* Springer, 2004.

[2] P. Aadaleesan, N. Miglan, R. Sharma and P. Saha. Nonlinear system identification using wiener type laguerre-wavelet network model. *Chemical Engineering Science*, 63(15)(2008) 3932–3941, http://dx.doi.org/10.1016/j.ces.2008.04.043.

[3] L. Coelho and M. Wicthoff. Nonlinear identification using a b-spline neural network and chaotic immune approaches. *Mechanical Systems and signal Processing*, 23(8)(2009) 2418–2434, http://dx.doi.org/10.1016/j.ymssp.2009.01.013.

[4] F. Farivar, M. A. Shoorehdeli and M. Teshnehlab. An interdisciplinary overview and intelligent control of human prosthetic eye movements system for the emotional support by a huggable pet-type robot from a biomechatronical viewpoint. *Journal of the Franklin Institute*, 349(7)(2012) 2243–2267, http://dx.doi.org/10.1016/j.jfranklin.2011.04.014S.

[5] M. Noorgard, O. Ravn, N. K. Poulsen and L. K. Hansen. *Neural Networks for Modelling and Control of Dynamic Systems.* 1st ed., Springer-Verlag, London, Berlin, Heidelberg, Great Britain,2000.

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

[7] Z. Yan,L. Xiuxia,Y. Peng,C. Zengqiang and Y. Zhuzhi. Modeling and control of nonlinear discrete-time systems based on compound neural networks. *Chinese Journal of Chemical Engineering*,17 (3) (2009) 454–459,http://dx.doi.org/10.1016/S1004-9541(08) 60230-X.

[8] S.C. Tong,Y.M. Li and H.G. Zhang. Adaptive neural network decentralized backstepping output-feedback control for nonlinear large-scale systems with time delays. *IEEE Transactions on Neural Networks*, 22 (7) (2011) 1073-1086.

[9] S.C. Tong,X.L. He and H.G. Zhang. A combined backstepping and small-gain approach to robust adaptive fuzzy output feedback control. *IEEE Transactions on Fuzzy Systems*, 17 (5) (2009) 1059-1069.

[10] R. Isermann and M. Munchhof. *Identification of Dynamic Systems, An Introduction with Applications.* Springer-Verlag, 2011.

[11] L. Ljung. *System Identification Theory For The User.* PTR Prentice Hall, 1999.

[12] L. dos Santos and M. Wicthoff. Nonlinear model identification of an experimental ball-and-tube system using a genetic programming approach. *Mechanical systems and signal*, 23 (8) (2009) 2418–2434, http://dx.doi.org/10.1016/j.ymssp.2009.01.013.

[13] L. dos Santos and M. Wicthoff. Nonlinear system identification and fault detection using hierarchical clustering analysis and local linear models. *Mediterranean Conference on Control and Automation.*, Athens 2007, http://dx.doi.org/10.1109/MED.2007.4433938.

[14] W. Luyben. *Process, Modeling, Simulation and Control for Chemical Engineers.* McGraw Hill Chemical Engineering Series, 2nd edition, 1996.

[15] B. Anderson, J. Moore and R. Hawkes. Model approximation via prediction error identification. *Automatica*, 14 (6) (1978) 615–622 http://dx.doi.org/10.1016/0005-1098(78)90051-1.

[16] L. Ljung and P. Caines. Asymptotic normality of prediction error estimators for approximative system models. *Stochastics*, 17 (1979) 29–46 http://dx.doi.org/10.1109/CDC.1978.268066.

[17] O. Nelles. *Nonlinear System Identification.* Springer-Verlag, Berlin, Heidelberg New York, Germany, 2001.

[18] J. Jang, C. Sun and E. Mizutani. *Neuro-Fuzzy and Soft Computing.* Prentice-Hall, Upper Saddle River, NJ 07458, 1997.

[19] Hua Bai, Pei Zhang, and Venkataramana Ajjarapu. A novel parameter identification approach via hybrid learning for aggregate load modeling. *IEEE Transactions on Power Systems*, 24 (3)(2009) 1145–1154, http://dx.doi.org/10.1109/TPWRS.2009.2022984.

[20] Jose de Jesus Rubio and Jaime Pacheco. An stable online clustering fuzzy neural network for nonlinear system identification. *Neural Computing and Applications*, 18 (2009) 633–641.

[21] W. Yu and A. Morales. Gasoline blending system modeling via static and dynamic neural networks. *International journal of modeling and simulation*, 24(3) (2004) 151–160.

[22] X. Han, W. Xie, Z. Fu, and W. Luo. Nonlinear systems identification using dynamic multi-time scale neural networks. *Neurocomputing*, 74(17) (2011) 3428–3439 http://dx.doi.org/10.1016/j.neucom.2011.06.007.

[23] A. Cichocki and R. Unbehauen. *Neural Networks for Optimization and Signal Processing.* 1st edition, John Wiley and Sons Ltd, Baffins Lane, Chichester, West Sussex, England, 1993.

[24] A. Poznyak, E. Sanchez, and W. Yu. *Differential Neural Networks for Robust Nonlinear Control.* 1st edition, World Scientific Publishing Co. Pte. Ltd, Singapore, 2001.

[25] E. Fieser and R. Beale. *Handbook of Neural Computation.* Institute of Physics Publishing and Oxford University Press, New York, 1997.

[26] G. Cybenko. Neural networks in computational science and engineering. *Neural Network in CSE*, 3 (1996) 36-42.

[27] H. Ge, W. Du, F. Qian, and Y. Liang. Identification and control of nonlinear systems by a time-delay recurrent neural network. *Neurocomputing*, 72 (2009) 2857-2864, http://dx.doi.org/10.1016/j.neucom.2008.06.030.

[28] S. Tzeng. Design of fuzzy wavelet neural networks using the ga approach for function approximation and system identification. *Fuzzy Sets and Systems*, 161 (19) (2010) 2585-2596, http://dx.doi.org/10.1016/j.fss.2010.06.002.

[29] B. Subudhi and D. Jenab. A differential evolution based neural network approach to nonlinear system identification. *Applied Soft Computing*, 11 (1) (2011) 861-871, http://dx.doi.org/10.1016/j.asoc.2010.01.006..

[30] W. Xie, Y. Zhu, Z. Zhao, and Y. Wong. Nonlinear system identification using optimized dynamic neural network. *Neurocomputing*, 72 (13-15) (2009) 32773287, http://dx.doi.org/10.1016/j.neucom.2009.02.004.

[31] W. Yu and Jose de Jesus Rubio. Recurrent neural networks training with stable bounding ellipsoid algorithm. *IEEE Transactions on Neural Networks*, 10 (6) (2009) 983–991, http://dx.doi.org/10.1016/10.1109/TNN.2009.2015079.

[32] G. Khalaj, H. Yoozbashizadeh, A. Khodabandeh and A. Nazari Artificial neural network to predict the effect of heat treatments on vickers microhardness of low-carbon nb microalloyed steels. *Neural Computing and Applications*, (2011) 1–10, http://dx.doi.org/10.1007/s00521-011-0779-z.

[33] H. Ge, F. Qian, Y. Liang, W. Du and L. Wang. Identification and control of nonlinear systems by a dissimilation particle swarm optimization-based elman neural network. *Nonlinear Analysis: Real World Applications*, 9 (4) (2008) 1345-1360, http://dx.doi.org/10.1016/j.nonrwa.2007.03.008.

[34] Jose de Jesus Rubio and W. Yu. Nonlinear system identification with recurrent neural networks and dead-zone Kalman filter algorithm. *Neurocomputing*, 70 (2007) 2460–2466, http://dx.doi.org/10.1016/j.neucom.2006.09.004,

[35] Xueli Wu, Jianhua Zhang and QuanminZhu. A generalized procedure in designing recurrent neural network identification and control of time-varying-delayed nonlinear dynamic systems. *Neurocomputing*, 73 (79) (2010) 1376-1383, http://dx.doi.org/10.1016/j.neucom.2009.12.002.

[36] D. P. Mandic and J. A. Chambers. *Recurrent Neural Networks for Prediction: Learning Algorithms,Architectures and Stability*. John Wiley and Sons, Inc. New York, NY, USA 2001.

[37] Haiquan Zhao, Xiangping Zeng, and Zhengyou He. Low-complexity nonlinear adaptive filter based on a pipelined bilinear recurrent neural network. *IEEE Transactions on Neural Networks*, 22(9) (2011) 1494–1507.

[38] Jeen-Shing Wang, Yu-Liang Hsu, Hung-Yi Lin, and Yen-Ping Chen. Minimal model dimension/order determination algorithms for recurrent neural networks. *Pattern Recognition Letters*, 30:812 – 819, 2009.

[39] Haiquan Zhao and Jiashu Zhang. Nonlinear dynamic system identification using pipelined functional link artificial recurrent neural network. *Neurocomputing*, 72 (1315) (2009) 30463054, http://dx.doi.org/10.1016/j.neucom.2009.04.001.

[40] M. Witters and J. Swevers. Black-box model identification for a continuously variable, electro-hydraulic semi-active damper. *Mechanical Systems and Signal Processing*, 24 (1) (2010) 4-18, http://dx.doi.org/10.1016/j.ymssp.2009.03.013.

[41] J. Paduart, L. Lauwers, R. Pintelon, and J. Schoukens. Identification of a Wiener-Hammerstein system using the polynomial nonlinear state space approach, *in: Proceedings of the 15th IFAC Symposium on System Identification*, Saint-Malo, France, 2009, http://dx.doi.org/10.3182/20090706-3-FR-2004.00179.

[42] L. Ljung. Perspectives on system identification. *In Plenary Talk at the 17th IFAC World Congress.* July 2008.

[43] L. Ljung. Perspectives on system identification. *Annual Reviews in Control*, 34 (1) (2012) 1–12, http://dx.doi.org/10.1016/j.arcontrol.2009.12.001.

[44] G. Bebis and M. Georgiopoulos. Feed-forward neural networks: why network size is so important. *IEEE Potentials*, 13 (4) (1994) 27-31.

[45] W. Yu and X. Li. Fuzzy identification using fuzzy neural networks with stable learning algorithms. *IEEE Transactions on Fuzzy Systems*, 12 (3) (2004) 411–420, http://dx.doi.org/10.1109/TFUZZ.2004.825067.

[46] Jinzhu Peng and Rickey Dubay. Identification and adaptive neural network control of a dc motor system with dead-zone characteristics. *ISA Transactions*, 50 (4) (2011) 588 – 598, http://dx.doi.org/10.1016/j.isatra.2011.06.005.

[47] J. de Jesus Rubio, P. Angelov and J. Pacheco. Uniformly stable backpropagation algorithm to train a feedforward neural network. *IEEE Transactions on Neural Networks*, 22 (3) (2011) 356–366, http://dx.doi.org/10.1109/TNN.2010.2098481.

[48] J. de Jesus Rubio, J. Humberto Perez-Cruz, A. Y. Alanis and Jaime Pacheco. System identification using multilayer differential neural networks: A new result. *Journal of Applied Mathematics*, (2012), http://dx.doi.org/10.1155/2012/529176.

[49] H. Zhang, W. Wu, and M. Yao. Boundedness and convergence of batch back-propagation algorithm with penalty for feedforward neural networks. *Neurocomputing*, 89 (2012) 141-146, http://dx.doi.org/10.1016/j.neucom.2012.02.029.

[50] S.-K. Oh and W. Pedrycz Genetic optimization-driven multi-layer hybrid fuzzy neural networks. *Simulation Modelling Practice and Theory*, 14 (5) (2006) 597-613, http://dx.doi.org/10.1016/j.simpat.2005.10.009.

[51] S. Loghmanian, H. Jamaluddin, R. Ahmad, R. Yusof and M. Khalid. Structure optimization of neural network for dynamic system modeling using multi-objective genetic algorithm. *Neural Computing and Applications*, (2011) 1–15.

[52] Y. Le Cun, J. Denker and S.A. Solla. Optimal brain damage. *Neural Information Processing Systems-2*, (1990) 598 – 605.

[53] B. Hassibi, D. G. Stork and G. J.Wolff. Optimal brain surgeon and general network pruning. *IEEE International Conference on Neural Networks*, 1 (1993) 293 – 299, http://dx.doi.org/10.1109/ICNN.1993.298572.

[54] R. Reed. Pruning algorithms–a survey. *IEEE Transactions on Neural Networks*, 4(5) (1993) 740–747.

[55] Wen Yu, F. O. Rodriguez, and M. A. Moreno-Armendariz. Hierarchical fuzzy cmac for nonlinear systems modeling. *IEEE Transactions on Fuzzy Systems*, 16(5) (2008) 1302–1314, http://dx.doi.org/10.1109/TFUZZ.2008.926579.

[56] J. de Jesus Rubio. Sofmls: Online self-organizing fuzzy modified least-squares network. *IEEE Transactions on Fuzzy Systems*, 17(6) (2009) 1296-1309, http://dx.doi.org/10.1109/TFUZZ.2009.2029569.

[57] Zhaozhao Zhang and Junfei Qiao. A node pruning algorithm for feedforward neural network based on neural complexity. *in:Proceedingsofthe2010 International Conference on, Intelligent Control and Information Processing (ICICIP)*, Dalian, China, 2010, http://dx.doi.org/10.1109/ICICIP.2010.5564272.

[58] P. Angelov. Fuzzily connected multimodel systems evolving autonomously from data streams. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41(4) (2011) 898–910, http://dx.doi.org/10.1109/TSMCB.2010.2098866.

[59] A. Lemos, W. Caminhas, and F. Gomide. Multivariable gaussian evolving fuzzy modeling system. *IEEE Transactions on Fuzzy Systems*, 19 (1) (2011) 91–104, http://dx.doi.org/10.1109/TFUZZ.2010.2087381.

[60] Liu Biao, Lu Qing-chun, Jin Zhen-hua, and Nie Sheng-fang. System identification of locomotive diesel engines with autoregressive neural network. *4th IEEE Conference on Industrial Electronics and Applications ICIEA* Xi'an, CHINA, 2009, http://dx.doi.org/10.1109/ICIEA.2009.5138836.

[61] Christian Endisch, Peter Stolze, Peter Endisch, Christoph Hackl, and Ralph Kennel. Levenberg-marquardt-based obs algorithm using adaptive pruning interval for system identification with dynamic neural networks. *IEEE International Conference on Systems, Man and Cybernetics SMC* San Antonio, Texas, USA, 2009, http://dx.doi.org/10.1109/ICSMC.2009.5346186.

[62] S.-K. Oh and W. Pedrycz. Genetic optimization-driven multi-layer hybrid fuzzy neural networks. *Simulation Modelling Practice and Theory*, 14 (5) (2006) 597-613, http://dx.doi.org/10.1016/j.simpat.2005.10.009.

[63] C. K. Goh, E.J. Teoh, and K.C. Tan. Hybrid multiobjective evolutionary design for artificial neural networks. *IEEE Transactions on Neural Networks*, 19 (9) (2008) 15311548.

[64] Ahmad T. Abdulsadda and Kamran Iqbal. An improved spsa algorithm for system identification using fuzzy rules for training neural networks. *International Journal of Automation and Computing*, 6 (3) (2011) 333–339, http://dx.doi.org/10.1007/s11633-011-0589-x.

[65] B. Subudhi, D. Jena and M.M. Gupta. Memetic differential evolution trained neural networks for nonlinear system identification. *2008 IEEE Region 10 Colloquium and the Third International Conference on Industrial and Information Systems* Kharagpur, INDIA, 2008, http://dx.doi.org/10.1109/ICIINFS.2008.4798417.

[66] B. Subudhi and D. Jenab. Nonlinear system identification using memetic differential evolution trained neural networks. *Neurocomputing*, 2 (1315) (2009) 3277-3287, http://dx.doi.org/10.1016/j.neucom.2009.02.004.

[67] H. Romero. Identificacin de sistemas utilizando redes neuronales. Master's thesis, Centro Nacional de Investigacin y Desarrollo Tecnolgico, 2008.

[68] H. M. Romero Ugalde, J.-C. Carmona, V. M. Alvarado, and J. Reyes-Reyes. Neural network design and model reduction approach for black box nonlinear system identification with reduced number of parameters. *Neurocomputing*, 101 (2013) 170–180, http://dx.doi.org/10.1016/j.neucom.2012.08.013.

[69] H. M. Romero Ugalde, J.-C. Carmona, and V. M. Alvarado. 1/2 Nonlinear system identification: a balanced accuracy/complexity neural network approach. *in: 2nd International Conference on Communications Computing and Control Applications* Marseille, France, 2012.

[70] H. M. Romero Ugalde, J.-C. Carmona, and V. M. Alvarado. 2/2 Training time optimization for balanced accuracy/complexity neural network models. *in: 2nd International Conference on Communications Computing and Control Applications* Marseille, France, 2012.

[71] S.Q. An, T. Lu and Y.J. MA. Simple adaptive control for siso nonlinear systems using neural network based on genetic algorithm. *IEEE Proceedings of the Ninth International Conference on Machine Learning and Cybernetics*, Qingdao, China, 2010.

[72] T. Fukuda and T. Shibata. Theory and application of neural networks for industrial control systems. *IEEE Transactions on Industrial Electronics,* , 39 (6) (1992) 472–489.

[73] R. Lippman. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4 (1987).

[74] O. Jovanovic. Identification of dynamic system using neural network. *The scientific Journal, Architecture and Civil Engineering*, 1(4)(1998) 525–532.

[75] G. Rajesh and S. Bhattacharyya. System identification for nonlinear maneuvering of large tankers using artificial neural network. *Applied Ocean Research*, 30 (2008) 256–263.

[76] E. I. Gaura, N. Steele, and R. J. Rider. A neural network approach for the identification of micromachined accelerometers. *In: Proceedings of the Second International Conference on Modeling and Simulation of Microsystems*, 1999.

[77] F. Guerra and L. dos Santos. Multi-step ahead nonlinear identification of lorenzs chaotic system using radial basis neural network with learning by clustering and particle swarm optimization. *Chaos, Solitons & Fractals*, 35 (5) (2008) 967-979, http://dx.doi.org/10.1016/j.chaos.2006.05.077.

[78] F. Mohamed and H. Koivo. Modelling of induction motor using non-linear neural network system identification. *SICE Annual Conference in Sapporo, Control Eng. Lab.* Sapporo, August 2004.

[79] Y.G. Lee. Three phese active rectifier power conditioning using neural network system identification. *IEEE, Detarment of Electrical Engineering, INHA Technical College.* Korea, 1998.

[80] D. Hyland, E. Collins, W. Haddad, and D. Hunter. Neural network system identification for improved noise rejection. In *American Control Conference Seattle*, Washington, 1995.

[81] L. Kiong, M. Rajeswari, and M. Rao. Nonlinear dynamic system identification and control via constructivism inspired neural network. *Applied Soft Computing*, 3 (3) (2003) 237-257, http://dx.doi.org/10.1016/S1568-4946(03)00037-1.

[82] N. Messai, B. Riera, and J. Zaytoon. Identification of a class of hybrid dynamic systems with feed-forward neural networks: About the validity of the global model. *Nonlinear Analysis: Hybrid Systems*, 2 (3) (2008) 773-785, http://dx.doi.org/10.1016/j.nahs.2007.11.008.

[83] S. Mohanty. Artificial neural network based system identification and model predictive control of a flotation column. *Journal of Process Control*, 19 (6) (2009) 991-999, http://dx.doi.org/10.1016/j.jprocont.2009.01.001.

[84] S. Curteanu and H. Cartwright. Neural networks applied in chemistry. i. determination of the optimal topology of multilayer perceptron neural networks. *Journal of Chemometrics*, 25 (10)(2011) 527–549, http://dx.doi.org/10.1002/cem.1401.

[85] S. Purwar, I.N. Kar, and A.N. Jha. On-line system identification of complex systems using chebyshev neural networks. *Applied Soft Computing*, 7 (2007) 362 – 372, http://dx.doi.org/10.1016/j.asoc.2005.08.001.

[86] U. Flores. Identificacin de sistemas no lineales mediante las estructuras narx y hammerstein-wiener. Master's thesis, Centro Nacional de Investigacin y Desarrollo Tecnolgico, 2011.

[87] J. Schoukens, J. Suykens, and L. Ljung. Wiener-hammerstein benchmark. *in:15th IFAC Symposium on System Identification* Saint-Malo, France, 2009.

Appendix A

Activation functions

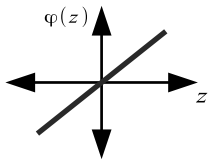
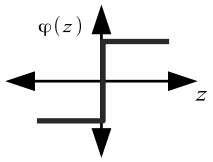
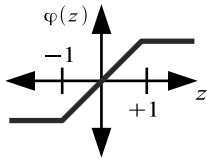
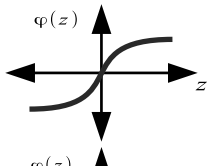
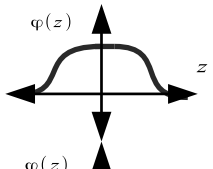
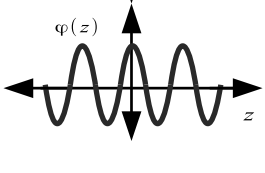
Name	Function	Range
Linear	$\varphi(z)=z$ 	$[-\infty,+\infty]$
Signo	$\varphi(z)=signo(z)$ $\varphi(z)=H(z)$ 	$\{-1,+1\}$ $\{0,+1\}$
Piecewise Linear	$\varphi(z)=\begin{cases} -1, & if \quad z<l \\ z, & if \quad +l\leqslant z\leqslant -l \\ +1, & if \quad z>+l \end{cases}$ 	$[-1,+1]$
Sigmoid	$\varphi(z)=\dfrac{1}{1+e^{-z}}$ $\varphi(z)=\tanh(z)$ 	$[0,+1]$ $[-1,+1]$
Gaussian	$\varphi(z)=A\,e^{-Bz^2}$ 	$[0,+1]$
Sinusoidal	$\varphi(z)=A\sin(\omega z)$ 	$[-1,+1]$

Figure A.1: Activation functions.

The most popular activation function in bibliography is the function $\tanh(\bullet)$ due to it is a saturation type function and have the interesting property that its derivative can be expressed as a simple function of its output $1-\tanh^2(\bullet)$ [17, 23, 24, 25]. This derivative is required in any gradient-based optimization technique.

Appendix B

Learning algorithms

The learning algorithm is an optimization algorithm used to adapt the synaptic weights of a defined neural network architecture. The goal of this adaptation is to minimize an objective function (MSE, Huber's function, etc). As mentioned above, if the data set used for the identification of a system is contaminated with outliers, robust functions such as the Huber's function is recommended regardless the computational cost required. If the data set is not contaminated, the quadratic criterion mays lead to a good accuracy.

Let us suppose a neuro-model parametrized by the synaptic weights w. The special case of the quadratic criterion is given by (1.2).

$$E(w, u^N, y^N) = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{2}e^2(k, w) \tag{B.1}$$

where N is the total number of data used for the parameters estimation and the prediction error $e(k, \hat{w})$ is:

$$e(k, \hat{w}) = y(k) - \hat{y}(k, \hat{w}) \tag{B.2}$$

A general family of algorithms to adapt the synaptic weights (w) is given by (B.3).

$$\hat{w}(k+1) = \hat{w}(k) - \eta[R]^{-1}\frac{\partial E}{\partial \hat{w}} \tag{B.3}$$

where R modifies the search direction, η is the step size known as leaning rate, $\frac{\partial E}{\partial \hat{w}}$ is the gradient of the objective function and $\hat{w}(k)$ denotes the value of $\hat{w}$ in the k_{th} iteration.

Here different algorithms (Steepest descent, Gauss-Newton, Levenberg-Marquardt, etc) can be derived in order to estimate the model parameters w.

The simplest selection $R = 1$ yields the steepest descent algorithm (B.4):

$$\hat{\theta}(k+1) = \hat{\theta}(k) - \eta\frac{\partial E}{\partial \hat{\theta}} \tag{B.4}$$

Here, the parameter η is commonly referred to as the learning rate or integration step size. It should be pointed that for the discrete time steepest descent algorithm, η should be bounded in a small range, to ensure the stability of the algorithm. It should be noted that a small η means that the convergence to a solution is slow while a large η means that the oscillations may occur and stability may be lost.

The choice $R = E^{''}(\hat{w})$ produces the damped Gauss-Newton algorithm (B.5).

$$\hat{w}(k+1) = \hat{w}(k) - \eta \left[\frac{\partial^2 E}{\partial \hat{w}^2}\right]^{-1} \frac{\partial E}{\partial \hat{w}} \tag{B.5}$$

Here, the particular choice of $\eta = 1$ leads the Gauss-Newton algorithm (B.6).

$$\hat{w}(k+1) = \hat{w}(k) - \left[\frac{\partial^2 E}{\partial \hat{w}^2}\right]^{-1} \frac{\partial E}{\partial \hat{w}} \tag{B.6}$$

The Levenberg-Marquardt procedure is achieved by choosing $\eta = 1$ and $R = E^{''}(\hat{w}) - \lambda I$ in (B.3), where λ is a positive scalar that is used to control the convergence in the iterative scheme rather than the step size parameter (η), and I is the identity matrix.

$$\hat{w}(k+1) = \hat{w}(k) - \left[\frac{\partial^2 E}{\partial \hat{w}^2} - \lambda I\right]^{-1} \frac{\partial E}{\partial \hat{w}} \tag{B.7}$$

Notice that with $\lambda = 0$ we have the gauss newton case (B.6).

Appendix C

Objective functions

Let us suppose a neuro-model parametrized by the synaptic weights w. As mentioned above, the adaptation rules of the synaptic weights depends on a objective function (or criterion) where the derivative is required by the gradient-based algorithms. This functions and the corresponding derivatives are:

Mean Square Error (C.1):

$$E(w,k) = \frac{1}{2}e^2(k,w) \tag{C.1}$$

MSE derivative:

$$\frac{\partial E}{\partial w} = e(w,k) \tag{C.2}$$

Huber's function (C.3):

$$E(w,k) = \begin{cases} \frac{1}{2}e^2(w,k) & ,if \mid e(w,k) \mid \leqq \beta, \\ \beta \mid e(w,k) \mid -\frac{\beta^2}{2} & ,if \mid e(w,k) \mid > \beta \end{cases} \tag{C.3}$$

Huber derivative:

$$\frac{\partial E}{\partial w} = \begin{cases} e(w,k) & ,if \mid e(w,k) \mid \leqq \beta, \\ \beta & ,if \mid e(w,k) \mid > \beta \quad and \quad e(w,k) > 1 \\ -\beta & ,if \mid e \mid > \beta \quad and \quad e(w,k) < 1 \end{cases} \tag{C.4}$$

Appendix D

Choice of n_a, n_b and n_k

The choice of the number pass outputs n_a, the number pass inputs n_b and the number of input samples that occur before the input affects the output n_k (also called the dead time in the system) are an important step in the system identification procedure. In this work, these orders are estimated by trial and error approach, depending on the evaluation of the criterion function each time a combination of n_a, n_b and n_k is used.

The idea is first to chose a random value for n_k, then with the fixed value of n_k (i.e. $n_k = 10$), test different combinations of n_a and n_b, for example $12 \leq n_a \leq< 14$ and $8 \leq n_b \leq 10$, in order to generate 24 different models (see Table D.1).

Table D.1: Performance measures.

Model	n_a	n_b	n_k
1	12	8	10
2	12	9	10
3	12	10	10
4	12	11	10
5	13	8	10
6	13	9	10
7	13	10	10
8	14	11	10
9	14	8	10
10	14	9	10
11	14	10	10
12	14	11	10
13	15	8	10
14	15	9	10
15	15	10	10
16	15	11	10

Finally all the models are estimated and evaluated according to a chosen criterion, (for this example the RMSE).

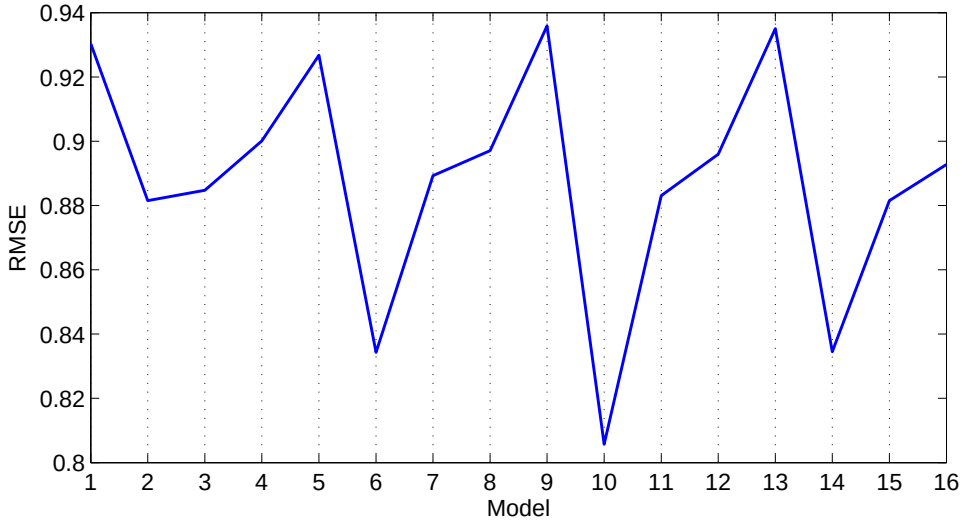


Figure D.1: Choice of n_a and n_b with $n_k = 10$.

From Fig. D.1 the model 10 corresponding to $n_a = 14$ and $n_b = 9$ (see Table D.1) is the one which present the smaller error $RMSE$. Remember that these values of n_a and n_b were computed with a random value of n_k. In a second computation, the values of $n_a = 14$ and $n_b = 9$ are fixed and n_k is variable between a range of values, i.e $1 \leq n_k \leq 10$.

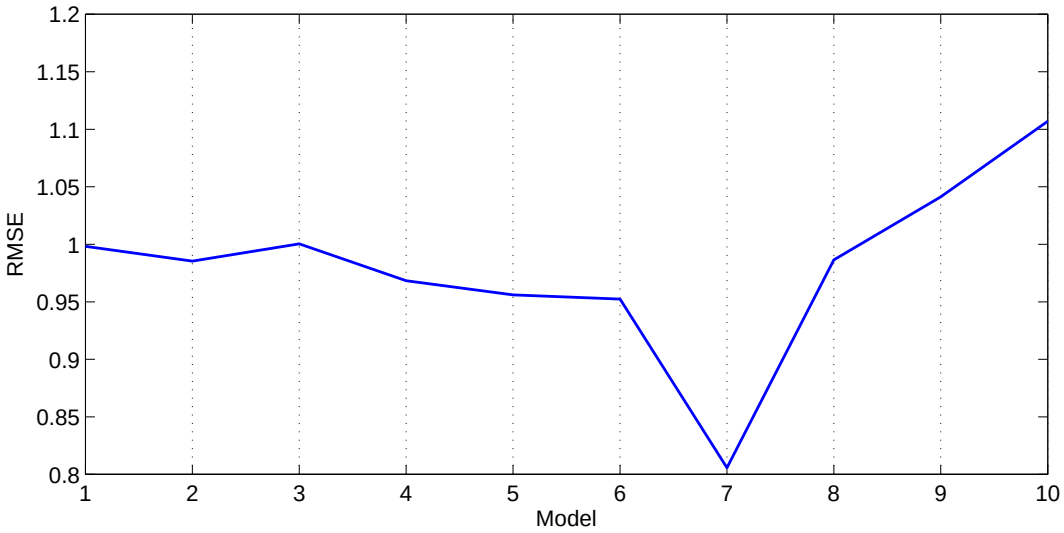


Figure D.2: Choice of n_k with $n_a = 14$ and $n_b = 9$.

From Fig. D.1 the model 7 corresponding to $n_k = 7$ is the one which present the smaller error $RMSE$.

Then, by following this approach we found that the "optimal" number of pass outputs n_a, pass inputs n_b and dead time in the system n_k are $n_a = 14$, $n_b = 9$ and $n_k = 7$.

Appendix E

Learning algorithms for the proposed architectures

In this Appendix, the adaptation algorithms for the adaptation of the synaptic weights of the proposed architectures are presented. This algorithms are obtained based on a classical back-propagation algorithm.

E.1 General learning algorithms

First of all, let us present the general rules to adapt the synaptic weights of the proposed architecture, as already mentioned, this rules are based on the discrete time steepest descent algorithm which is the most used in neural network.

$$X(k+1) = X(k) - \eta \frac{\partial E}{\partial X} \tag{E.1}$$

$$Z_h(k+1) = Z_h(k) - \eta \frac{\partial E}{\partial Z_h} \tag{E.2}$$

$$Z_b(k+1) = Z_b(k) - \eta \frac{\partial E}{\partial Z_b} \tag{E.3}$$

$$Z_a(k+1) = Z_a(k) - \eta \frac{\partial E}{\partial Z_a} \tag{E.4}$$

$$V_{bh}(k+1) = V_{bh}(k) - \eta \frac{\partial E}{\partial V_{bh}} \tag{E.5}$$

$$V_{ah}(k+1) = V_{ah}(k) - \eta \frac{\partial E}{\partial V_{ah}} \tag{E.6}$$

$$V_{b_i}(k+1) = V_{b_i}(k) - \eta \frac{\partial E}{\partial V_{b_i}} \tag{E.7}$$

$$V_{a_i}(k+1) = V_{a_i}(k) - \eta \frac{\partial E}{\partial V_{a_i}} \tag{E.8}$$

$$W_{bh_i}(k+1) = W_{bh_i}(k) - \eta \frac{\partial E}{\partial W_{bh_i}} \tag{E.9}$$

$$W_{ah_i}(k+1) = W_{ah_i}(k) - \eta \frac{\partial E}{\partial W_{ah_i}} \tag{E.10}$$

$$W_{b_i}(k+1) = W_{b_i}(k) - \eta \frac{\partial E}{\partial W_{b_i}} \tag{E.11}$$

$$W_{a_i}(k+1) = W_{a_i}(k) - \eta \frac{\partial E}{\partial W_{a_i}} \tag{E.12}$$

where $i = 1, 2, \cdots, nn$.

In these general adaption rules, the partial derivatives define the adaptation algorithms of each architecture. These partial derivatives are computed by the chain rule as follow:

$$\frac{\partial E}{\partial X} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial X} \tag{E.13}$$

$$\frac{\partial E}{\partial Z_h} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial Z_h} \tag{E.14}$$

$$\frac{\partial E}{\partial Z_b} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial Z_b} \tag{E.15}$$

$$\frac{\partial E}{\partial Z_a} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial Z_a} \tag{E.16}$$

$$\frac{\partial E}{\partial V_{bh}} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial r_b}\frac{\partial r_b}{\partial V_{bh}} \tag{E.17}$$

$$\frac{\partial E}{\partial V_{ah}} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial r_a}\frac{\partial r_a}{\partial V_{ah}} \tag{E.18}$$

$$\frac{\partial E}{\partial V_{b_i}} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial r_b}\frac{\partial r_b}{\partial V_{b_i}} \tag{E.19}$$

$$\frac{\partial E}{\partial V_{a_i}} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial r_a}\frac{\partial r_a}{\partial V_{a_i}} \tag{E.20}$$

$$\frac{\partial E}{\partial W_{bh_i}} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial r_b}\frac{\partial r_b}{\partial W_{bh_i}} \tag{E.21}$$

$$\frac{\partial E}{\partial W_{ah_i}} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial r_a}\frac{\partial r_a}{\partial W_{ah_i}} \tag{E.22}$$

$$\frac{\partial E}{\partial W_{b_i}} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial r_b}\frac{\partial r_b}{\partial W_{b_i}} \tag{E.23}$$

$$\frac{\partial E}{\partial W_{a_i}} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial r_a}\frac{\partial r_a}{\partial W_{a_i}} \tag{E.24}$$

As already mentioned, the learning rule for the adaptation of each synaptic weights of each architecture, depends on the particular structure of a proposed neuro-model. Let us now present the learning rules for the adaptation of the synaptic weights of the neural architectures presented in this thesis.

E.2 FFH1 adaptation algorithms

$$X(k+1) = X(k) + \eta e(k) T \tag{E.25}$$

$$Z_h(k+1) = Z_a(k) + \eta e(k) X \tag{E.26}$$

$$Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \tag{E.27}$$

$$Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \tag{E.28}$$

$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X Z_b \tanh(J_u W_{b_i}) \tag{E.29}$$

$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X Z_a \tanh(J_{\hat{y}} W_{a_i}) \tag{E.30}$$

$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X Z_b V_{b_i} sech^2(J_u W_{b_i}) J_u \tag{E.31}$$

$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X Z_a V_{a_i} sech^2(J_{\hat{y}} W_{a_i}) J_{\hat{y}} \tag{E.32}$$

E.3 FFH2 adaptation algorithms

$$X(k+1) = X(k) + \eta e(k) T \tag{E.33}$$
$$Z_h(k+1) = Z_h(k) + \eta e(k) X \tag{E.34}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X tanh(r_b) \tag{E.35}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X tanh(r_a) \tag{E.36}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X Z_b sech^2(r_b) J_u W_{b_i} \tag{E.37}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X Z_a sech^2(r_a) J_{\hat{y}} W_{a_i} \tag{E.38}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X Z_b sech^2(r_b) V_{b_i} J_u \tag{E.39}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X Z_a sech^2(r_a) V_{a_i} J_{\hat{y}} \tag{E.40}$$

E.4 ARXH adaptation algorithms

$$X(k+1) = X(k) + \eta e(k) T \tag{E.41}$$
$$Z_h(k+1) = Z_h(k) + \eta e(k) X \tag{E.42}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \tag{E.43}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \tag{E.44}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X Z_b J_u W_{b_i} \tag{E.45}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X Z_a J_{\hat{y}} W_{a_i} \tag{E.46}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X Z_b V_{b_i} J_u \tag{E.47}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X Z_a V_{a_i} J_{\hat{y}} \tag{E.48}$$

E.5 NLARXH adaptation algorithms

$$X(k+1) = X(k) + \eta e(k) \tanh(T) \tag{E.49}$$
$$Z_h(k+1) = Z_h(k) + \eta e(k) X sech^2(T) \tag{E.50}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X sech^2(T) r_b \tag{E.51}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X sech^2(T) r_a \tag{E.52}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X sech^2(T) Z_b J_u W_{b_i} \tag{E.53}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X sech^2(T) Z_a J_{\hat{y}} W_{a_i} \tag{E.54}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X sech^2(T) Z_b V_{b_i} J_u \tag{E.55}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X sech^2(T) Z_a V_{a_i} J_{\hat{y}} \tag{E.56}$$

E.6 FF1 adaptation algorithms

$$X(k+1) = X(k) + \eta e(k) T \tag{E.57}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \tag{E.58}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \tag{E.59}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X Z_b \tanh(J_u W_{b_i}) \tag{E.60}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X Z_a \tanh(J_{\hat{y}} W_{a_i}) \tag{E.61}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X Z_b V_{b_i} sech^2(J_u W_{b_i}) J_u \tag{E.62}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X Z_a V_{a_i} sech^2(J_{\hat{y}} W_{a_i}) J_{\hat{y}} \tag{E.63}$$

E.7 FF2 adaptation algorithms

$$X(k+1) = X(k) + \eta e(k) T \tag{E.64}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X tanh(r_b) \tag{E.65}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X tanh(r_a) \tag{E.66}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X Z_b sech^2(r_b) J_u W_{b_i} \tag{E.67}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X Z_a sech^2(r_a) J_{\hat{y}} W_{a_i} \tag{E.68}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X Z_b sech^2(r_b) V_{b_i} J_u \tag{E.69}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X Z_a sech^2(r_a) V_{a_i} J_{\hat{y}} \tag{E.70}$$

E.8 ARX adaptation algorithms

$$X(k+1) = X(k) + \eta e(k) T \tag{E.71}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \tag{E.72}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \tag{E.73}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X Z_b J_u W_{b_i} \tag{E.74}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X Z_a J_{\hat{y}} W_{a_i} \tag{E.75}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X Z_b V_{b_i} J_u \tag{E.76}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X Z_a V_{a_i} J_{\hat{y}} \tag{E.77}$$

E.9 NLARX adaptation algorithms

$$X(k+1) = X(k) + \eta e(k) \tanh(T) \tag{E.78}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X sech^2(T) r_b \tag{E.79}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X sech^2(T) r_a \tag{E.80}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X sech^2(T) Z_b J_u W_{b_i} \tag{E.81}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X sech^2(T) Z_a J_{\hat{y}} W_{a_i} \tag{E.82}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X sech^2(T) Z_b V_{b_i} J_u \tag{E.83}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X sech^2(T) Z_a V_{a_i} J_{\hat{y}} \tag{E.84}$$

E.10 FFHH1 adaptation algorithms

$$X(k+1) = X(k) + \eta e(k) T \tag{E.85}$$
$$Z_h(k+1) = Z_h(k) + \eta e(k) X \tag{E.86}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \tag{E.87}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \tag{E.88}$$
$$V_{bh}(k+1) = V_{bh}(k) + \eta e(k) X Z_b \tag{E.89}$$
$$V_{ah}(k+1) = V_{ah}(k) + \eta e(k) X Z_a \tag{E.90}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X Z_b \tanh(J_u W_{b_i} + W_{bh_i}) \tag{E.91}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X Z_a \tanh(J_{\hat{y}} W_{a_i} + W_{ah_i}) \tag{E.92}$$
$$W_{bh_i}(k+1) = W_{bh_i}(k) + \eta e(k) X Z_b V_{b_i} sech^2(J_u W_{b_i} + W_{bh_i}) \tag{E.93}$$
$$W_{ah_i}(k+1) = W_{ah_i}(k) + \eta e(k) X Z_a V_{a_i} sech^2(J_{\hat{y}} W_{a_i} + W_{ah_i}) \tag{E.94}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X Z_b V_{b_i} sech^2(J_u W_{b_i} + W_{bh_i}) J_u \tag{E.95}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X Z_a V_{a_i} sech^2(J_{\hat{y}} W_{a_i} + W_{ah_i}) J_{\hat{y}} \tag{E.96}$$

E.11 FFHH2 adaptation algorithms

$$X(k+1) = X(k) + \eta e(k) T \tag{E.97}$$
$$Z_h(k+1) = Z_h(k) + \eta e(k) X \tag{E.98}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X \tanh(r_b) \tag{E.99}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X \tanh(r_a) \tag{E.100}$$
$$V_{bh}(k+1) = V_{bh}(k) + \eta e(k) X Z_b sech^2(r_b) \tag{E.101}$$
$$V_{ah}(k+1) = V_{ah}(k) + \eta e(k) X Z_a sech^2(r_a) \tag{E.102}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X Z_b sech^2(r_b) J_u W_{b_i} \tag{E.103}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X Z_a sech^2(r_a) J_{\hat{y}} W_{a_i} \tag{E.104}$$
$$W_{bh_i}(k+1) = W_{bh_i}(k) + \eta e(k) X Z_b sech^2(r_b) V_{b_i} \tag{E.105}$$
$$W_{ah_i}(k+1) = W_{ah_i}(k) + \eta e(k) X Z_a sech^2(r_a) V_{a_i} \tag{E.106}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X Z_b sech^2(r_b) V_{b_i} J_u \tag{E.107}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X Z_a sech^2(r_a) V_{a_i} J_{\hat{y}} \tag{E.108}$$

E.12 ARXHH adaptation algorithms

$$X(k+1) = X(k) + \eta e(k) T \tag{E.109}$$
$$Z_h(k+1) = Z_h(k) + \eta e(k) X \tag{E.110}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \tag{E.111}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \tag{E.112}$$
$$V_{bh}(k+1) = V_{bh}(k) + \eta e(k) X Z_b \tag{E.113}$$
$$V_{ah}(k+1) = V_{ah}(k) + \eta e(k) X Z_a \tag{E.114}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X Z_b J_u W_{b_i} \tag{E.115}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X Z_a J_{\hat{y}} W_{a_i} \tag{E.116}$$
$$W_{bh_i}(k+1) = W_{bh_i}(k) + \eta e(k) X Z_b V_{b_i} \tag{E.117}$$
$$W_{ah_i}(k+1) = W_{ah_i}(k) + \eta e(k) X Z_a V_{a_i} \tag{E.118}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X Z_b V_{b_i} J_u \tag{E.119}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X Z_a V_{a_i} J_{\hat{y}} \tag{E.120}$$

E.13 NLARXHH adaptation algorithms

$$X(k+1) = X(k) + \eta e(k) \tanh(T) \tag{E.121}$$
$$Z_h(k+1) = Z_h(k) + \eta e(k) X sech^2(T) \tag{E.122}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X sech^2(T) r_b \tag{E.123}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X sech^2(T) r_a \tag{E.124}$$
$$V_{bh}(k+1) = V_{bh}(k) + \eta e(k) X sech^2(T) Z_b \tag{E.125}$$
$$V_{ah}(k+1) = V_{ah}(k) + \eta e(k) X sech^2(T) Z_a \tag{E.126}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X sech^2(T) Z_b J_u W_{b_i} \tag{E.127}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X sech^2(T) Z_a J_{\hat{y}} W_{a_i} \tag{E.128}$$
$$W_{bh_i}(k+1) = W_{bh_i}(k) + \eta e(k) X sech^2(T) Z_b V_{b_i} \tag{E.129}$$
$$W_{ah_i}(k+1) = W_{ah_i}(k) + \eta e(k) X sech^2(T) Z_a V_{a_i} \tag{E.130}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X sech^2(T) Z_b V_{b_i} J_u \tag{E.131}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X sech^2(T) Z_a V_{a_i} J_{\hat{y}} \tag{E.132}$$

As already mentioned, the LM algorithm was implemented for estimation of the parameters of the FFHH1 neuro-model.

E.14 FFHH1 adaptation algorithms based on Levenberg-Marquardt

$$X(k+1) = X(k) + [\lambda]^{-1} e(k) T \tag{E.133}$$
$$Z_h(k+1) = Z_h(k) + [\lambda]^{-1} e(k) X \tag{E.134}$$
$$Z_b(k+1) = Z_b(k) + [\lambda]^{-1} e(k) X r_b \tag{E.135}$$
$$Z_a(k+1) = Z_a(k) + [\lambda]^{-1} e(k) X r_a \tag{E.136}$$
$$V_{bh}(k+1) = V_{bh}(k) + [\lambda]^{-1} e(k) X Z_b \tag{E.137}$$
$$V_{ah}(k+1) = V_{ah}(k) + [\lambda]^{-1} e(k) X Z_a \tag{E.138}$$
$$V_{b_i}(k+1) = V_{b_i}(k) + [\lambda]^{-1} e(k) X Z_b \tanh(J_u W_{b_i} + W_{bh_i}) \tag{E.139}$$
$$V_{a_i}(k+1) = V_{a_i}(k) + [\lambda]^{-1} e(k) X Z_a \tanh(J_{\hat{y}} W_{a_i} + W_{ah_i}) \tag{E.140}$$
$$W_{bh_i}(k+1) = W_{bh_i}(k) + [H_{W_{bh}} + \lambda]^{-1} e(k) X Z_b V_{b_i} sech^2(J_u W_{b_i} + W_{bh_i}) \tag{E.141}$$
$$W_{ah_i}(k+1) = W_{ah_i}(k) + [H_{W_{ah}} + \lambda]^{-1} e(k) X Z_a V_{a_i} sech^2(J_{\hat{y}} W_{a_i} + W_{ah_i}) \tag{E.142}$$
$$W_{b_i}(k+1) = W_{b_i}(k) + [H_{W_b} + \lambda]^{-1} e(k) X Z_b V_{b_i} sech^2(J_u W_{b_i} + W_{bh_i}) J_u \tag{E.143}$$
$$W_{a_i}(k+1) = W_{a_i}(k) + [H_{W_a} + \lambda]^{-1} e(k) X Z_a V_{a_i} sech^2(J_{\hat{y}} W_{a_i} + W_{ah_i}) J_{\hat{y}} \tag{E.144}$$

where:

$$H_{W_b} = e(k) X Z_b V_b sech^2(J_u W_b + W_{bh}) \left(2 \tanh(J_u W_b + W_{bh}) J_u^T J_u - 1\right) \tag{E.145}$$
$$H_{W_a} = e(k) X Z_a V_a sech^2(J_{\hat{y}} W_a + W_{ah}) \left(2 \tanh(J_{\hat{y}} W_a + W_{ah}) J_{\hat{y}}^T J_{\hat{y}} - 1\right) \tag{E.146}$$
$$H_{W_{bh}} = e(k) X Z_b V_b 2 sech^2(J_u W_b + W_{bh}) \tanh(J_u W_b + W_{bh}) \tag{E.147}$$
$$H_{W_{ah}} = e(k) X Z_a V_a 2 sech^2(J_{\hat{y}} W_a + W_{ah}) \tanh(J_{\hat{y}} W_a + W_{ah}) \tag{E.148}$$

Appendix F

Learning rate adaptation η

The learning rate or integration step size, plays an important role in the parameters adaptation in particular the convergence speed of the algorithm. It should be pointed out that for the discrete time steepest descent algorithm, η should be bounded in a small range, to nevertheless ensure the stability of the algorithm. A small η means that the convergence to a solution is slow while a large η means a faster convergence where oscillations may occur and stability may be lost.

In order to improve the learning algorithm, different solutions for the adaptation of η have been implemented in this work.

F.1 Search then convergence algorithm

Let us start with a modification of the "search then convergence algorithm" used in [23] is given by (F.1).

$$\eta(n) = \eta_0 \frac{1}{1 + \frac{n}{k_0 \gamma}} \tag{F.1}$$

where:

η_0 is the initial value of η proposed by the user.
$k_0 = \frac{10N}{3}$
$n = 1, 2, \cdots, (N \times \gamma)$ is the iteration increasing during all the training phase (all the epochs).
$\gamma = 1, 2, \cdots, \gamma*$ is a the number of epochs elapsed.
N is the number of data used for the neural network training.
γ^* is a "optimal" number of epochs required for the adaptation of the model parameters.

Every time that one epoch has elapsed $\eta_0 = \eta(\gamma)$ and $\gamma = \gamma + 1$.

From (1.6) we can deduce that when n increases the value of the learning coefficient $\eta(k)$ decreases (see Fig. 1.8). It is desirable because at the beginning of the training the synaptic weights are supposed far of their optimal values and they should be adapted as fast as possible ($\eta = \eta_0$). When a period of time is over ($k \to (N \times \gamma)$) the synaptic weights are supposed near their optimal values. Then, the parameters should be adapted slowly ($\eta \to 0$) in order to improve the convergence.

F.2 Heuristic rules

A different algorithm based on evaluation of the objective function each iteration is presented in [18]. Here, the adaptation of the learning rate is based on the following heuristic rules:

- If the objective function undergoes 4 consecutive reductions then:

$$\eta = \eta + (0.1 * \eta)$$

- If the objective function undergoes 2 consecutive combinations of one increase and one reduction:

$$\eta = \eta - (0.1 * \eta)$$

Appendix G

Application of the proposed method to the other models

Let us follow the proposed system identification procedure:

G.1 Model FF2

- **Choice of model structure:** The first step in the proposed system identification method is to choose the activation functions according to Assumption 1.

  Model FF2: By selecting $\varphi_3(z) = \varphi_1(z) = z$ and $\varphi_2(z) = nonlinear$ in (4.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) \\ r_b &= \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{G.1}$$

- **Model computation:** Let us follow the improved procedure:

  Step 1.- Neural network training under particular assumptions

  According to Assumption 2, the initial conditions of the synaptic weights are chosen equals group by group as follows $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{b_1}(0) = W_{b_j}(0)$ and $W_{a_1}(0) = W_{a_j}(0)$ with $j = 2, 3, \cdots, nn$.

  Then, the proposed neuro-model is trained following the computational cost reduction approach presented in the proof of Theorem 2, that is:

To train the equivalent model given by (G.2):

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) \\ r_b &= nn \times V_{b_1}(J_u W_{b_1}) \\ r_a &= nn \times V_{a_1}(J_{\hat{y}} W_{a_1}) \end{aligned} \tag{G.2}$$

with the synaptic weights computed as follows:

$$\begin{aligned} X(k+1) &= X(k) + \eta e(k) T \\ Z_b(k+1) &= Z_b(k) + \eta e(k) X tanh(r_b) \\ Z_a(k+1) &= Z_a(k) + \eta e(k) X tanh(r_a) \\ V_{b_1}(k+1) &= V_{b_1}(k) + \eta e(k) X Z_b sech^2(r_b) J_u W_{b_1} \\ V_{a_1}(k+1) &= V_{a_1}(k) + \eta e(k) X Z_a sech^2(r_a) J_{\hat{y}} W_{a_1} \\ W_{b_1}(k+1) &= W_{b_1}(k) + \eta e(k) X Z_b sech^2(r_b) V_{b_1} J_u \\ W_{a_1}(k+1) &= W_{a_1}(k) + \eta e(k) X Z_a sech^2(r_a) V_{a_1} J_{\hat{y}} \end{aligned}$$

Once the neural network model given by (G.1) is trained under these two assumptions, we obtain:

$$\begin{aligned} \hat{y}(k) &= X^* T \\ T &= Z_b^* \varphi_2(r_b) + Z_a^* \varphi_2(r_a) \\ r_b &= \sum_{i=1}^{nn} V_{b_i}^* (J_u W_{b_i}^*) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}^* (J_{\hat{y}} W_{a_i}^*) \end{aligned} \tag{G.3}$$

Since the initial conditions of the synaptic weights are chosen equals group by group (see Assumption 2), and each group is trained by the same adaptation rule, the final values of the synaptic weights are: $V_{b_1}^* = V_{b_j}^*$, $V_{a_1}^* = V_{a_j}^*$, $W_{b_1}^* = W_{b_j}^*$ and $W_{a_1}^* = W_{a_j}^*$ with $j = 2, 3, \cdots, nn$.

Step 2.- Model transformation

According to Theorem 1, the final values of the synaptic weights are: $W_{b_1}^* = W_{b_j}^*$, $V_{b_1}^* = V_{b_j}^*$, $W_{a_1}^* = W_{a_j}^*$ and $V_{a_1}^* = V_{a_j}^*$ with $j = 2, 3, \cdots, nn$. Then, in (G.3) it is indeed possible to make the following algebraic operations:

$$\begin{aligned} Z_B^* &= X^* \times Z_b^* \\ Z_A^* &= X^* \times Z_a^* \\ W_{B_i}^* &= V_{b_i}^* \times W_{b_i}^* \\ W_{A_i}^* &= V_{a_i}^* \times W_{a_i}^* \end{aligned}$$

where $W_{B_1}^* = W_{B_j}^*$ and $W_{A_1}^* = W_{A_j}^*$ (with $j = 2, 3, \cdots, nn$) due to Assumption 2 (Step 1).

Then, the 2nn-2-1 neuro-model given by (G.3) becomes:

$$\begin{aligned} \hat{y}(k) &= T \\ T &= Z_B \varphi_2(r_b) + Z_A \varphi_2(r_a) \\ r_b &= \sum_{i=1}^{nn} (J_u W^*_{B_i}) \\ r_a &= \sum_{i=1}^{nn} (J_{\hat{y}} W^*_{A_i}) \end{aligned} \tag{G.4}$$

A supplementary transformation is achieved in order to change the redefined 2nn-1 model (G.4) into a 2-1 representation by the following algebraic operations:

$$\begin{aligned} \sum_{i=1}^{nn} (J_u W^*_{B_i}) &= nn \times (J_u W^*_B) \\ \sum_{i=1}^{nn} (J_{\hat{y}} W^*_{A_i}) &= nn \times (J_{y_n} W^*_A) \end{aligned}$$

where:

$$\begin{aligned} W^*_B &= W^*_{B_1} \\ W^*_A &= W^*_{A_1} \end{aligned}$$

The resulting model after the model transformation has the following mathematical form:

$$\hat{y}(k) = Z^*_B \varphi_2(J_u W^*_B) + Z^*_A \varphi_2(J_{\hat{y}} W^*_A) \tag{G.5}$$

G.2 Model ARX

- **Choice of model structure:** The first step in the proposed system identification method is to choose the activation functions according to Assumption 1.

  Model ARX: By selecting $\varphi_3(z) = \varphi_2(z) = \varphi_1(z) = z$ in (4.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b(r_b) + Z_a(r_a) \\ r_b &= \sum_{i=1}^{nn} V_{b_i} (J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i} (J_{\hat{y}} W_{a_i}) \end{aligned} \tag{G.6}$$

- **Model computation:** Let us follow the improved procedure:

Step 1.- Neural network training under particular assumptions

According to Assumption 2, the initial conditions of the synaptic weights are chosen equals group by group as follows $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{b_1}(0) = W_{b_j}(0)$ and $W_{a_1}(0) = W_{a_j}(0)$ with $j = 2, 3, \cdots, nn$.

Then, the proposed neuro-model is trained following the computational cost reduction approach presented in the proof of Theorem 2, that is:

To train the equivalent model given by (G.7):

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b(r_b) + Z_a(r_a) \\ r_b &= nn \times V_{b_1}(J_u W_{b_1}) \\ r_a &= nn \times V_{a_1}(J_{\hat{y}} W_{a_1}) \end{aligned} \tag{G.7}$$

with the synaptic weights computed as follows:

$$\begin{aligned} X(k+1) &= X(k) + \eta e(k) T \\ Z_b(k+1) &= Z_b(k) + \eta e(k) X r_b \\ Z_a(k+1) &= Z_a(k) + \eta e(k) X r_a \\ V_{b_i}(k+1) &= V_{b_i}(k) + \eta e(k) X Z_b J_u W_{b_i} \\ V_{a_i}(k+1) &= V_{a_i}(k) + \eta e(k) X Z_a J_{\hat{y}} W_{a_i} \\ W_{b_i}(k+1) &= W_{b_i}(k) + \eta e(k) X Z_b V_{b_i} J_u \\ W_{a_i}(k+1) &= W_{a_i}(k) + \eta e(k) X Z_a V_{a_i} J_{\hat{y}} \end{aligned}$$

Once the neural network model given by (G.6) is trained under these two assumptions, we obtain:

$$\begin{aligned} \hat{y}(k) &= X^* T \\ T &= Z_b^*(r_b) + Z_a^*(r_a) \\ r_b &= \sum_{i=1}^{nn} V_{b_i}^*(J_u W_{b_i}^*) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}^*(J_{\hat{y}} W_{a_i}^*) \end{aligned} \tag{G.8}$$

Since the initial conditions of the synaptic weights are chosen equals group by group (see Assumption 2), and each group is trained by the same adaptation rule, the final values of the synaptic weights are: $V_{b_1}^* = V_{b_j}^*$, $V_{a_1}^* = V_{a_j}^*$, $W_{b_1}^* = W_{b_j}^*$ and $W_{a_1}^* = W_{a_j}^*$ with $j = 2, 3, \cdots, nn$.

Step 2.- Model transformation

According to Theorem 1, the final values of the synaptic weights are: $W^*_{b_1} = W^*_{b_j}$, $V^*_{b_1} = V^*_{b_j}$, $W^*_{a_1} = W^*_{a_j}$ and $V^*_{a_1} = V^*_{a_j}$ with $j = 2, 3, \cdots, nn$. Then, in (G.8) it is indeed possible to make the following algebraic operations:

$$W^*_{B_i} = X^* \times Z^*_b \times V^*_{b_i} \times W^*_{b_i}$$
$$W^*_{A_i} = X^* \times Z^*_a \times V^*_{a_i} \times W^*_{a_i}$$

where $W^*_{B_1} = W^*_{B_j}$ and $W^*_{A_1} = W^*_{A_j}$ (with $j = 2, 3, \cdots, nn$) due to Assumption 2 (Step 1).

Then, the 2nn-2-1 neuro-model given by (G.8) becomes:

$$\begin{aligned} \hat{y}(k) &= T \\ T &= r_b + r_a \\ r_b &= \sum_{i=1}^{nn} (J_u W^*_{B_i}) \\ r_a &= \sum_{i=1}^{nn} (J_{\hat{y}} W^*_{A_i}) \end{aligned} \tag{G.9}$$

A supplementary transformation is achieved in order to change the redefined 2nn-1 model (G.9) into a 2-1 representation by the following algebraic operations:

$$\sum_{i=1}^{nn} (J_u W^*_{B_i}) = nn \times (J_u W^*_B)$$
$$\sum_{i=1}^{nn} (J_{\hat{y}} W^*_{A_i}) = nn \times (J_{\hat{y}} W^*_A)$$

where:

$$W^*_B = W^*_{B_1}$$
$$W^*_A = W^*_{A_1}$$

The resulting model after the model transformation has the following mathematical form:

$$\hat{y}(k) = (J_u W^*_B) + (J_{\hat{y}} W^*_A) \tag{G.10}$$

G.3 Model NLARX

- **Choice of model structure:** The first step in the proposed system identification method is to choose the activation functions according to Assumption 1.

  Model NLARX: By selecting $\varphi_2(z) = \varphi_1(z) = z$ and $\varphi_3(z) = \textit{nonlinear}$ in (4.1) we obtain:

$$\begin{aligned}
\hat{y}(k) &= X\varphi_3(T) \\
T &= Z_b(r_b) + Z_a(r_a) \\
r_b &= \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i}) \\
r_a &= \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i})
\end{aligned} \tag{G.11}$$

- **Model computation:** Let us follow the improved procedure:

Step 1.- Neural network training under particular assumptions

According to Assumption 2, the initial conditions of the synaptic weights are chosen equals group by group as follows $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{b_1}(0) = W_{b_j}(0)$ and $W_{a_1}(0) = W_{a_j}(0)$ with $j = 2, 3, \cdots, nn$.

Then, the proposed neuro-model is trained following the computational cost reduction approach presented in the proof of Theorem 2, that is:

To train the equivalent model given by (G.12):

$$\begin{aligned}
\hat{y}(k) &= X\varphi_3(T) \\
T &= Z_b(r_b) + Z_a(r_a) \\
r_b &= nn \times V_{b_1}(J_u W_{b_1}) \\
r_a &= nn \times V_{a_1}(J_{\hat{y}} W_{a_1})
\end{aligned} \tag{G.12}$$

with the synaptic weights computed as follows:

$$\begin{aligned}
X(k+1) &= X(k) + \eta e(k) \tanh(T) \\
Z_b(k+1) &= Z_b(k) + \eta e(k) X sech^2(T) r_b \\
Z_a(k+1) &= Z_a(k) + \eta e(k) X sech^2(T) r_a \\
V_{b_i}(k+1) &= V_{b_i}(k) + \eta e(k) X sech^2(T) Z_b J_u W_{b_i} \\
V_{a_i}(k+1) &= V_{a_i}(k) + \eta e(k) X sech^2(T) Z_a J_{\hat{y}} W_{a_i} \\
W_{b_i}(k+1) &= W_{b_i}(k) + \eta e(k) X sech^2(T) Z_b V_{b_i} J_u \\
W_{a_i}(k+1) &= W_{a_i}(k) + \eta e(k) X sech^2(T) Z_a V_{a_i} J_{\hat{y}}
\end{aligned}$$

Once the neural network model given by (G.11) is trained under these two assumptions, we obtain:

$$\begin{aligned}
\hat{y}(k) &= X^*\varphi_3(T) \\
T &= Z_b^*(r_b) + Z_a^*(r_a) \\
r_b &= \sum_{i=1}^{nn} V_{b_i}^*(J_u W_{b_i}^*) \\
r_a &= \sum_{i=1}^{nn} V_{a_i}^*(J_{\hat{y}} W_{a_i}^*)
\end{aligned} \tag{G.13}$$

Since the initial conditions of the synaptic weights are chosen equals group by group (see Assumption 2), and each group is trained by the same adaptation rule, the final values of the synaptic weights are: $V_{b_1}^* = V_{b_j}^*$, $V_{a_1}^* = V_{a_j}^*$, $W_{b_1}^* = W_{b_j}^*$ and $W_{a_1}^* = W_{a_j}^*$ with $j = 2, 3, \cdots, nn$.

Step 2.- Model transformation

According to Theorem 1, the final values of the synaptic weights are: $W_{b_1}^* = W_{b_j}^*$, $V_{b_1}^* = V_{b_j}^*$, $W_{a_1}^* = W_{a_j}^*$ and $V_{a_1}^* = V_{a_j}^*$ with $j = 2, 3, \cdots, nn$. Then, in (G.13) it is indeed possible to make the following algebraic operations:

$$W_{B_i}^* = Z_b^* \times V_{b_i}^* \times W_{b_i}^*$$
$$W_{A_i}^* = Z_a^* \times V_{a_i}^* \times W_{a_i}^*$$

where $W_{B_1}^* = W_{B_j}^*$ and $W_{A_1}^* = W_{A_j}^*$ (with $j = 2, 3, \cdots, nn$) due to Assumption 2 (Step 1).

Then, the 2nn-2-1 neuro-model given by (G.13) becomes:

$$\begin{aligned} \hat{y}(k) &= X^* \varphi_3(T) \\ T &= r_b + r_a \\ r_b &= \sum_{i=1}^{nn} (J_u W_{B_i}^*) \\ r_a &= \sum_{i=1}^{nn} (J_{\hat{y}} W_{A_i}^*) \end{aligned} \tag{G.14}$$

A supplementary transformation is achieved in order to change the redefined 2nn-1 model (G.14) into a 2-1 representation by the following algebraic operations:

$$\sum_{i=1}^{nn} (J_u W_{B_i}^*) = nn \times (J_u W_B^*)$$
$$\sum_{i=1}^{nn} (J_{\hat{y}} W_{A_i}^*) = nn \times (J_{y_n} W_A^*)$$

where:

$$W_B^* = W_{B_1}^*$$
$$W_A^* = W_{A_1}^*$$

The resulting model after the model transformation has the following mathematical form:

$$\hat{y}(k) = X^* \varphi_3((J_u W_B^*) + (J_{\hat{y}} W_A^*)) \tag{G.15}$$

G.4 Model FF2H

- **Choice of model structure:** The first step in the proposed system identification method is to choose the activation functions according to Assumption 1.

  Model FF2H: By selecting $\varphi_3(z) = \varphi_1(z) = z$ and $\varphi_2(z) = nonlinear$ in (2.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) + Z_h \\ r_b &= \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{G.16}$$

- **Model computation:** Let us follow the improved procedure:

  Step 1.- Neural network training under particular assumptions

  According to Assumption 2, the initial conditions of the synaptic weights are chosen equals group by group as follows $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{b_1}(0) = W_{b_j}(0)$ and $W_{a_1}(0) = W_{a_j}(0)$ with $j = 2, 3, \cdots, nn$.

  Then, the proposed neuro-model is trained following the computational cost reduction approach presented in the proof of Theorem 2, that is:

  To train the equivalent model given by (G.17):

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) + Z_h \\ r_b &= nn \times V_{b_1}(J_u W_{b_1}) \\ r_a &= nn \times V_{a_1}(J_{\hat{y}} W_{a_1}) \end{aligned} \tag{G.17}$$

  with the synaptic weights computed as follows:

$$\begin{aligned} X(k+1) &= X(k) + \eta e(k) T \\ Z_h(k+1) &= Z_a(k) + \eta e(k) X \\ Z_b(k+1) &= Z_b(k) + \eta e(k) X tanh(r_b) \\ Z_a(k+1) &= Z_a(k) + \eta e(k) X tanh(r_a) \\ V_{b_1}(k+1) &= V_{b_1}(k) + \eta e(k) X Z_b sech^2(r_b) J_u W_{b_1} \\ V_{a_1}(k+1) &= V_{a_1}(k) + \eta e(k) X Z_a sech^2(r_a) J_{\hat{y}} W_{a_1} \\ W_{b_1}(k+1) &= W_{b_1}(k) + \eta e(k) X Z_b sech^2(r_b) V_{b_1} J_u \\ W_{a_1}(k+1) &= W_{a_1}(k) + \eta e(k) X Z_a sech^2(r_a) V_{a_1} J_{\hat{y}} \end{aligned}$$

Once the neural network model given by (G.16) is trained under these two assumptions, we obtain:

$$\begin{aligned}
&\hat{y}(k) = X^* T \\
&T = Z_b^* \varphi_2(r_b) + Z_a^* \varphi_2(r_a) + Z_h^* \\
&r_b = \sum_{i=1}^{nn} V_{b_i}^* (J_u W_{b_i}^*) \\
&r_a = \sum_{i=1}^{nn} V_{a_i}^* (J_{\hat{y}} W_{a_i}^*)
\end{aligned} \tag{G.18}$$

Since the initial conditions of the synaptic weights are chosen equals group by group (see Assumption 2), and each group is trained by the same adaptation rule, the final values of the synaptic weights are: $V_{b_1}^* = V_{b_j}^*$, $V_{a_1}^* = V_{a_j}^*$, $W_{b_1}^* = W_{b_j}^*$ and $W_{a_1}^* = W_{a_j}^*$ with $j = 2, 3, \cdots, nn$.

Step 2.- Model transformation

According to Theorem 1, the final values of the synaptic weights are: $W_{b_1}^* = W_{b_j}^*$, $V_{b_1}^* = V_{b_j}^*$, $W_{a_1}^* = W_{a_j}^*$ and $V_{a_1}^* = V_{a_j}^*$ with $j = 2, 3, \cdots, nn$. Then, in (G.18) it is indeed possible to make the following algebraic operations:

$$\begin{aligned}
H^* &= X^* \times Z_h^* \\
Z_B^* &= X^* \times Z_b^* \\
Z_A^* &= X^* \times Z_a^* \\
W_{B_i}^* &= V_{b_i}^* \times W_{b_i}^* \\
W_{A_i}^* &= V_{a_i}^* \times W_{a_i}^*
\end{aligned}$$

where $W_{B_1}^* = W_{B_j}^*$ and $W_{A_1}^* = W_{A_j}^*$ (with $j = 2, 3, \cdots, nn$) due to Assumption 2 (Step 1).

Then, the 2nn-2-1 neuro-model given by (G.18) becomes:

$$\begin{aligned}
&\hat{y}(k) = T \\
&T = Z_B^* \varphi_2(r_b) + Z_A^* \varphi_2(r_a) + H^* \\
&r_b = \sum_{i=1}^{nn} (J_u W_{B_i}^*) \\
&r_a = \sum_{i=1}^{nn} (J_{\hat{y}} W_{A_i}^*)
\end{aligned} \tag{G.19}$$

A supplementary transformation is achieved in order to change the redefined 2nn-1 model (G.19) into a 2-1 representation by the following algebraic operations:

$$\sum_{i=1}^{nn}(J_u W_{B_i}^*) = nn \times (J_u W_B^*)$$
$$\sum_{i=1}^{nn}(J_{\hat{y}} W_{A_i}^*) = nn \times (J_{y_n} W_A^*)$$

where:

$$W_B^* = W_{B_1}^*$$
$$W_A^* = W_{A_1}^*$$

The resulting model after the model transformation has the following mathematical form:

$$\hat{y}(k) = Z_B^* \varphi_2(J_u W_B^*) + Z_A^* \varphi_2(J_{\hat{y}} W_A^*) + H^* \tag{G.20}$$

G.5 Model ARXH

- **Choice of model structure:** The first step in the proposed system identification method is to choose the activation functions according to Assumption 1.

  Model ARXH: By selecting $\varphi_3(z) = \varphi_2(z) = \varphi_1(z) = z$ in (2.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b(r_b) + Z_a(r_a) + Z_h \\ r_b &= \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{G.21}$$

- **Model computation:** Let us follow the improved procedure:

  Step 1.- Neural network training under particular assumptions

  According to Assumption 2, the initial conditions of the synaptic weights are chosen equals group by group as follows $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{b_1}(0) = W_{b_j}(0)$ and $W_{a_1}(0) = W_{a_j}(0)$ with $j = 2, 3, \cdots, nn$.

  Then, the proposed neuro-model is trained following the computational cost reduction approach presented in the proof of Theorem 2, that is:

To train the equivalent model given by (G.22):

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b(r_b) + Z_a(r_a) + Z_h \\ r_b &= nn \times V_{b_1}(J_u W_{b_1}) \\ r_a &= nn \times V_{a_1}(J_{\hat{y}} W_{a_1}) \end{aligned} \tag{G.22}$$

with the synaptic weights computed as follows:

$$\begin{aligned} X(k+1) &= X(k) + \eta e(k) T \\ Z_h(k+1) &= Z_a(k) + \eta e(k) X \\ Z_b(k+1) &= Z_b(k) + \eta e(k) X r_b \\ Z_a(k+1) &= Z_a(k) + \eta e(k) X r_a \\ V_{b_i}(k+1) &= V_{b_i}(k) + \eta e(k) X Z_b J_u W_{b_i} \\ V_{a_i}(k+1) &= V_{a_i}(k) + \eta e(k) X Z_a J_{\hat{y}} W_{a_i} \\ W_{b_i}(k+1) &= W_{b_i}(k) + \eta e(k) X Z_b V_{b_i} J_u \\ W_{a_i}(k+1) &= W_{a_i}(k) + \eta e(k) X Z_a V_{a_i} J_{\hat{y}} \end{aligned}$$

Once the neural network model given by (G.21) is trained under these two assumptions, we obtain:

$$\begin{aligned} \hat{y}(k) &= X^* T \\ T &= Z_b^*(r_b) + Z_a^*(r_a) + Z_h^* \\ r_b &= \sum_{i=1}^{nn} V_{b_i}^*(J_u W_{b_i}^*) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}^*(J_{\hat{y}} W_{a_i}^*) \end{aligned} \tag{G.23}$$

Since the initial conditions of the synaptic weights are chosen equals group by group (see Assumption 2), and each group is trained by the same adaptation rule, the final values of the synaptic weights are: $V_{b_1}^* = V_{b_j}^*$, $V_{a_1}^* = V_{a_j}^*$, $W_{b_1}^* = W_{b_j}^*$ and $W_{a_1}^* = W_{a_j}^*$ with $j = 2, 3, \cdots, nn$.

Step 2.- Model transformation

According to Theorem 1, the final values of the synaptic weights are: $W_{b_1}^* = W_{b_j}^*$, $V_{b_1}^* = V_{b_j}^*$, $W_{a_1}^* = W_{a_j}^*$ and $V_{a_1}^* = V_{a_j}^*$ with $j = 2, 3, \cdots, nn$. Then, in (G.23) it is indeed possible to make the following algebraic operations:

$$\begin{aligned} H^* &= X^* \times Z_h^* \\ W_{B_i}^* &= X^* \times Z_b^* \times V_{b_i}^* \times W_{b_i}^* \\ W_{A_i}^* &= X^* \times Z_a^* \times V_{a_i}^* \times W_{a_i}^* \end{aligned}$$

where $W_{B_1}^* = W_{B_j}^*$ and $W_{A_1}^* = W_{A_j}^*$ (with $j = 2, 3, \cdots, nn$) due to Assumption 2 (Step 1).

Then, the 2nn-2-1 neuro-model given by (G.23) becomes:

$$\begin{aligned} \hat{y}(k) &= T \\ T &= r_b + r_a + H^* \\ r_b &= \sum_{i=1}^{nn} (J_u W^*_{B_i}) \\ r_a &= \sum_{i=1}^{nn} (J_{\hat{y}} W^*_{A_i}) \end{aligned} \tag{G.24}$$

A supplementary transformation is achieved in order to change the redefined 2nn-1 model (G.24) into a 2-1 representation by the following algebraic operations:

$$\sum_{i=1}^{nn} (J_u W^*_{B_i}) = nn \times (J_u W^*_B)$$
$$\sum_{i=1}^{nn} (J_{\hat{y}} W^*_{A_i}) = nn \times (J_{y_n} W^*_A)$$

where:

$$W^*_B = W^*_{B1}$$
$$W^*_A = W^*_{A1}$$

The resulting model after the model transformation has the following mathematical form:

$$\hat{y}(k) = (J_u W^*_B) + (J_{\hat{y}} W^*_A) + H^* \tag{G.25}$$

G.6 Model NLARXH

- **Choice of model structure:** The first step in the proposed system identification method is to choose the activation functions according to Assumption 1.

  Model NLARX: By selecting $\varphi_2(z) = \varphi_1(z) = z$ and $\varphi_3(z) = nonlinear$ in (2.1) we obtain:

$$\begin{aligned} \hat{y}(k) &= X\varphi_3(T) \\ T &= Z^*_b(r_b) + Z_a(r_a) + Z_h \\ r_b &= \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{G.26}$$

- **Model computation:** Let us follow the improved procedure:

Step 1.- Neural network training under particular assumptions

According to Assumption 2, the initial conditions of the synaptic weights are chosen equals group by group as follows $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{b_1}(0) = W_{b_j}(0)$ and $W_{a_1}(0) = W_{a_j}(0)$ with $j = 2, 3, \cdots, nn$.

Then, the proposed neuro-model is trained following the computational cost reduction approach presented in the proof of Theorem 2, that is:

To train the equivalent model given by (G.27):

$$\begin{aligned} \hat{y}(k) &= X\varphi_3(T) \\ T &= Z_b(r_b) + Z_a(r_a) + Z_h \\ r_b &= nn \times V_{b_1}(J_u W_{b_1}) \\ r_a &= nn \times V_{a_1}(J_{\hat{y}} W_{a_1}) \end{aligned} \tag{G.27}$$

with the synaptic weights computed as follows:

$$\begin{aligned} X(k+1) &= X(k) + \eta e(k) \tanh(T) \\ Z_h(k+1) &= Z_h(k) + \eta e(k) X sech^2(T) \\ Z_b(k+1) &= Z_b(k) + \eta e(k) X sech^2(T) r_b \\ Z_a(k+1) &= Z_a(k) + \eta e(k) X sech^2(T) r_a \\ V_{b_i}(k+1) &= V_{b_i}(k) + \eta e(k) X sech^2(T) Z_b J_u W_{b_i} \\ V_{a_i}(k+1) &= V_{a_i}(k) + \eta e(k) X sech^2(T) Z_a J_{\hat{y}} W_{a_i} \\ W_{b_i}(k+1) &= W_{b_i}(k) + \eta e(k) X sech^2(T) Z_b V_{b_i} J_u \\ W_{a_i}(k+1) &= W_{a_i}(k) + \eta e(k) X sech^2(T) Z_a V_{a_i} J_{\hat{y}} \end{aligned}$$

Once the neural network model given by (G.26) is trained under these two assumptions, we obtain:

$$\begin{aligned} \hat{y}(k) &= X^* \varphi_3(T) \\ T &= Z_b^*(r_b) + Z_a^*(r_a) + Z_h^* \\ r_b &= \sum_{i=1}^{nn} V_{b_i}^*(J_u W_{b_i}^*) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}^*(J_{\hat{y}} W_{a_i}^*) \end{aligned} \tag{G.28}$$

Since the initial conditions of the synaptic weights are chosen equals group by group (see Assumption 2), and each group is trained by the same adaptation rule, the final values of the synaptic weights are: $V_{b_1}^* = V_{b_j}^*$, $V_{a_1}^* = V_{a_j}^*$, $W_{b_1}^* = W_{b_j}^*$ and $W_{a_1}^* = W_{a_j}^*$ with $j = 2, 3, \cdots, nn$.

Step 2.- Model transformation

According to Theorem 1, the final values of the synaptic weights are: $W^*_{b_1} = W^*_{b_j}$, $V^*_{b_1} = V^*_{b_j}$, $W^*_{a_1} = W^*_{a_j}$ and $V^*_{a_1} = V^*_{a_j}$ with $j = 2, 3, \cdots, nn$. Then, in (G.28) it is indeed possible to make the following algebraic operations:

$$\begin{aligned} H^* &= Z^*_h \\ W^*_{B_i} &= Z^*_b \times V^*_{b_i} \times W^*_{b_i} \\ W^*_{A_i} &= Z^*_a \times V^*_{a_i} \times W^*_{a_i} \end{aligned}$$

where $W^*_{B_1} = W^*_{B_j}$ and $W^*_{A_1} = W^*_{A_j}$ (with $j = 2, 3, \cdots, nn$) due to Assumption 2 (Step 1).

Then, the 2nn-2-1 neuro-model given by (G.28) becomes:

$$\begin{aligned} \hat{y}(k) &= X^* \varphi_3(T) \\ T &= r_b + r_a + H^* \\ r_b &= \sum_{i=1}^{nn} (J_u W^*_{B_i}) \\ r_a &= \sum_{i=1}^{nn} (J_{\hat{y}} W^*_{A_i}) \end{aligned} \tag{G.29}$$

A supplementary transformation is achieved in order to change the redefined 2nn-1 model (G.29) into a 2-1 representation by the following algebraic operations:

$$\begin{aligned} \sum_{i=1}^{nn} (J_u W^*_{B_i}) &= nn \times (J_u W^*_B) \\ \sum_{i=1}^{nn} (J_{\hat{y}} W^*_{A_i}) &= nn \times (J_{y_n} W^*_A) \end{aligned}$$

where:

$$\begin{aligned} W^*_B &= W^*_{B_1} \\ W^*_A &= W^*_{A_1} \end{aligned}$$

The resulting model after the model transformation has the following mathematical form:

$$\hat{y}(k) = X^* \varphi_3((J_u W^*_B) + (J_{\hat{y}} W^*_A) + H^*) \tag{G.30}$$

G.7 Model FF2HH

- **Choice of model structure:** The first step in the proposed system identification method is to choose the activation functions according to Assumption 1.

  Model FF2HH: By selecting $\varphi_3(z) = \varphi_1(z) = z$ and $\varphi_2(z) = nonlinear$ in (4.14) we obtain:

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) + Z_h \\ r_b &= V_{bh} + \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i} + W_{bh_i}) \\ r_a &= V_{ah} + \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i} + W_{ah_i}) \end{aligned} \tag{G.31}$$

- **Model computation:** Let us follow the improved procedure:

  Step 1.- Neural network training under particular assumptions

  According to Assumption 2, the initial conditions of the synaptic weights are chosen equals group by group, for this architecture that is: $W_{b_1}(0) = W_{b_j}(0)$, $W_{a_1}(0) = W_{a_j}(0)$, $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{bh_1}(0) = W_{bh_j}(0)$ and $W_{ah_1}(0) = W_{ah_j}(0)$ with $j = 2, 3, \cdots, nn$.

  Then, the proposed neuro-model is trained following the computational cost reduction approach presented in the proof of Theorem 2, that is:

  To train the equivalent model given by (G.32):

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) + Z_h \\ r_b &= V_{bh} + nn \times V_{b_i}(J_u W_{b_i} + W_{bh_i}) \\ r_a &= V_{ah} + nn \times V_{a_i}(J_{\hat{y}} W_{a_i} + W_{ah_i}) \end{aligned} \tag{G.32}$$

  with the synaptic weights computed as follows:

$$\begin{aligned} X(k+1) &= X(k) + \eta e(k) T \\ Z_h(k+1) &= Z_h(k) + \eta e(k) X \\ Z_b(k+1) &= Z_b(k) + \eta e(k) X \tanh(r_b) \\ Z_a(k+1) &= Z_a(k) + \eta e(k) X \tanh(r_a) \end{aligned}$$

$$
\begin{aligned}
V_{bh}(k+1) &= V_{bh}(k) + \eta e(k) X Z_b sech^2(r_b) \\
V_{ah}(k+1) &= V_{ah}(k) + \eta e(k) X Z_a sech^2(r_a) \\
V_{b_i}(k+1) &= V_{b_i}(k) + \eta e(k) X Z_b sech^2(r_b) J_u W_{b_i} \\
V_{a_i}(k+1) &= V_{a_i}(k) + \eta e(k) X Z_a sech^2(r_a) J_{\hat{y}} W_{a_i} \\
W_{bh_i}(k+1) &= W_{bh_i}(k) + \eta e(k) X Z_b sech^2(r_b) V_{b_i} \\
W_{ah_i}(k+1) &= W_{ah_i}(k) + \eta e(k) X Z_a sech^2(r_a) V_{a_i} \\
W_{b_i}(k+1) &= W_{b_i}(k) + \eta e(k) X Z_b sech^2(r_b) V_{b_i} J_u \\
W_{a_i}(k+1) &= W_{a_i}(k) + \eta e(k) X Z_a sech^2(r_a) V_{a_i} J_{\hat{y}}
\end{aligned}
$$

Once the neural network model given by (G.31) is trained under these two assumptions, we obtain:

$$
\begin{aligned}
\hat{y}(k) &= X^* T \\
T &= Z_b^* \varphi_2(r_b) + Z_a^* \varphi_2(r_a) + Z_h^* \\
r_b &= V_{bh}^* + \sum_{i=1}^{nn} V_{b_i}^* (J_u W_{b_i}^* + W_{bh_i}^*) \\
r_a &= V_{ah}^* + \sum_{i=1}^{nn} V_{a_i}^* (J_{\hat{y}} W_{a_i}^* + W_{ah_i}^*)
\end{aligned}
\tag{G.33}
$$

Since the initial conditions of the synaptic weights are chosen equals group by group (see Assumption 2), and each group is trained by the same adaptation rule, the final values of the synaptic weights are: $W_{b_1}^* = W_{b_j}^*$, $V_{b_1}^* = V_{b_j}^*$, $W_{bh_1}^* = W_{bh_j}^*$, $W_{a_1}^* = W_{a_j}^*$, $V_{a_1}^* = V_{a_j}^*$ and $W_{ah_1}^* = W_{ah_j}^*$ with $j = 2, 3, \cdots, nn$.

Step 2.- Model transformation

According to Theorem 1, the final values of the synaptic weights are: $W_{b_1}^* = W_{b_j}^*$, $V_{b_1}^* = V_{b_j}^*$, $W_{bh_1}^* = W_{bh_j}^*$, $W_{a_1}^* = W_{a_j}^*$, $V_{a_1}^* = V_{a_j}^*$ and $W_{ah_1}^* = W_{ah_j}^*$ with $j = 2, 3, \cdots, nn$. Then, in (G.33) it is indeed possible to make the following algebraic operations:

$$
\begin{aligned}
H^* &= X^* \times Z_h^* \\
Z_B^* &= X^* \times Z_b^* \\
Z_A^* &= X^* \times Z_a^* \\
W_{B_i}^* &= V_{b_i}^* \times W_{b_i}^* \\
W_{A_i}^* &= V_{a_i}^* \times W_{a_i}^* \\
W_{BH_i}^* &= V_{b_i}^* \times W_{bh_i}^* \\
W_{AH_i}^* &= V_{a_i}^* \times W_{ah_i}^*
\end{aligned}
$$

where $W_{B_1}^* = W_{B_j}^*$, $W_{A_1}^* = W_{A_j}^*$, $W_{BH_1}^* = W_{BH_j}^*$ and $W_{AH_1}^* = W_{AH_j}^*$ (with $j = 2, 3, \cdots, nn$) due to Assumption 2 (Step 1).

Then, the 2nn-2-1 neuro-model given by (G.33) becomes:

$$\begin{aligned}
&\hat{y}(k) = T \\
&T = Z_B^* \varphi_2(r_b) + Z_A^* \varphi_2(r_a) + H^* \\
&r_b = V_{bh}^* + \sum_{i=1}^{nn} (J_u W_{B_i}^* + W_{BH_i}^*) \\
&r_a = V_{ah}^* + \sum_{i=1}^{nn} (J_{\hat{y}} W_{A_i}^* + W_{AH_i}^*)
\end{aligned} \tag{G.34}$$

A supplementary transformation is achieved in order to change the redefined 2nn-1 model (G.34) into a 2-1 representation by the following algebraic operations:

$$V_{bh}^* + \sum_{i=1}^{nn} (J_u W_{B_i}^* + W_{BH_i}^*) = V_{bh}^* + (nn \times (J_u W_B^* + W_{BH_1}^*))$$
$$V_{ah}^* + \sum_{i=1}^{nn} (J_{\hat{y}} W_{A_i}^* + W_{AH_i}^*) = V_{ah}^* + (nn \times (J_{\hat{y}} W_A^* + W_{AH_1}^*))$$

where:

$$W_B^* = W_{B_1}^*$$
$$W_A^* = W_{A_1}^*$$

The resulting model after the model transformation has the following mathematical form:

$$\hat{y}(k) = Z_B^* \varphi_2(J_u W_B^* + W_{BH}^*) + Z_A^* \varphi_2(J_{\hat{y}} W_A^* + W_{AH}^*) + H^* \tag{G.35}$$

with:

$$W_{BH}^* = V_{bh} + W_{BH_1}^*$$
$$W_{AH}^* = V_{ah} + W_{AH_1}^*$$

G.8 Model ARXHH

- **Choice of model structure:** The first step in the proposed system identification method is to choose the activation functions according to Assumption 1.

  Model ARXH: By selecting $\varphi_3(z) = \varphi_2(z) = \varphi_1(z) = z$ in (4.14) we obtain:

$$\begin{aligned}
&\hat{y}(k) = XT \\
&T = Z_b r_b + Z_a r_a + Z_h \\
&r_b = V_{bh} + \sum_{i=1}^{nn} V_{b_i} (J_u W_{b_i} + W_{bh_i}) \\
&r_a = V_{ah} + \sum_{i=1}^{nn} V_{a_i} (J_{\hat{y}} W_{a_i} + W_{ah_i})
\end{aligned} \tag{G.36}$$

- **Model computation:** Let us follow the improved procedure:

Step 1.- Neural network training under particular assumptions

According to Assumption 2, the initial conditions of the synaptic weights are chosen equals group by group as follows $W_{b_1}(0) = W_{b_j}(0)$, $W_{a_1}(0) = W_{a_j}(0)$, $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{bh_1}(0) = W_{bh_j}(0)$ and $W_{ah_1}(0) = W_{ah_j}(0)$ with $j = 2, 3, \cdots, nn$.

Then, the proposed neuro-model is trained following the computational cost reduction approach presented in the proof of Theorem 2, that is:

To train the equivalent model given by (G.37):

$$\begin{aligned} &\hat{y}(k) = XT \\ &T = Z_b r_b + Z_a r_a + Z_h \\ &r_b = V_{bh} + nn \times V_{b_i}(J_u W_{b_i} + W_{bh_i}) \\ &r_a = V_{ah} + nn \times V_{a_i}(J_{\hat{y}} W_{a_i} + W_{ah_i}) \end{aligned} \tag{G.37}$$

with the synaptic weights computed as follows:

$$\begin{aligned} &X(k+1) = X(k) + \eta e(k) T \\ &Z_h(k+1) = Z_h(k) + \eta e(k) X \\ &Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \\ &Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \\ &V_{bh}(k+1) = V_{bh}(k) + \eta e(k) X Z_b \\ &V_{ah}(k+1) = V_{ah}(k) + \eta e(k) X Z_a \\ &V_{b_1}(k+1) = V_{b_1}(k) + \eta e(k) X Z_b J_u W_{b_1} \\ &V_{a_1}(k+1) = V_{a_1}(k) + \eta e(k) X Z_a J_{\hat{y}} W_{a_1} \\ &W_{bh_1}(k+1) = W_{bh_1}(k) + \eta e(k) X Z_b V_{b_1} \\ &W_{ah_1}(k+1) = W_{ah_1}(k) + \eta e(k) X Z_a V_{a_1} \\ &W_{b_1}(k+1) = W_{b_1}(k) + \eta e(k) X Z_b V_{b_1} J_u \\ &W_{a_1}(k+1) = W_{a_1}(k) + \eta e(k) X Z_a V_{a_1} J_{\hat{y}} \end{aligned}$$

Once the neural network model given by (G.36) is trained under these two assumptions, we obtain:

$$\begin{aligned} &\hat{y}(k) = X^* T \\ &T = Z_b^* r_b + Z_a^* r_a + Z_h^* \\ &r_b = V_{bh}^* + \sum_{i=1}^{nn} V_{b_i}^* (J_u W_{b_i}^* + W_{bh_i}^*) \\ &r_a = V_{ah}^* + \sum_{i=1}^{nn} V_{a_i}^* (J_{\hat{y}} W_{a_i}^* + W_{ah_i}^*) \end{aligned} \tag{G.38}$$

Since the initial conditions of the synaptic weights are chosen equals group by group (see Assumption 2), and each group is trained by the same adaptation rule, the final values of the synaptic weights are: $W^*_{b_1} = W^*_{b_j}$, $V^*_{b_1} = V^*_{b_j}$, $W^*_{bh_1} = W^*_{bh_j}$, $W^*_{a_1} = W^*_{a_j}$, $V^*_{a_1} = V^*_{a_j}$ and $W^*_{ah_1} = W^*_{ah_j}$ with $j = 2, 3, \cdots, nn$.

Step 2.- Model transformation

According to Theorem 1, the final values of the synaptic weights are: $W^*_{b_1} = W^*_{b_j}$, $V^*_{b_1} = V^*_{b_j}$, $W^*_{bh_1} = W^*_{bh_j}$, $W^*_{a_1} = W^*_{a_j}$, $V^*_{a_1} = V^*_{a_j}$ and $W^*_{ah_1} = W^*_{ah_j}$ with $j = 2, 3, \cdots, nn$. Then, in (G.38) it is indeed possible to make the following algebraic operations:

$$
\begin{aligned}
Z^*_H &= X^* \times Z^*_h \\
V^*_{BH} &= X^* \times Z^*_b \times V^*_{bh} \\
V^*_{AH} &= X^* \times Z^*_a \times V^*_{ah} \\
W^*_{BH_i} &= X^* \times Z^*_b \times V^*_{b_i} \times W^*_{bh_i} \\
W^*_{AH_i} &= X^* \times Z^*_a \times V^*_{a_i} \times W^*_{ah_i} \\
W^*_{B_i} &= X^* \times Z^*_b \times V^*_{b_i} \times W^*_{b_i} \\
W^*_{A_i} &= X^* \times Z^*_a \times V^*_{a_i} \times W^*_{a_i}
\end{aligned}
$$

where $W^*_{B_1} = W^*_{B_j}$, $W^*_{A_1} = W^*_{A_j}$, $W^*_{BH_1} = W^*_{BH_j}$ and $W^*_{AH_1} = W^*_{AH_j}$ (with $j = 2, 3, \cdots, nn$) due to Assumption 2 (Step 1).

Then, the 2nn-2-1 neuro-model given by (G.38) becomes:

$$
\begin{aligned}
\hat{y}(k) &= T \\
T &= r_b + r_a + Z^*_H \\
r_b &= V^*_{BH} + \sum_{i=1}^{nn} (J_u W^*_{B_i} + W^*_{BH_i}) \\
r_a &= V^*_{AH} + \sum_{i=1}^{nn} (J_{\hat{y}} W^*_{A_i} + W^*_{AH_i})
\end{aligned}
\tag{G.39}
$$

A supplementary transformation is achieved in order to change the redefined 2nn-1 model (G.39) into a 2-1 representation by the following algebraic operations:

$$
\begin{aligned}
V^*_{BH} + \sum_{i=1}^{nn} (J_u W^*_{B_i} + W^*_{BH_i}) &= V^*_{BH} + (nn \times (J_u W^*_B + W^*_{BH_1})) \\
V^*_{AH} + \sum_{i=1}^{nn} (J_{\hat{y}} W^*_{A_i} + W^*_{AH_i}) &= V^*_{AH} + (nn \times (J_{y_n} W^*_A + W^*_{BH_1}))
\end{aligned}
$$

where:

$$
\begin{aligned}
W^*_B &= W^*_{B_1} \\
W^*_A &= W^*_{A_1}
\end{aligned}
$$

The resulting model after the model transformation has the following mathematical form:

$$\hat{y}(k) = (J_u W_B^*) + (J_{\hat{y}} W_A^*) + H^* \tag{G.40}$$

with:

$$H^* = Z_H^* + V_{BH}^* + V_{AH}^* + (nn \times (W_{BH_1} + W_{AH_1})).$$

G.9 Model NLARXHH

- **Choice of model structure:** The first step in the proposed system identification method is to choose the activation functions according to Assumption 1.

  Model NLARX: By selecting $\varphi_2(z) = \varphi_1(z) = z$ and $\varphi_3(z) = nonlinear$ in (4.14) we obtain:

$$\begin{aligned} \hat{y}(k) &= X\varphi_3(T) \\ T &= Z_b^*(r_b) + Z_a(r_a) + Z_h \\ r_b &= V_{bh} + \sum_{i=1}^{nn} V_{b_i}(J_u W_{b_i} + W_{bh_i}) \\ r_a &= V_{ah} + \sum_{i=1}^{nn} V_{a_i}(J_{\hat{y}} W_{a_i} + W_{ah_i}) \end{aligned} \tag{G.41}$$

- **Model computation:** Let us follow the improved procedure:

  Step 1.- Neural network training under particular assumptions

  According to Assumption 2, the initial conditions of the synaptic weights are chosen equals group by group as follows $W_{b_1}(0) = W_{b_j}(0)$, $W_{a_1}(0) = W_{a_j}(0)$, $V_{b_1}(0) = V_{b_j}(0)$, $V_{a_1}(0) = V_{a_j}(0)$, $W_{bh_1}(0) = W_{bh_j}(0)$ and $W_{ah_1}(0) = W_{ah_j}(0)$ with $j = 2, 3, \cdots, nn$.

  Then, the proposed neuro-model is trained following the computational cost reduction approach presented in the proof of Theorem 2, that is:

  To train the equivalent model given by (G.42):

$$\begin{aligned} \hat{y}(k) &= X\varphi_3(T) \\ T &= Z_b(r_b) + Z_a(r_a) + Z_h \\ r_b &= V_{bh} + nn \times V_{b_i}(J_u W_{b_i} + W_{bh_i}) \\ r_a &= V_{ah} + nn \times V_{a_i}(J_{\hat{y}} W_{a_i} + W_{ah_i}) \end{aligned} \tag{G.42}$$

  with the synaptic weights computed as follows:

$$
\begin{aligned}
X(k+1) &= X(k) + \eta e(k) \tanh(T) \\
Z_h(k+1) &= Z_h(k) + \eta e(k) X sech^2(T) \\
Z_b(k+1) &= Z_b(k) + \eta e(k) X sech^2(T) r_b \\
Z_a(k+1) &= Z_a(k) + \eta e(k) X sech^2(T) r_a \\
V_{bh}(k+1) &= V_{bh}(k) + \eta e(k) X sech^2(T) Z_b \\
V_{ah}(k+1) &= V_{ah}(k) + \eta e(k) X sech^2(T) Z_a \\
V_{b_i}(k+1) &= V_{b_i}(k) + \eta e(k) X sech^2(T) Z_b J_u W_{b_i} \\
V_{a_i}(k+1) &= V_{a_i}(k) + \eta e(k) X sech^2(T) Z_a J_{\hat{y}} W_{a_i} \\
W_{bh_i}(k+1) &= W_{bh_i}(k) + \eta e(k) X sech^2(T) Z_b V_{b_i} \\
W_{ah_i}(k+1) &= W_{ah_i}(k) + \eta e(k) X sech^2(T) Z_a V_{a_i} \\
W_{b_i}(k+1) &= W_{b_i}(k) + \eta e(k) X sech^2(T) Z_b V_{b_i} J_u \\
W_{a_i}(k+1) &= W_{a_i}(k) + \eta e(k) X sech^2(T) Z_a V_{a_i} J_{\hat{y}}
\end{aligned}
$$

Once the neural network model given by (G.41) is trained under these two assumptions, we obtain:

$$
\begin{aligned}
\hat{y}(k) &= X^* \varphi_3(T) \\
T &= Z_b^*(r_b) + Z_a^*(r_a) + Z_h^* \\
r_b &= V_{bh}^* + \sum_{i=1}^{nn} V_{b_i}^* (J_u W_{b_i}^* + W_{bh_i}^*) \\
r_a &= V_{ah}^* + \sum_{i=1}^{nn} V_{a_i}^* (J_{\hat{y}} W_{a_i}^* + W_{ah_i}^*)
\end{aligned}
\tag{G.43}
$$

Since the initial conditions of the synaptic weights are chosen equals group by group (see Assumption 2), and each group is trained by the same adaptation rule, the final values of the synaptic weights are: $W_{b_1}^* = W_{b_j}^*$, $V_{b_1}^* = V_{b_j}^*$, $W_{bh_1}^* = W_{bh_j}^*$, $W_{a_1}^* = W_{a_j}^*$, $V_{a_1}^* = V_{a_j}^*$ and $W_{ah_1}^* = W_{ah_j}^*$ with $j = 2, 3, \cdots, nn$.

Step 2.- Model transformation

According to Theorem 1, the final values of the synaptic weights are: $W_{b_1}^* = W_{b_j}^*$, $V_{b_1}^* = V_{b_j}^*$, $W_{bh_1}^* = W_{bh_j}^*$, $W_{a_1}^* = W_{a_j}^*$, $V_{a_1}^* = V_{a_j}^*$ and $W_{ah_1}^* = W_{ah_j}^*$ with $j = 2, 3, \cdots, nn$. Then, in (G.43) it is indeed possible to make the following algebraic operations:

$$
\begin{aligned}
V_{BH}^* &= Z_b^* \times V_{bh}^* \\
V_{AH}^* &= Z_a^* \times V_{ah}^* \\
W_{BH_i}^* &= Z_b^* \times V_{b_i}^* \times W_{bh_i}^* \\
W_{AH_i}^* &= Z_a^* \times V_{a_i}^* \times W_{ah_i}^* \\
W_{B_i}^* &= Z_b^* \times V_{b_i}^* \times W_{b_i}^* \\
W_{A_i}^* &= Z_a^* \times V_{a_i}^* \times W_{a_i}^*
\end{aligned}
$$

where $W_{B_1}^* = W_{B_j}^*$, $W_{A_1}^* = W_{A_j}^*$, $W_{BH_1}^* = W_{BH_j}^*$ and $W_{AH_1}^* = W_{AH_j}^*$ (with $j = 2, 3, \cdots, nn$) due to Assumption 2 (Step 1).

Then, the 2nn-2-1 neuro-model given by (G.43) becomes:

$$\begin{aligned}
&\hat{y}(k) = X^{*}\varphi_3(T) \\
&T = r_b + r_a + Z_h^{*} \\
&r_b = V_{BH}^{*} + \sum_{i=1}^{nn} (J_u W_{B_i}^{*} + W_{BH_i}^{*}) \\
&r_a = V_{AH}^{*} + \sum_{i=1}^{nn} (J_{\hat{y}} W_{A_i}^{*} + W_{AH_i}^{*})
\end{aligned} \tag{G.44}$$

A supplementary transformation is achieved in order to change the redefined 2nn-1 model (G.44) into a 2-1 representation by the following algebraic operations:

$$V_{BH}^{*} + \sum_{i=1}^{nn} (J_u W_{B_i}^{*} + W_{BH_i}^{*}) = V_{BH}^{*} + (nn \times (J_u W_B^{*} + W_{BH_1}^{*}))$$
$$V_{AH}^{*} + \sum_{i=1}^{nn} (J_{\hat{y}} W_{A_i}^{*} + W_{AH_i}^{*}) = V_{AH}^{*} + (nn \times (J_{\hat{y}} W_A^{*} + W_{AH_1}^{*}))$$

where:

$$\begin{aligned}
W_B^{*} &= W_{B_1}^{*} \\
W_A^{*} &= W_{A_1}^{*}
\end{aligned}$$

The resulting model after the model transformation has the following mathematical form:

$$\hat{y}(k) = X^{*}\varphi_3((J_u W_B^{*}) + (J_{\hat{y}} W_A^{*}) + H^{*}) \tag{G.45}$$

with:

$$H^{*} = Z_h^{*} + V_{BH} + W_{BH_1}^{*} + V_{AH} + W_{AH_1}^{*}$$

Appendix H

A different way to see the contributions

In Chapter 2, three different neural networks were presented. A 2nn-2-1 architecture (Fig. 2.1), a 2-2-1 architecture (Fig. 2.3) and a 2-1 neural network (Fig. 2.2). Following Theorem 1 and Theorem 2, an efficient system identification method is proposed in Chapter 4. The models derived by this method have the accuracy of the 2nn-2-1 architecture, the complexity of the 2-1 neural network and the computational bulk of the 2-2-1 architecture.

According to the system identification method, the idea is the following:

First, we propose a neural network architecture, more precisely we propose a 2nn-2-1 neural network with the activation functions according to Assumption 1 and the initial conditions of the synaptic weights according to Assumption 2.

Let us take for this example the FF1 model given by (H.1) presented in Chapter 4.

$$\begin{aligned} \hat{y}(k) &= XT \\ T &= Z_b r_b + Z_a r_a \\ r_b &= \sum_{i=1}^{nn} V_{b_i} \varphi_1(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{nn} V_{a_i} \varphi_1(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{H.1}$$

This model satisfies Assumption 1. Now, following the method proposed in Chapter 4 we establish the initial conditions according to Assumption 2. Classically, the synaptic weights are computed as follows.

$$X(k+1) = X(k) + \eta e(k) T \tag{H.2}$$

$$Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \tag{H.3}$$

$$Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \tag{H.4}$$

$$V_{b_i}(k+1) = V_{b_i}(k) + \eta e(k) X Z_b \tanh(J_u W_{b_i}) \tag{H.5}$$

$$V_{a_i}(k+1) = V_{a_i}(k) + \eta e(k) X Z_a \tanh(J_{\hat{y}} W_{a_i}) \tag{H.6}$$

$$W_{b_i}(k+1) = W_{b_i}(k) + \eta e(k) X Z_b V_{b_i} sech^2(J_u W_{b_i}) J_u \tag{H.7}$$

$$W_{a_i}(k+1) = W_{a_i}(k) + \eta e(k) X Z_a V_{a_i} sech^2(J_{\hat{y}} W_{a_i}) J_{\hat{y}} \tag{H.8}$$

with $i = 1, \cdots, nn$.

Once the neural network is trained, we obtain the following model:

$$\begin{aligned}\hat{y}(k) &= X^*T \\ T &= Z_b^* r_b + Z_a^* r_a \\ r_b &= \sum_{i=1}^{nn} V_{b_i}^* \varphi_1(J_u W_{b_i}^*) \\ r_a &= \sum_{i=1}^{nn} V_{a_i}^* \varphi_1(J_{\hat{y}} W_{a_i}^*)\end{aligned} \tag{H.9}$$

where the final values of the synaptic weights after the training are: $V_{b_1}^* = V_{b_j}^*$, $V_{a_1}^* = V_{a_j}^*$, $W_{b_1}^* = W_{b_j}^*$ and $W_{a_1}^* = W_{a_j}^*$ with $j = 2, 3, \cdots, nn$.

From this result an idea arises, that is, to adapt V_{b_i}, V_{a_i}, W_{b_i} and W_{a_i} only one time. According to the above the synaptic weights are adapted as follows:

$$X(k+1) = X(k) + \eta e(k) T \tag{H.10}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \tag{H.11}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \tag{H.12}$$
$$V_{b_1}(k+1) = V_{b_1}(k) + \eta e(k) X Z_b \tanh(J_u W_{b_1}) \tag{H.13}$$
$$V_{a_1}(k+1) = V_{a_1}(k) + \eta e(k) X Z_a \tanh(J_{\hat{y}} W_{a_1}) \tag{H.14}$$
$$W_{b_1}(k+1) = W_{b_1}(k) + \eta e(k) X Z_b V_{b_1} sech^2(J_u W_{b_1}) J_u \tag{H.15}$$
$$W_{a_1}(k+1) = W_{a_1}(k) + \eta e(k) X Z_a V_{a_1} sech^2(J_{\hat{y}} W_{a_1}) J_{\hat{y}} \tag{H.16}$$

and the model (H.1) should be redefined:

$$\begin{aligned}\hat{y}(k) &= XT \\ T &= Z_b r_b + Z_a r_a \\ r_b &= \sum_{i=1}^{nn} V_{b_1} \varphi_1(J_u W_{b_1}) \\ r_a &= \sum_{i=1}^{nn} V_{a_1} \varphi_1(J_{\hat{y}} W_{a_1})\end{aligned} \tag{H.17}$$

Notice that, since the arguments of the sums are the same, i.e.

$$\begin{aligned}r_b &= \sum_{i=1}^{nn} V_{b_1} \varphi_1(J_u W_{b_1}) \\ r_a &= \sum_{i=1}^{nn} V_{a_1} \varphi_1(J_{\hat{y}} W_{a_1})\end{aligned}$$

this model (H.17) can be redefined again as:

$$\begin{aligned}
&\hat{y}(k) = XT \\
&T = Z_b r_b + Z_a r_a \\
&r_b = nn \times V_{b_1} \varphi_1(J_u W_{b_1}) \\
&r_a = nn \times V_{a_1} \varphi_1(J_{\hat{y}} W_{a_1})
\end{aligned} \tag{H.18}$$

Once this neural network is trained we have:

$$\begin{aligned}
&\hat{y}(k) = X^* T \\
&T = Z_b^* r_b + Z_a^* r_a \\
&r_b = nn \times V_{b_1}^* \varphi_1(J_u W_{b_1}^*) \\
&r_a = nn \times V_{a_1}^* \varphi_1(J_{\hat{y}} W_{a_1}^*)
\end{aligned} \tag{H.19}$$

Following Theorem 1 this model becomes:

$$\hat{y}(k) = V_B^* \varphi_1(J_u W_B^*) + V_A^* \varphi_1(J_{\hat{y}} W_A^*) \tag{H.20}$$

From these model transformations different questions may arise.

Why do not to train directly the model (H.20)?

Complete experiments developed in Chapter 5 shown that if we train directly the 2-1 model (H.20) we do not reach the same accuracy as when the 2nn-2-1 neuro-model (H.1) is trained. This is somehow natural, is well known in neural network theory that more neurons lead a better accuracy.

On the contrary, if we start training the 2-2-1 neural network depicted in Fig. 2.3, and we improve the adaptation algorithms by the introduction of the parameter nn, then it is possible to reach the same accuracy as the one achieved by training first the complex 2nn-2-1 architecture.

Notice that, this parameter nn comes from the complex $2nn - 2 - 1$ neural network architecture.

In order to demonstrate the above mentioned, let us take the 2-2-1 architecture corresponding to the FF1 model:

$$\begin{aligned}
&\hat{y}(k) = XT \\
&T = Z_b r_b + Z_a r_a \\
&r_b = V_{b_1} \varphi_1(J_u W_{b_1}) \\
&r_a = V_{a_1} \varphi_1(J_{\hat{y}} W_{a_1})
\end{aligned} \tag{H.21}$$

The learning algorithms for the adaptation of the synaptic weights of such model are:

$$X(k+1) = X(k) + \eta e(k) T \tag{H.22}$$

$$Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \tag{H.23}$$

$$Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \tag{H.24}$$
$$V_{b_1}(k+1) = V_{b_1}(k) + \eta e(k) X Z_b \tanh(J_u W_{b_1}) \tag{H.25}$$
$$V_{a_1}(k+1) = V_{a_1}(k) + \eta e(k) X Z_a \tanh(J_{\hat{y}} W_{a_1}) \tag{H.26}$$
$$W_{b_1}(k+1) = W_{b_1}(k) + \eta e(k) X Z_b V_{b_1} sech^2(J_u W_{b_1}) J_u \tag{H.27}$$
$$W_{a_1}(k+1) = W_{a_1}(k) + \eta e(k) X Z_a V_{a_1} sech^2(J_{\hat{y}} W_{a_1}) J_{\hat{y}} \tag{H.28}$$

Notice that the adaption algorithms of the model (H.21) and (H.18) are the same. Nevertheless, both models are different and therefore the accuracy reached is not the same.

In order to reach the same accuracy we have to introduce the parameter nn (named number of neurons) to the model (H.21). Then the model given by (H.21) becomes (H.18).

Notice that, trained in a classical way the parameters of the model (H.18) should be computed as follows:

$$X(k+1) = X(k) + \eta e(k) T \tag{H.29}$$
$$Z_b(k+1) = Z_b(k) + \eta e(k) X r_b \tag{H.30}$$
$$Z_a(k+1) = Z_a(k) + \eta e(k) X r_a \tag{H.31}$$
$$V_{b_1}(k+1) = V_{b_1}(k) + \eta e(k) nn X Z_b \tanh(J_u W_{b_1}) \tag{H.32}$$
$$V_{a_1}(k+1) = V_{a_1}(k) + \eta e(k) nn X Z_a \tanh(J_{\hat{y}} W_{a_1}) \tag{H.33}$$
$$W_{b_1}(k+1) = W_{b_1}(k) + \eta e(k) nn X Z_b V_{b_1} sech^2(J_u W_{b_1}) J_u \tag{H.34}$$
$$W_{a_1}(k+1) = W_{a_1}(k) + \eta e(k) nn X Z_a V_{a_1} sech^2(J_{\hat{y}} W_{a_1}) J_{\hat{y}} \tag{H.35}$$

We can see that, the algorithms (H.25), (H.26), (H.27) and (H.28) are different (see nn) to the algorithms (H.32), (H.33), (H.34) and (H.35), hence the accuracy of the estimation is different.

From these reflexions, we can conclude that we can reach the same accuracy as the 2nn-2-1 model of Fig. 2.1 by training directly the 2-2-1 neural network of Fig. (2.3) if we add the parameter nn (number of neurons) to the model given by (H.21) and we compute the parameters of such model according to (H.22), (H.23), (H.24), (H.25), (H.26), (H.27) and (H.28). In fact, we add the parameter nn to the model and we develop the learning algorithms as if the parameter does not exist.

Here it is interesting to notice that, we can see the main contributions of this thesis in two different ways:

1) As presented in Chapter 4, that is, a model complexity and computational cost reduction procedures, where we start with a complex 2nn-2-1 neural network architecture (Fig. 2.1), and then, thanks to Assumption 1 and Assumption 2 we can reduce both the complexity and the computational cost.

2) As presented in this appendix, that is, a 2-2-1 architecture (Fig. 2.3) with an optimized learning algorithm (to add the parameter nn to the model and develop the learning algorithms as if the parameter does not exist).

Appendix I

Model reduction: generalization

The main contributions of this thesis were presented in Chapter 3. The proposed model complexity reduction approach allows us to reduce the number of parameters of a 2nn-2-1 architecture into the number of parameters of a 2-1 neural network. A computational cost reduction approach, allows us to reduce the computational bulk required to train a 2nn-2-1 architecture into the one required to train a 2-2-1 neural network.

Notice that, these contributions are presented as theorems, which are defined based on a particular architecture proposed in this thesis. In this section, these theorems are generalized in order to be applicable for a more general type of neural network architecture, i. e. the three layers perceptron presented in [6], depicted in Fig. I.1 with the mathematical representation given by (I.1).

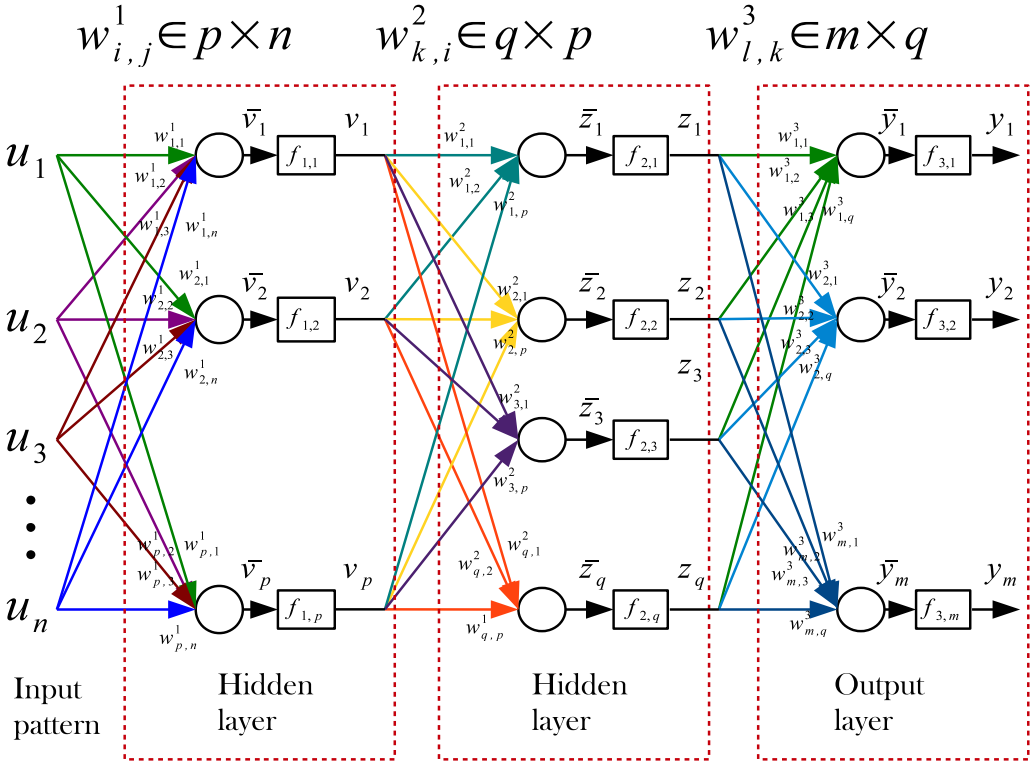


Figure I.1: Multilayer perceptron of Narendra.

$$
\begin{aligned}
y_l &= f_{3,l}\left(\sum_{k=1}^{q}\left(w_{l,k}^{3} z_k\right)\right) \\
z_k &= f_{2,k}\left(\sum_{i=1}^{p}\left(w_{k,i}^{2} v_i\right)\right) \\
v_i &= f_{1,i}\left(\sum_{j=1}^{n}\left(w_{i,j}^{1} u_j\right)\right)
\end{aligned}
\tag{I.1}
$$

with $j = 1, \cdots, n$ and $l = 1, \cdots, m$, where n is the number of inputs, p is the number of neurons in the first hidden layer, q is the number of neurons in the second hidden layer and m is the number of outputs.

I.1 Validity assumptions for the generalized model reduction approach

As same as for Theorem 1 and Theorem 2, two simple assumptions whose purpose is to achieve two design conditions should be satisfied. The first one is a neural architecture design condition and the second one is a training design condition. The reader shall notice that Assumption 4, represents the originality of this work.

Assumption 3: All the activation functions of each layer of the neural network should be chosen equals, that is, $f_{1,1}(T) = f_{1,k}(T)$, with $k = 2, \cdots, p$, $f_{2,1}(T) = f_{2,l}$ with $l = 2, \cdots, q$ and $f_{3,1}(T) = f_{3,j}(T)$ with $j = 2, \cdots, m$ in Fig. I.1.

Notice that, p is the number of neurons in the first hidden layer, q is the number of neurons in the second hidden layer and m is the number of outputs.

Assumption 4: The designer should select the initial condition of the synaptic weights equals group by group, i. e., $w_{1,j}^{1}(0) = w_{i,j}^{1}(0)$, with $i = 2, \cdots, p$ and $j = 1, \cdots, n$, $w_{1,1}^{2}(0) = w_{k,i}^{2}(0)$, with $i = 1, \cdots, p$ and $k = 1, \cdots, q$, $w_{l,1}^{3}(0) = w_{l,k}^{3}(0)$, with $k = 2, \cdots, q$ and $l = 1, \cdots, m$.

I.2 Generalized model complexity reduction approach

Theorem 3: Consider the neural network whose architecture n-p-q-m is expressed in (I.1) and depicted in Fig. I.1, if Assumptions 3 and 4 are fulfilled, then such neural network can be reduced into a n-1-1-m equivalent architecture (see Fig. I.2).
Proof. Let us consider the architecture of Fig. I.1 corresponding to a three layers neural network, with the input-output mapping given by (I.1). For better understanding we decide to divide the model transformation in the two following steps.

Step 1.- Neural network training under the proposed assumptions.

The activation functions of the neural network model given by (I.1) are chosen according to Assumption 3. Then (I.1) becomes:

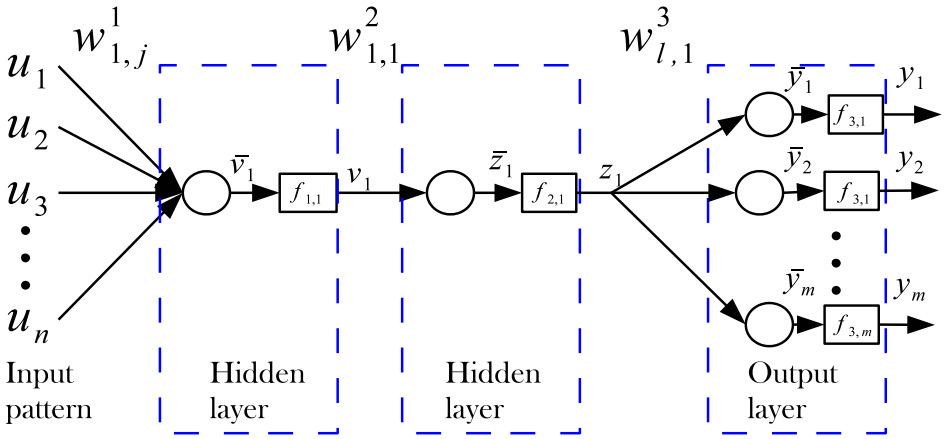


Figure I.2: Reduced multilayer perceptron of Narendra.

$$
\begin{aligned}
y_l &= f_{3,1}\left(\sum_{k=1}^{q}\left(w_{l,k}^{3} z_k\right)\right) \\
z_k &= f_{2,1}\left(\sum_{i=1}^{p}\left(w_{k,i}^{2} v_i\right)\right) \\
v_i &= f_{1,1}\left(\sum_{j=1}^{n}\left(w_{i,j}^{1} u_j\right)\right)
\end{aligned}
\tag{I.2}
$$

The initial condition of the synaptic weights are chosen according to Assumption 4.

Notice that, by choosing the initial condition according to Assumption 4, all the neurons of the first hidden layer receive the same inputs:

$$
v_i = f_{1,1}\left(\sum_{j=1}^{n}\left(w_{1,j}^{1} u_j\right)\right) \tag{I.3}
$$

with $i = 1, \cdots, p$, therefore, the outputs of the neurons of the first hidden layer are the same:

$$
v_1 = v_i \tag{I.4}
$$

Following the same idea, we can deduce that the outputs of the neurons of the second hidden layer are equals:

$$
z_1 = z_k \tag{I.5}
$$

where $k = 2, \cdots, q$. Finally this output is weighted by the different synaptic weights of the outputs neurons in order to generate the different outputs of the neural network:

$$
y_l = f_{3,1}\left(w_{l,1}^{3} z_1\right)
$$

where $l = 2, \cdots, m$. Since the initial conditions of the synaptic weights are set according to Assumption 4 and it has been shown that all the neurons in the hidden layers receive the same inputs, we can deduce that the final values of the synaptic weights are equals group by group, i. e., $w_{1,j}^{1*} = w_{i,j}^{1*}$, with $i = 2, \cdots, p$ and $j = 1, \cdots, n$, $w_{1,1}^{2*} = w_{k,i}^{2*}$, with $i = 1, \cdots, p$ and $k = 1, \cdots, q$, $w_{l,1}^{3*} = w_{l,k}^{3*}$, with $k = 2, \cdots, q$ and $l = 1, \cdots, m$.

Step 2.- Model transformation.

Once the neural network model given by (I.2) is trained under these two assumptions, we obtain:

$$y_l = f_{3,1}\left(\sum_{k=1}^{q}\left(w_{l,k}^{3*} z_k\right)\right) \tag{I.6}$$

$$z_k = f_{2,1}\left(\sum_{i=1}^{p}\left(w_{k,i}^{2*} v_i\right)\right) \tag{I.7}$$

$$v_i = f_{1,1}\left(\sum_{j=1}^{n}\left(w_{i,j}^{1*} u_j\right)\right) \tag{I.8}$$

Since $w_{1,j}^{1*} = w_{i,j}^{1*}$, with $i = 2, \cdots, p$ and $j = 1, \cdots, n$ (I.8) can be redefined as:

$$v_1 = v_i = f_{1,1}\left(\sum_{j=1}^{n}\left(w_{1,j}^{1*} u_j\right)\right) \tag{I.9}$$

and (I.7) becomes:

$$z_k = f_{2,1}\left(\sum_{i=1}^{p}\left(w_{k,i}^{2*} v_1\right)\right) \tag{I.10}$$

Since $w_{1,1}^{2*} = w_{k,i}^{2*}$, with $i = 1, \cdots, p$ and $k = 1, \cdots, q$, (I.10) becomes:

$$z_1 = z_k = f_{2,1}\left(\sum_{i=1}^{p}\left(w_{1,1}^{2*} v_1\right)\right) = f_{2,1}\left(p \times \left(w_{1,1}^{2*} v_1\right)\right) \tag{I.11}$$

Since $w_{l,1}^{3*} = w_{l,k}^{3*}$, with $k = 2, \cdots, q$ and $l = 1, \cdots, m$, (I.6) becomes:

$$y_l = f_{3,1}\left(\sum_{k=1}^{q}\left(w_{l,1}^{3*} z_1\right)\right) = f_{3,1}\left(q \times \left(w_{l,1}^{3*} z_1\right)\right) \tag{I.12}$$

Finally we have transformed the n-p-q-m model given by (I.2) into the following reduced model:

$$\begin{aligned} y_l &= f_{3,1}\left(q \times \left(w_{l,1}^{3*} z_1\right)\right) \\ z_1 &= f_{2,1}\left(p \times \left(w_{1,1}^{2*} v_1\right)\right) \\ v_1 &= f_{1,1}\left(\sum_{j=1}^{n}\left(w_{1,j}^{1*} u_j\right)\right) \end{aligned} \tag{I.13}$$

This completes the proof. ■

Remark 5: As well as for Theorem 1, the non reduced n-p-q-m model (I.2) and the reduced n-1-1-m model (I.13) are equivalents, therefore the reduced model keeps the same accuracy as the complex one.

IDENTIFICATION DE SYSTEMES UTILISANT LES RESEAUX DE NEURONES: UN COMPROMIS ENTRE PRECISION, COMPLEXITE ET CHARGE DE CALCULS.

RESUME : Ce rapport porte sur le sujet de recherche de l'identification boîte noire des systèmes non linéaires. Parmi toutes les techniques nombreuses et variées développées dans ce domaine de recherche ces dernières décennies, il semble toujours intéressant d'étudier l'approche réseau de neurones dans l'estimation de modèles de systèmes complexes. Même si des modèles précis ont été obtenus, les principaux inconvénients de ces techniques restent le grand nombre de paramètres nécessaire et, en conséquence, le coût important de calcul pour obtenir en pratique, le niveau de précision requis du modèle estimé. Par conséquent, motivés pour remédier à ces inconvénients, nous proposons une méthodologie complète et efficace d'identification de systèmes non linéaires offrant un équilibre quasi optimal entre précision, complexité et coût, en proposant, d'une part, de nouvelles structures de réseaux de neurones particulièrement adaptées à une utilisation très large en matière de modélisation systèmes non linéaires, et d'autre part, une simple et efficace technique de réduction de modèle, et, enfin, une procédure de réduction du coût de calcul. Il est important de noter que ces deux dernières techniques de réduction peuvent être appliquées à une très large gamme d'architectures de réseaux de neurones sous deux simples hypothèses spécifiques qui ne sont pas du tout contraignantes ni restrictives en pratique. Enfin, la dernière contribution importante de ce travail est d'avoir montré que cette phase d'estimation peut être obtenue dans un cadre robuste si la qualité des données d'identification le nécessite. Afin de valider la procédure d'identification proposée, des exemples d'application en simulation et sur un procédé réel, ont permis de valider de manière satisfaisante les propositions de cette thèse, confirmant tout l'intérêt de ce travail.

Mots clés : identification de système, modèles *boîte-noire*, non linéaire, réseaux de neurones, estimation robuste.

SYSTEM IDENTIFICATION USING NEURAL NETWORKS: A BALANCED ACCURACY, COMPLEXITY AND COMPUTATIONAL COST APPROACH.

ABSTRACT: This report concerns the research topic of black box nonlinear system identification. In effect, among all the various and numerous techniques developed in this field of research these last decades, it seems still interesting to investigate the neural network approach in complex system model estimation. Even if accurate models have been derived, the main drawbacks of these techniques remain the large number of parameters required and, as a consequence, the important computational cost necessary to obtain the convenient level of the model accuracy desired. Hence, motivated to address these drawbacks, we achieved a complete and efficient system identification methodology providing balanced accuracy, complexity and cost models by proposing, firstly, new neural network structures particularly adapted to a very wide use in practical nonlinear system modeling, secondly, a simple and efficient model reduction technique, and, thirdly, a computational cost reduction procedure. It is important to notice that these last two reduction techniques can be applied to a very large range of neural network architectures under two simple specific assumptions which are not at all restricting. Finally, the last important contribution of this work is to have shown that this estimation phase can be achieved in a robust framework if the quality of identification data compels it. In order to validate the proposed system identification procedure, application examples driven in simulation and on a real process, satisfactorily validated all the contributions of this thesis, confirming all the interest of this work.

Keywords : System identification, black-box models, nonlinear, neural networks, robust estimation.

ARTS ET MÉTIERS ParisTech

ParisTech INSTITUT DES SCIENCES ET TECHNOLOGIES PARIS INSTITUTE OF TECHNOLOGY