



# **WAVELET BASED NEURO-FUZZY SYSTEM IN FORECASTING OF DYNAMIC SYSTEMS**

**ABSTRACT  
THESIS**

**SUBMITTED FOR THE AWARD OF THE DEGREE OF**

**Doctor of Philosophy**

**IN**

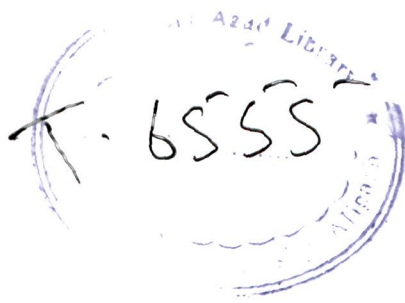
**ELECTRICAL ENGINEERING**

**BY**

***AHMAD BANAKAR***

**DEPARTMENT OF ELECTRICAL ENGINEERING  
ZAKIR HUSAIN COLLEGE OF ENGINEERING AND TECHNOLOGY  
ALIGARH MUSLIM UNIVERSITY  
ALIGARH (INDIA)**

**2007**



## **ABSTRACT**

In this thesis, application of wavelet functions in wavelet networks and neuro-fuzzy models are considered. The ability of localized analysis of wavelets jointly in frequency and time domain in addition to the learning ability of artificial neural network, prompt the Wavelet Neural Network (WNN) a superior system model for complex and seismic application. The presented work is an attempt to propose a comparative study for three types of wavelet function used in WNN, namely, Mexican hat, Morlet and Sinc wavelet functions. A conjunction of sigmoid and wavelet activation functions, by summation and product operators, is propose to combine the localize approximation property of wavelets with functional approximation properties of neural network (with sigmoid activation function).

In describing the behavior of many complex and ill-defined systems, precise mathematical models may fail to give satisfactory results. In such cases, fuzzy models are used to reflect the uncertainty of the systems in a proper way. In this thesis, Wavelet Neuro-Fuzzy (WNF) model is introduced where the consequent part of each fuzzy rule corresponds to a sub-WNN consisting of wavelet with the specified dilation value. Therefore, a WNF model has the ability to deal with impreciseness and uncertainty in a better way than ANFIS because of localizes region property of the wavelets. A hybrid learning method of gradient descent and genetic algorithm is applied to learn the parameters of the WNF model. A comparative study of the Parallel and the Series-Parallel configurations in parameter identification of the TSK neuro-fuzzy model is also presented in this work.

In the series of development of different recurrent network and neuron model, the presented work, based on WNN, proposes different types of recurrent neuron model to compare sigmoid and wavelet function for incorporating the dynamics inside the model of dynamic systems. Due to the dynamic behavior of recurrent network, they are suitable in dealing with the modeling of dynamic systems as compared to static behavior of feed-forward network.

A number of theorems cover universal approximation capability of all the proposed networks. An adaptive learning rate based of Lyapunov stability theorem is also applied to guarantee the convergence and the stability of the parameter learning process by determine the upper bound of learning rates.

The propose networks/models are tested upon six different types of dynamic systems and finally is applied to predict the Indian summer monsoon rainfall data.

## Motivation

During the nineteenth century Fourier transform, solved many problems in physics and engineering. This prominence led scientists and engineers to think of them as the preferred way to analyze phenomena of all kinds. This ubiquity forced a close examination of the method. As a result, through the twentieth century, mathematicians, physicists, and engineers came to realize a drawback of the Fourier transforms: they have trouble reproducing transient signals or signals with abrupt changes, such as the spoken word or the rap of a snare drum.

At the present scenario, wavelet decomposition emerges as a new powerful tool for function approximation due to its multi-resolution property. Recent advances have shown the existence of orthonormal wavelet bases, from which follows the availability of rates of convergence for approximation by wavelet based networks.

Several works has been done and so many works are going on for wavelets. Its application in neural network and neuro-fuzzy model gives tremendous performance for function approximation. However, until this time, selection of parameters and support of wavelet properties are mystery. Due to these discrepancies and multi resolution property of the wavelets, we have motivated to work with wavelet for forecasting and modeling applications of dynamic systems.



The organization of the thesis is as follows:

### **Chapter 2: Wavelet Networks**

A comparative study of two existing wavelet networks namely Wavelet Synapses Neural Network (WSNN) and Wavelet Activation Neural Network (WANN), based on three different wavelet function is presented in this chapter. Feed-forward neural networks show the ability to deal with complex problems and especially in input-output data systems. In addition, wavelet transformation has the ability of representing a function and revealing the properties of the function in the localized regions of the joint time frequency space. The chapter covers some basic concept of wavelet same as wavelet transform, Continuous Wavelet Transform (CWT) and Discrete Wavelet Transform (DWT). Three types of non-orthogonal wavelet are introduced in this section. These wavelets when used in feed-forward network give wavelet network.

### **Chapter 3: Generalized Wavelet Networks**

The main objective of this thesis is to improve existing one layer feed-forward network with SAF and WAF. Feed-forward neural networks show the ability to deal with complex problems and especially in input-output data systems. In addition, wavelet transformation has the ability of representing a function and revealing the properties of the function in the localized regions of the joint time frequency space. Due to above ability, in this chapter, combination of sigmoid and wavelet activation function is proposed. It has shown that a smart combination of these not only decreases the size of the network, it also increases the accuracy of the network. Two proposed wavelet neural network namely Summation Sigmoid-Wavelet (SS-W) and Multiplication Sigmoid-Wavelet (MS-W) neuron model are discussed in details. One

method for structure identification of the model is introduced. General approximation capability of the network has also been presented in this chapter with different theorems.

#### **Chapter 4: Neuro-Fuzzy Model**

This chapter serves as an introduction into the basic concept of parameter identification for neuro-fuzzy models. Two parameter identification schemes, namely Parallel (P) and the Series-Parallel (S-P) configurations, are described in this chapter. A combination of these two configurations is proposed for neuro-fuzzy models. Modified mountain clustering is applied to neuro-fuzzy models for structure determination and initialization of the neuro-fuzzy models. An algorithm with adaptive learning rate is used to learn learning parameters of the model. Convergence of the learning procedure is guaranteed by Lyapunov stability theorem.

#### **Chapter 5: Wavelet Neuro-Fuzzy Model**

This chapter discusses about the wavelet neuro-fuzzy model. The proposed network in chapter 3 with better performance is used in the consequent part of each fuzzy rule in TSK neuro-fuzzy model that results WNF model. A hybrid of Genetic Algorithm and Gradient Descent has been employed to learn the model parameters.

#### **Chapter 6: Recurrent Wavelet Networks**

In this chapter, recurrent neuron models are introduced. Due to the dynamic behavior of recurrent networks, they are suitable in dealing with the modeling of dynamic systems as compared to static behavior of feed-forward network. The quantitative behavior of the sigmoid and wavelet activation functions for dealing with

and saving the dynamic of systems are considered. The general approximation properties of the recurrent neuron models are also evaluated. Since the convergence analysis plays an important role in the recurrent networks, the Lyapunov stability approach is employed to guarantee the convergence of network.

#### **Chapter 7: Case study, Indian Monsoon Rain-Fall**

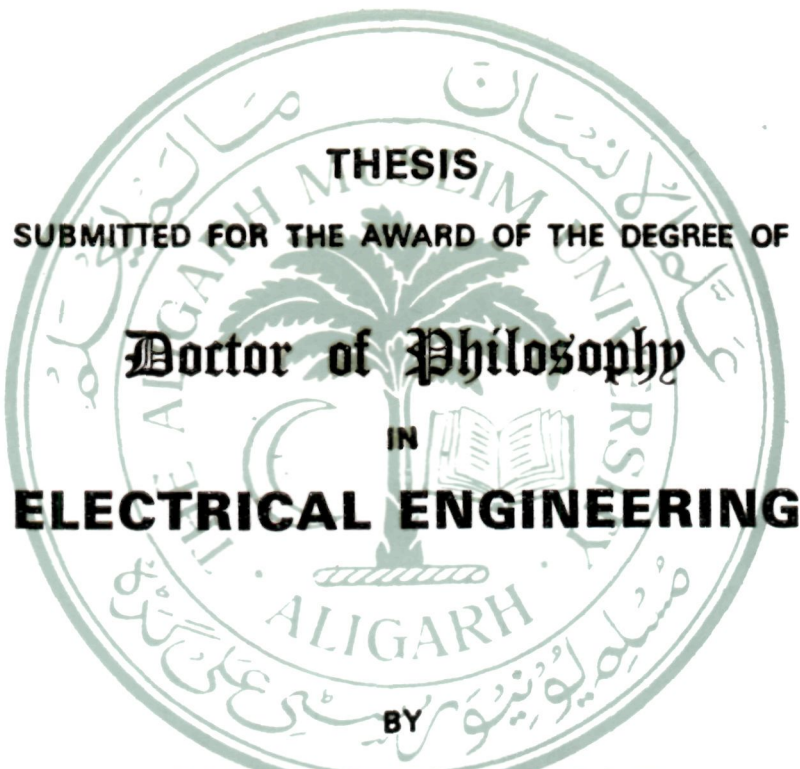
The agricultural economy of India is closely linked to the performance of summer monsoon rainfall all over India. The ability to understand and predict circulation and rainfall during the Asian summer monsoon on various time-scales is of prime importance to the economy of several nations of this region because of its affect on agriculture, drinking water, transportation, health, power, and the very livelihood of billions people living in the monsoon region. Due to these reason, in this chapter, all the proposed networks are tested on rainfall data.

#### **Chapter 8: Conclusion**

Finally, conclusions of the thesis and suggestions for the future work have been covered in chapter 8.



# **WAVELET BASED NEURO-FUZZY SYSTEM IN FORECASTING OF DYNAMIC SYSTEMS**



**THESIS**

**SUBMITTED FOR THE AWARD OF THE DEGREE OF**

**Doctor of Philosophy**

**IN**

**ELECTRICAL ENGINEERING**

**BY**

**AHMAD BANAKAR**

**DEPARTMENT OF ELECTRICAL ENGINEERING  
ZAKIR HUSAIN COLLEGE OF ENGINEERING AND TECHNOLOGY  
ALIGARH MUSLIM UNIVERSITY  
ALIGARH (INDIA)**

**2007**



T6555

## **THESIS APPROVAL SHEET**

The thesis entitled “WAVELET BASED NEURO-FUZZY SYSTEM IN FORECASTING OF DYNAMIC SYSTEMS” submitted by Mr. Ahmad Banakar, Department of Electrical Engineering, Zakir Husain College of Engineering and Technology, Aligarh Muslim University, Aligarh, India, is approved for the award of the degree of Doctor of Philosophy (Ph.D.).

-----  
Internal Examiner

-----  
External Examiner

-----  
Chairman,  
Dept.of Electrical Engg.,  
AMU, Aligarh.

## CERTIFICATE

This is to certify that the thesis entitled "WAVELET BASED NEURO-FUZZY SYSTEM IN FORECASTING OF DYNAMIC SYSTEMS", which is being submitted by **Mr. Ahmad Banakar** for the award of the degree of **Doctor of Philosophy in Electrical Engineering** of the Faculty of Engineering and Technology, Aligarh Muslim University, Aligarh, India, is entirely based on the work carried out by him under my supervision. The work reported, embodies the work of candidate himself and is original one, has not been submitted to any other University or Institution for the award of any degree or diploma, according to best of my knowledge.

Dated: 15/09/2007

  
(Dr. Mohammad)Fazle Azeem  
(Supervisor)

## ABSTRACT

In this thesis, application of wavelet functions in wavelet networks and neuro-fuzzy models are considered. The ability of localized analysis of wavelets jointly in frequency and time domain in addition to the learning ability of artificial neural network, prompt the Wavelet Neural Network (WNN) a superior system model for complex and seismic application. The presented work is an attempt to propose a comparative study for three types of wavelet function used in WNN, namely, Mexican hat, Morlet and Sinc wavelet functions. A conjunction of sigmoid and wavelet activation functions, by summation and product operators, is propose to combine the localize approximation property of wavelets with functional approximation properties of neural network (with sigmoid activation function).

In describing the behavior of many complex and ill-defined systems, precise mathematical models may fail to give satisfactory results. In such cases, fuzzy models are used to reflect the uncertainty of the systems in a proper way. In this thesis, Wavelet Neuro-Fuzzy (WNF) model is introduced where the consequent part of each fuzzy rule corresponds to a sub-WNN consisting of wavelet with the specified dilation value. Therefore, a WNF model has the ability to deal with impreciseness and uncertainty in a better way than ANFIS because of localizes region property of the wavelets. A hybrid learning method of gradient descent and genetic algorithm is applied to learn the parameters of the WNF model. A comparative study of the Parallel and the Series-Parallel configurations in parameter identification of the TSK neuro-fuzzy model is also presented in this work.

In the series of development of different recurrent network and neuron model, the presented work, based on WNN, proposes different types of recurrent neuron model to compare sigmoid and wavelet function for incorporating the dynamics inside the model of dynamic systems. Due to the dynamic behavior of recurrent network, they are suitable in dealing with the modeling of dynamic systems as compared to static behavior of feed-forward network.

A number of theorems cover universal approximation capability of all the proposed networks. An adaptive learning rate based of Lyapunov stability theorem is also applied to guarantee the convergence and the stability of the parameter learning process by determine the upper bound of learning rates.

The propose networks/models are tested upon six different types of dynamic systems and finally is applied to predict the Indian summer monsoon rainfall data.



## ACKNOWLEDGMENT

I thank God to give me three years more to complete this research and allow me to see and understand another culture. I receive to this point that all human have a lot of co-operation with together to forget differences between themselves.

It is my great pleasure to express my appreciation to everyone who is directly or indirectly related with this research.

I should give my full respect to my supervisor, Dr. Mohammad Fazle Azeem, His always-cheerful conversations, a friendly behavior, and his unique way to make his students realize their hidden talents are extraordinary. I heartily acknowledge his constant encouragements, genuine efforts and being tolerant to my irregularities during the entire course of this work.

I express my sincere gratitude to Professor M.S. Jamil Asghar who always answers me kindly and make co-operation with me.

I would like to thanks the chairman of the Department of Electrical Engineering, Professor Prabhat Kumar, and Ex-Chairman of the Department of Electrical Engineering, Professor A. K. Gupta, for providing me the facility for successful completion of this research work.

I also extend my thanks to all teachers and staff member of the Department of Electrical Engineering.

It is my pleasure to remember myself and mention the name of my close friends and seniors in Aligarh, Professor Asef, Dr. Rahmani, Roohi, Farzaneh, Huseini, Bazae, Golami, Alidad, Basir, Heidari, Haqi, Mohammadianfar, Kord, Oanaq, Bameri, Mirakizadeh, Faraz, Mojarabian, Monazami, Adhami, Kamarzarin, Soltanzadeh, Khubfekr, Vinod Kumar, Jahangir Eqbal, Golam Abas, Akbar Ali Jafri.

I cannot forget the Indian Government and the UGC office for helping me, which supported me and allowed me to take JRF scholarship. I respect all Indian and this ground that respect me, patience to me when I was angry and smile with me when I was glad.

Finally, I wish to extend my fully thanks to my family:


My father, when I started my course he was waiting for its completion.

But he expired. I wish Allah for "rahmat" on him.

My mother, who staying with me, I do not know how can I write a sentence to thanks her. I can say her only "I love you!"

My brother, Mr. Abdorahim Banker & his wife Maryam, my sisters Efat and Mahboubeh and my heart in India.

My relations, Mr. Amiri, Mohammad Saleh & his wife, Zeinab, Mohsen, Fatemeh, Zahra



18-07-07

Ahmad Banakar

*Dedicated to*

*My Parents*

# CONTENTS

	<b>Page No.</b>
Thesis Approval Sheet	i
Certificate	ii
Abstract	iii
Acknowledgment	iv
Contents	vi
List of Figures	xviii
List of Tables	xix
List of principal Symbols and Abbreviations	xx
<b>Chapter 1: Literature Review</b>	<b>1</b>
1-1 Introduction	1
1-2 Identification	2
1-3 Soft Computing	2
1-3.1 What is absorbed in Soft Computing?	3
1-3.2 Importance of Soft Computing	4
1-4 Wavelet	4
1-5 Motivation	7
1-6 Scope of the Thesis	8
1-7 Organization of Thesis	10
1-8 Description of Some Dynamic Systems	13
<b>Chapter 2: Wavelet Networks</b>	<b>19</b>
2-1 Introduction	19

2-2	Artificial Neural Network	21
2-3	Wavelet	22
2-3.1	Continuous time Wavelet Transform (CWT)	23
2-2.2	Discrete time Wavelet Transform (DWT)	25
2-3.3	Types of the wavelet	25
	a) Mexican hat wavelet	27
	b) Morlet wavelet	27
	c) Sinc (Shannon) wavelet	29
2-4	Wavelet Neural Network	30
2-4.1	Wavelet Synapses Neural Network	31
2-4.2	Wavelet Activation function Neural Network	32
2-5	Gradient Descent learning of parameters	34
2-6	Simulation Results	37
2-7	Conclusions	51
<b>Chapter 3:</b>	<b>Generalized Wavelet Networks</b>	<b>53</b>
3-1	Introduction	53
3-2	Localized Wavelet Neural Network (LWNN)	55
3-3	Sigmoid-Wavelet Neuron Networks	56
3-3.1	Feed-forward network	57
3-3.2	Summation Sigmoid-Wavelet (SS-W) Neuron	57
3-3.3	Multiplication Sigmoid-Wavelet (MS-W) Neuron	58
3-4	Universal approximation of the S-W neuron networks	59
3-5	Gradient Descent learning of parameters	61
3-6	Structure determination of S-W neuron networks	64
3-7	Simulation Results	66

3-8	Conclusions	102
<b>Chapter 4:</b>	<b>Parameter Identification of Neuro-Fuzzy model</b>	<b>103</b>
4-1	Introduction	103
4-2	Neuro-Fuzzy model	106
4-3	Configurations for Parameter Identification	110
4-3.1	Parallel Configuration	110
4-3.2	Series-Parallel configuration	112
4-3.3	Proposed Configurations	113
	a) Consequent Series-Parallel configuration	115
	b) Premise Series-Parallel configuration	116
4-4	Learning procedure	116
4-4.1	Structure determination and Initialization	117
4-4.2	Training	117
	a) Gradient Descent Technique of the parameters	118
	b) Learning Convergence theorems	119
	c) Adaptive learning rate	120
4-5	Simulation Results	122
4-6	Conclusions	139
<b>Chapter 5:</b>	<b>Wavelet Neuro-Fuzzy Model</b>	<b>141</b>
5-1	Introduction	141
5-2	Wavelet Neuro-Fuzzy	143
5-2.1	Architecture of Proposed Wavelet Neuro-Fuzzy	143
5-3	Genetic Algorithm and Gradient Descent	146
5-3.1	Basic of the Genetic Algorithm	146
5-3.2	Components of GA	148

a)	Solution Representation (Encoding & Decoding)	148
b)	Initialization	149
c)	Evaluation function	149
d)	Selection, Crossover, Mutation and Reproduction	150
5-3.3	Gradient Descent learning of parameters	150
5-3.4	Learning Convergence theorem	152
5-4	Simulation Results	153
5-5	Conclusions	182
<b>Chapter 6:</b>	<b>Recurrent Wavelet Networks</b>	<b>183</b>
6-1	Introduction	183
6-2	Structures of Recurrent Neuron Models	186
6-2.1	Sigmoid-Recurrent Wavelet (S-RW) Neuron	186
6-2.2	Recurrent Sigmoid–Wavelet (RS-W) Neuron	187
6-2.3	Feedback to Sigmoid from Wavelet (FS-W) Neuron	188
6-2.4	Feedback to Wavelet from Sigmoid (FW -S) Neuron	189
6-2.5	Recurrent Neuron (RN)	190
6-3	Universal approximation of the proposed recurrent neuron models	191
6-4	Gradient Descent learning of parameters	194
6-4.1	Sigmoid-Recurrent Wavelet (S-RW) Neuron	194
6-4.2	Recurrent Sigmoid–Wavelet (RS-W) Neuron	195
6-4.3	Feedback to Sigmoid from Wavelet (FS-W) Neuron	196
6-4.4	Feedback to Wavelet from Sigmoid (FW-S) Neuron	197
6-4.5	Recurrent Neuron (RN)	197
6-4.6	Stability analysis of the recurrent neuron models	198
6-5	Simulation Results	203

6-6	Conclusions	235
<b>Chapter 7:</b>	<b>CASE STUDY: Indian Summer Monsoon Rainfall</b>	<b>237</b>
7-1	Introduction	237
7-2	Forecasting Ability of Rainfall Data	239
7-3	Simulation Results	241
7-5	Conclusions	251
<b>Chapter 8:</b>	<b>Conclusions &amp; Future work</b>	<b>253</b>
8-1	Conclusions	253
8-2	Future work	255
<b>Appendix A:</b>	<b>Universal Approximation Theory</b>	<b>257</b>
<b>Appendix B:</b>	<b>Modified Mountain Clustering</b>	<b>269</b>
<b>Appendix C:</b>	<b>Stability and Convergence Analysis of Gradient Descent Learning Algorithm</b>	<b>273</b>
<b>References</b>		<b>291</b>
<b>Bio-data</b>		<b>302</b>

## LIST OF FIGURES

<b>Fig. No.</b>	<b>Title</b>
<b>Fig. 1.1:</b>	Control action of an operator
<b>Fig. 2.1:</b>	Architecture of the single layer neural network
<b>Fig. 2.2:</b>	Representation of (a) a wave and (b) a wavelet
<b>Fig. 2.3:</b>	Mexican hat wavelet functions
<b>Fig. 2.4:</b>	Morlet wavelet functions
<b>Fig. 2.5:</b>	Sinc wavelet functions
<b>Fig. 2.6:</b>	(a) Simple neuron model (b) Wavelet activation function neuron model and (c) Wavelet synapses neuron model
<b>Fig. 2.7:</b>	Wavelet Synapses Neural Network (WSNN)
<b>Fig. 2.8:</b>	Wavelet Activation Function Neural Network (WANN)
<b>Fig. 2.9:</b>	Actual output and network output with WSNN (Mexican hat) network and the error for Example 1
<b>Fig. 2.10:</b>	Actual output and network output with WANN (Mexican hat) network and the error for Example 2
<b>Fig. 2.11:</b>	Actual output and network output with WANN (Mexican hat) network and the error for Example 3
<b>Fig. 2.12:</b>	Actual output and network output with WANN (Mexican hat) network and the error for Example 4
<b>Fig. 2.13:</b>	Actual output and network output with WANN (Mexican hat) network and the error for Example 5
<b>Fig. 2.14:</b>	Actual output and network output with WANN (Mexican hat) network and the error for Example 6
<b>Fig. 2.15:</b>	Actual output and network output with WANN (Mexican hat) network and the error for Example 7
<b>Fig. 2.16:</b>	Actual output and network output with WANN (Mexican hat) network and the error for Example 8
<b>Fig. 3.1:</b>	Localized Wavelet Neural Network (LWNN)



- Fig. 3. 2:** Feed-forward Neural Network
- Fig. 3.3:** Summation Sigmoid-Wavelet (SS-W) neuron network
- Fig. 3.4:** Multiplication Sigmoid-Wavelet (MS-W) neuron network
- Fig. 3.5:** Wavelet with different scaling factor and shifting
- Fig. 3.6:** Algorithm for Structure Determination
- Fig. 3.7:** Learning pattern of feed-forward network with SS-W neuron network using Morlet activation function with scaling factor  $a=1, 2$  &  $3$  for Example 1
- Fig. 3.8:** Learning pattern of SS-W neuron network with all wavelet functions for Example 1
- Fig. 3.9:** Learning pattern of feed-forward network with MS-W neuron network using Morlet activation function with scaling factor  $a=1, 2$  &  $3$  for Example 1
- Fig. 3.10:** Learning pattern of MS-W neuron network with all wavelet functions for Example 1
- Fig. 3.11:** Actual output and network output with MS-W (Morlet) neuron network and the error for Example 1
- Fig. 3.12:** Learning pattern of feed-forward network with SS-W neuron network using Morlet activation function with scaling factor  $a=1, 2, 3$  &  $4$  for Example 2
- Fig. 3.13:** Learning pattern of SS-W neuron network with all wavelet functions for Example 2
- Fig. 3.14:** Learning pattern of feed-forward network with MS-W neuron network using Morlet activation function with scaling factor  $a=1, 2, 3$  &  $4$  for Example 2
- Fig. 3.15:** Learning pattern of MS-W neuron network with all wavelet functions for Example 2
- Fig. 3.16:** Actual output & predicted output of MS-W (Morlet) neuron network and the error for Example 2
- Fig. 3.17:** Learning pattern of feed-forward network with SS-W neuron network using Morlet activation function with scaling factor  $a=1, 2$  &  $3$  for Example 3
- Fig. 3.18:** Learning pattern of SS-W neuron network with all wavelet functions for Example 3
- Fig. 3.19:** Learning pattern of feed-forward network with MS-W neuron network with scaling factor  $a=1, 2$  &  $3$  for Example 3

- Fig. 3.20:** Learning pattern of MS-W neuron network with all wavelet functions for Example 3
- Fig. 3.21:** Actual output & predicted output of MS-W (Morlet) neuron network and the error for Example 3
- Fig. 3.22:** Learning pattern of feed-forward network with SS-W neuron network using Morlet activation function with scaling factor  $a=1, 2 \& 3$  for Example 4
- Fig. 3.23:** Learning pattern of SS-W neuron network with all wavelet functions for Example 4
- Fig. 3.24:** Learning pattern of feed-forward network with MS-W neuron network with scaling factor  $a=1, 2 \& 3$  for Example 4
- Fig. 3.25:** Learning pattern of MS-W neuron network with all wavelet functions for Example 4
- Fig. 3.26:** Actual output & predicted output of SS-W (Morlet) neuron network and the error for Example 4
- Fig. 3.27:** Learning pattern of feed-forward network with SS-W neuron network using Morlet activation function with scaling factor  $a=1, 2, 3 \& 4$  for Example 5
- Fig. 3.28:** Learning pattern of SS-W neuron network with all wavelet functions for Example 5
- Fig. 3.29:** Learning pattern of feed-forward network with MS-W neuron network with scaling factor  $a=1, 2 \& 3$  for Example 5
- Fig. 3.30:** Learning pattern of MS-W neuron network with all wavelet functions for Example 5
- Fig. 3.31:** Actual output & output of MS-W (Morlet) neuron network and the error for Example 5
- Fig. 3.32:** Learning pattern of feed-forward network with SS-W neuron network using Morlet activation function with scaling factor  $a=1, 2 \& 3$  for Example 6
- Fig. 3.33:** Learning pattern of SS-W neuron network with all wavelet functions for Example 6
- Fig. 3.34:** Learning pattern of feed-forward network with MS-W neuron network with scaling factor  $a=1, 2, 3 \& 4$  for Example 6
- Fig. 3.35:** Learning pattern of MS-W neuron network with all wavelet functions for Example 6
- Fig. 3.36:** Actual output & output of MS-W (Mexican hat) neuron network and the error for Example 6
- Fig. 3.37:** Performance index of feed-forward SS-W neuron network with different scaling factor ' $a$ ' for all examples

- Fig. 3.38:** Performance index of feed-forward MS-W neuron network with different scaling factor ' $\alpha$ ' for all examples
- Fig. 4.1:** Neuro-Fuzzy model
- Fig. 4.2:** Parallel configuration
- Fig. 4.3:** Series-parallel configuration
- Fig. 4.4:** Proposed parallel and Series-parallel configuration
- Fig. 4.5:** Output of the plant feedback to the consequent part and the output of model feedback to the premise part
- Fig. 4.6:** Output of the plant feedback to the premise and the output of model feedback to the consequent part
- Fig. 4.7:** Learning algorithm for Neuro-Fuzzy model
- Fig. 4.8:** Learning pattern of all configurations for Example 1
- Fig. 4.9:** Initial membership functions of the normalized inputs for Example 1
- Fig. 4.10:** Learned membership functions of the normalized inputs for Example 1
- Fig. 4.11:** Actual output & model output with CS-P configuration and the error for Example 1
- Fig. 4.12:** Learning pattern of all configurations for Example 2
- Fig. 4.13:** Initial membership functions of the normalized inputs for Example 2
- Fig. 4.14:** Learned membership functions of the normalized inputs for Example 2
- Fig. 4.15:** Actual output & model output with CS-P configuration and the error for Example 2
- Fig. 4.16:** Learning pattern of all configurations for Example 3
- Fig. 4.17:** Initial membership functions of the normalized inputs for Example 3
- Fig. 4.18:** Learned membership functions of the normalized inputs for Example 3
- Fig. 4.19:** Actual output & model output with CS-P configuration and the error for Example 3
- Fig. 4.20:** Learning pattern of all configurations for Example 4
- Fig. 4.21:** Initial membership functions of the inputs for Example 4
- Fig. 4.22:** Learned membership functions of the normalized inputs for Example 4

- Fig. 4.23:** Actual output & model output with CS-P configuration and the error for Example 4
- Fig. 4.24:** Learning pattern of all configurations for Example 5
- Fig. 4.25:** Initial membership functions of the normalized inputs for Example 5
- Fig. 4.26:** Learned membership functions of the normalized inputs for Example 5
- Fig. 4.27:** Actual output & model output with CS-P configuration and the error for Example 5
- Fig. 5.1:** Proposed Wavelet Neuro-Fuzzy Model
- Fig. 5. 2:** Algorithm for initialized and learning of the learning parameters
- Fig. 5.3:** *Maximum fitness of GA up to each generation for Example 1*
- Fig. 5.4:** Initialized membership functions, learned by GA, of the normalized inputs for Example 1
- Fig. 5.5:** Learned membership functions, obtained by GA & GD, of the normalized inputs for Example 1
- Fig. 5.6:** Learning pattern of WNF model by Genetic Algorithm and Gradient Descent for Example 1
- Fig. 5.7:** Actual output & WNF model output and the error for Example 1
- Fig. 5.8:** Maximum fitness of GA up to each generation for Example 2
- Fig. 5.9:** Initialized membership functions, learned by GA, of the normalized inputs for Example 2
- Fig. 5.10:** Learned membership functions, obtained by GA & GD, of the normalized inputs for Example 2
- Fig. 5.11:** Learning pattern of WNF model by Genetic Algorithm and Gradient Descent for Example 2
- Fig. 5.12:** Actual output & WNF model output and the error for Example 2
- Fig. 5.13:** Maximum fitness of GA up to each generation for Example 3
- Fig. 5.14:** Initialized membership functions, learned by GA, of the normalized inputs for Example 3
- Fig. 5.15:** Learned membership functions, obtained by GA & GD, of the normalized inputs for Example 3
- Fig. 5.16:** Learning pattern of WNF model by Genetic Algorithm and Gradient Descent for Example 3

- Fig. 5.17:** Actual output & WNF model output and the error for Example 3
- Fig. 5.18:** Maximum fitness of GA up to each generation for Example 4
- Fig. 5.19:** Initialized membership functions, learned by GA, of the normalized inputs for Example 4
- Fig. 5.20:** Learned membership functions, obtained by GA & GD, of the normalized inputs for Example 4
- Fig. 5.21:** Learning pattern of WNF model by Genetic Algorithm and Gradient Descent for Example 4
- Fig. 5.22:** Actual output & WNF model output and the error for Example 4
- Fig. 5.23:** Maximum fitness of GA up to each generation for Example 5
- Fig. 5.24:** *Initialized membership functions, learned by GA, of the normalized inputs for Example 5*
- Fig. 5.25:** Learned membership functions, obtained by GA & GD, of the normalized inputs for Example 5
- Fig. 5.26:** Learning pattern of WNF model by Genetic Algorithm and Gradient Descent for Example 5
- Fig. 5.27:** Actual output & WNF model output and the error for Example 5
- Fig. 5.28:** Maximum fitness of GA up to each generation for Example 6
- Fig. 5.29:** Initialized membership functions, learned by GA, of the normalized inputs for Example 6
- Fig. 5.30:** Learned membership functions, obtained by GA & GD, of the normalized inputs for Example 6
- Fig. 5.31:** Learning pattern of WNF model by Genetic Algorithm and Gradient Descent for Example 6
- Fig. 5.32:** Actual output & WNF model output and the error for Example 6
- Fig. 6.1:** Summation Sigmoid-Recurrent Wavelet (SS-RW) neuron model
- Fig. 6.2:** Multiplication Sigmoid-Recurrent Wavelet (MS-RW) neuron model
- Fig. 6.3:** Summation Recurrent Sigmoid-Wavelet (SRS-W) neuron model
- Fig. 6.4:** Multiplication Recurrent Sigmoid-Wavelet (MRS-W) neuron model
- Fig. 6.5:** Summation Feedback to Sigmoid from Wavelet (SFS-W) neuron model
- Fig. 6.6:** Multiplication Feedback to Sigmoid from Wavelet (MFS-W) neuron model

- Fig. 6.7:** Summation Feedback to Wavelet from Sigmoid (SFW -S) neuron model
- Fig. 6.8:** Multiplication Feedback to Wavelet from Sigmoid (MFW -S) neuron model
- Fig. 6.9:** Summation Recurrent Neuron (SRN)
- Fig. 6.10:** Multiplication Recurrent Neuron (MRN)
- Fig. 6.11:** Learning pattern of feed-forward network with recurrent SS-W neuron models for Example 1
- Fig. 6.12:** Learning pattern of feed-forward network with recurrent MS-W neuron models for Example 1
- Fig. 6.13:** Actual output and network output with MFW-S model and the error for Example 1
- Fig. 6.14:** Learning pattern of feed-forward network with recurrent SS-W neuron models for Example 2
- Fig. 6.15:** Learning pattern of feed-forward network with recurrent MS-W neuron models for Example 2
- Fig. 6.16:** Actual output and network output with MFW-S model and the error for Example 2
- Fig. 6.17:** Learning pattern of feed-forward network with recurrent SS-W neuron models for Example 3
- Fig. 6.18:** Learning pattern of feed-forward network with recurrent MS-W neuron models for Example 3
- Fig. 6.19:** Actual output and network output with SS-RW model and the error for Example 3
- Fig. 6.20:** Learning pattern of feed-forward network with recurrent SS-W neuron models for Example 4
- Fig. 6.21:** Learning pattern of feed-forward network with recurrent MS-W neuron models for Example 4
- Fig. 6.22:** Actual output and network output with SRN model and the error for Example 4
- Fig. 6.23:** Learning pattern of feed-forward network with recurrent SS-W neuron models for Example 5
- Fig. 6.24:** Learning pattern of feed-forward network with recurrent MS-W neuron models for Example 5
- Fig. 6.25:** Actual output and network output with MFW-S model and the error for Example 5

## LIST OF TABLES

**Table No. Title**

**Table 2.1:** Performance Index of WSNN and WANN networks for different wavelet activation functions

**Table 3.1:** Performance Index ( $J$ ) with different networks and wavelet functions for Example 1

**Table 3.2:** Performance Index ( $J$ ) with different networks and wavelet functions for Example 2

**Table 3.3:** Performance Index ( $J$ ) with different networks and wavelet functions for Example 3

**Table 3.4:** Performance Index ( $J$ ) with different networks and wavelet functions for Example 4

**Table 3.5:** Performance Index ( $J$ ) with different networks and wavelet functions for Example 5

**Table 3.6:** Performance Index ( $J$ ) with different networks and wavelet functions for Example 6

**Table 4.1:** Performance index of different configuration identification models

**Table 5.1:** Performance Index ( $J$ ) of MS-W neuron model, NF and WNF models

**Table 6.1:** Performance Index for Recurrent SS-W neuron models

**Table 7.1:** Performance Index ( $J$ ) with different networks for rainfall data

- Fig. 6.26:** Learning pattern of feed-forward network with recurrent SS-W neuron models for Example 6
- Fig. 6.27:** Learning pattern of feed-forward network with recurrent MS-W neuron models for Example 6
- Fig. 6.28:** Actual output and network output with MS-RW model and the error for Example 6
- Fig. 7.1:**  $H$  estimate for the original and random rainfall data
- Fig. 7.2:** Performance index of Wavelet Neural Network (WNN) with different scaling factor  $a$
- Fig. 7.3:** Performance index of feed-forward SS-W neuron model with different scaling factor  $a$
- Fig. 7.4:** Performance index of feed-forward MS-W neuron model with different scaling factor  $a$
- Fig. 7.5:** Maximum fitness of GA up to each generation for rainfall data
- Fig. 7.6:** Learning pattern of WNF model by Gradient Descent for rainfall data
- Fig. 7.7:** Learned membership function, obtained by GA & GD, of the normalized input  $x(t-1)$  for rainfall data
- Fig. 7.8:** Learned membership function, obtained by GA & GD, of the normalized input  $x(t-2)$  for rainfall data
- Fig. 7.9:** Learned membership function, obtained by GA & GD, of the normalized input  $x(t-3)$  for rainfall data
- Fig. 7.10:** Learned membership function, obtained by GA & GD, of the normalized input  $x(t-4)$  for rainfall data
- Fig. 7.11:** Learned membership function, obtained by GA & GD, of the normalized input  $x(t-5)$  for rainfall data
- Fig. 7.12:** Actual output and model output with WNF model and the error for Indian Monsoon Rainfall data



## LIST OF PRINCIPAL SYMBOLS AND ABBREVIATIONS

<b>Symbols</b>	<b>Description</b>
$t$	Time
$a_0, a_k \text{ \& } b_k$	Coefficients in Fourier series
$f$	Function
$k$	Integer variable
$\pi$	Pie number
$u$	Input function
$y$	Dynamic function
$\sin$	Sine function
$\cos$	Cosine function
$CO_2$	Oxide Carbon
$a$	Scaling factor in wavelet function
$b$	Shifting in wavelet function
$\Phi$	Activation function
$M$	Number of hidden layer in neural network
$n$	Number of inputs
$b$	Bias for each hidden neuron in neural network
$C$	Weights between first (input) & second (hidden) layer
$W$	Weights between second (hidden) & third (output) layer
$\hat{y}$	Output of models\networks
$j$	Number of hidden neurons in neural network
$i$	Number of inputs
$\psi$	Wavelet or mother wavelet function
$*$	Complex conjugation

$\mathfrak{R}$	Real number
$L^2$	Square integrable function
$z$	Integer variables
$\Psi(\omega)$	The Fourier transform of $\psi(t)$ .
$n$	Integer number
$\exp$	Exponential function
$(x, y)$	Data sample in real numbers
$M$	Maximum number of scaling factor
$\hat{Y}_{WSNN}$	Output of WSNN network
$\hat{Y}_{WANN}$	Output of WANN network
$P$	Number of dataset
$\nu$	Learning parameter
$q$	Epoch
$\alpha_m$	Momentum update coefficient
$J$	Performance index
$e$	Error
$\gamma_f$	Decay factor
$\eta$	Adaptive learning rate
$O^1$	Output of layer one
'G'	S-W neurons
$\theta$	Sigmoid activation function
$\  \cdot \ $	Norm function
$\varepsilon$	Epsilon number
$h$	Real function
$\mathcal{A}$	Compact set

$\rho$	Integer value
$C_w$	Weights to inputs signal for wavelet activation function
$C_s$	Weights to inputs signal for sigmoid activation function
$[t_0, T]$	Time-interval
$X$	Inputs set
$R$	Rule
$\mu$	Firing strength of rules
$A$	Linguistic labels of fuzzy sets
$m$	Number of rule
$\tau_{iq}$	Corresponding delay of input
$\tau_o$	Corresponding delay of output
$u_q$	$q^{\text{th}}$ input
$\bar{x}$	Center of Gaussian membership function
$\sigma$	Standard deviation of Gaussian membership function
$X'$	Inputs set of the Consequent part or local model
$\gamma$	Coefficient
$y^o$	Model \ network output
$\hat{y}_{WNN}$	Output of WNN
$c$	Center matrix
$h$	Length of binary string
$p_c$	Probability of crossover
$p_m$	Probability of mutation
$A_1^2$	Premise variable membership function for input 1 & rule 2
$A_1^U$	Learned premise variable membership functions for input 1 & rule 2
$\hat{Y}^1$	Outputs of initialized local model

$\hat{Y}^{lf}$	Outputs of learned local model
$Q_w$	Feedback weight in S-RW
$Q_s$	Feedback weight in RS-W
$Q_{ws}$	Feedback weight in FS-W
$Q_{sw}$	Feedback weight in FW-S
$Q$	Feedback weight in RN
$X_t$	Time series
$m$	Mean value of original time series
$N$	Number of observations
$R$	Cumulative deviations
$H$	Hurst exponent

<b>Abbreviations</b>	<b>Description</b>
<b>ANN</b>	Artificial Neural Networks
<b>SC</b>	Soft Computing
<b>FL</b>	Fuzzy Logic
<b>NN</b>	Neural Computing
<b>EC</b>	Evolutionary Computation
<b>ML</b>	Machine Learning
<b>PR</b>	Probabilistic Reasoning
<b>MIQ</b>	Machine Intelligence Quotient
<b>WNN</b>	Wavelet Neural Networks
<b>NN</b>	Neural Network
<b>SAF</b>	Sigmoid Activation Function

<b>WAF</b>	Wavelet Activation Functions
<b>WNF</b>	Wavelet Neuro- Fuzzy
<b>TSK</b>	Takagi-Sugeno-Kang
<b>WSNN</b>	Wavelet Synapses Neural Network
<b>WANN</b>	Wavelet Activation Neural Network
<b>CWT</b>	Continuous Wavelet Transform
<b>DWT</b>	Discrete Wavelet Transform
<b>SS-W</b>	Summation Sigmoid-Wavelet
<b>MS-W</b>	Multiplication Sigmoid-Wavelet
<b>P</b>	Parallel
<b>S-P</b>	Series-Parallel
<b>FFNN</b>	Feed Forward Neural Network
<b>WT</b>	Wavelet Transform
<b>TFR</b>	Time-Frequency Representation
<b>GD</b>	Gradient Descent
<b>LWNN</b>	Localized Wavelet Neural Network
<b>S-W</b>	Sigmoid-Wavelet
<b>H.N.</b>	Hidden Neuron
<b>Ex</b>	Example
<b>ANFIS</b>	Adaptive Neuro-Fuzzy Inference System
<b>MISO</b>	Multi-Input and Single-Output
<b>NARMA</b>	Non-linear Auto-Regressive Moving Average
<b>MMC</b>	Modified Mountain Clustering
<b>GA</b>	Genetic Algorithm
<b>NF</b>	Neuro-Fuzzy
<b>S-RW</b>	Sigmoid-Recurrent Wavelet
<b>RS-W</b>	Recurrent Sigmoid-Wavelet

<b>FS-W</b>	Feedback to Sigmoid from Wavelet
<b>FW-S</b>	Feedback to Wavelet from Sigmoid
<b>RN</b>	Recurrent Neuron
<b>SS-RW</b>	Summation Sigmoid-Recurrent Wavelet
<b>SRS-W</b>	Summation Recurrent Sigmoid–Wavelet
<b>SFS-W</b>	Summation Feedback to Sigmoid from Wavelet
<b>SFW-S</b>	Summation Feedback to Wavelet from Sigmoid
<b>SRN</b>	Summation Recurrent Neuron
<b>MS-RW</b>	Multiplication Sigmoid-Recurrent Wavelet
<b>MRS-W</b>	Multiplication Recurrent Sigmoid–Wavelet
<b>MFS-W</b>	Multiplication Feedback to Sigmoid from Wavelet
<b>MFW-S</b>	Multiplication Feedback to Wavelet from Sigmoid
<b>MRN</b>	Multiplication Recurrent Neuron
<b>ISMR</b>	India Summer Monsoon Rainfall
<b>IMD</b>	India Meteorological Department
<b>GCM</b>	General Circulation Models
<b>RR</b>	Rescaled Range
<b>H</b>	Hurst
<b>IITM</b>	Indian Institute of Tropical Meteorology

# Chapter 1

## Literature Review

### 1-1 Introduction

Science has evolved from trying to understand and predict the behavior of the universe and systems within it. Much of this is based on finding suitable models, which agree with observations, and analyzing the results. These models can come in many different forms such as regression models, Artificial Neural Networks (ANN) and Fuzzy systems.

*Forecasting is a systematic effort to anticipate future events or conditions. The most well known type of forecast may be that of the meteorologist who prepares daily weather forecasts that help us decide how to dress each day and whether to take an umbrella when we leave for work in the morning. Other common forecasts are those that anticipate future economic conditions, traffic patterns, and even the size and number of classrooms that will be needed in local schools.*

In a prediction framework, the results of a statistical analysis based on data about the past are used to speculate about the future and to make decisions. In other way, forecasting and decision-making are very closely related. In a prediction context, researchers use data about the

past with the newest data about actual to speculate about the future and they encourage policy makers to act on that statistical vision of the future.

## **1-2 Identification**

Forecasting and identification have very close relationship with each other. Hence, better identification model results the high precision forecasting. Identification is a process through which one ascertains the identity of another person or entity.

Simulations (and models, too) are abstractions of reality. Often they deliberately emphasize one part of reality at the expense of other parts. Whereas models are mathematical, logical, or some other structured representation of reality, simulations are the specific application of models to arrive at some outcome.

In order to achieve the mission and goals, more industrial specific properties should be needed to enable the sharing and the reusing of semantics of models among different domains, territories or countries.

## **1-3 Soft Computing**

Soft computing refers to a collection of computational techniques in computer science, artificial intelligence, machine learning and some engineering disciplines, which attempt to study, model, and analyze very complex phenomena: those for which more conventional methods have not yielded low cost, analytic, and complete solutions. Earlier computational approaches could model and precisely analyze only relatively simple systems. More complex systems arising in biology, medicine, the humanities, management sciences, and similar fields often remained intractable to conventional mathematical and analytical methods. That said, it



should be pointed out that simplicity and complexity of systems are relative, and many conventional mathematical models have been both challenging and very productive. †

Unlike hard computing schemes, which strive for exactness and full truth, soft computing techniques exploit the given tolerance of imprecision, partial truth, and uncertainty for a particular problem. Another common contrast comes from the observation that inductive reasoning plays a larger role in soft computing than in hard computing.

In effect, the role model for soft computing is the human mind. The guiding principle of soft computing is: Exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness and low solution cost. The basic ideas underlying soft computing in its current incarnation have links to many earlier influences, among them Zadeh's 1965 paper on fuzzy sets; the 1973 paper on the analysis of complex systems and decision processes; and the 1979 report (1981 paper) on possibility theory and soft data analysis. The inclusion of neural computing and genetic computing in soft computing came at a later point.

### ***1-3.1 What is absorbed in Soft Computing?***

Now, the principal constituents of Soft Computing (SC) are Fuzzy Logic (FL), Neural Computing (NC), Evolutionary Computation (EC) Machine Learning (ML) and Probabilistic Reasoning (PR), with the latter subsuming belief networks, chaos theory and parts of learning theory. What is important to note is that soft computing is not a melange. Rather, it is a partnership in which each of the partners contributes a distinct methodology for addressing problems in its domain. In this perspective, the principal constituent methodologies in SC are complementary rather than competitive. Furthermore, soft computing may be viewed as a foundation component for the emerging field of conceptual intelligence.

### **1-3.2 Importance of Soft Computing**

The complementarities of FL, NC, GC, and PR have an important consequence: in many cases a problem can be solved most effectively by using FL, NC, GC and PR in combination rather than exclusively. A striking example of a particularly effective combination is what has come to be known as "neuro-fuzzy systems". Such systems are becoming increasingly visible as consumer products ranging from air conditioners and washing machines to photocopiers and camcorders. Less visible but perhaps even more important are neuro-fuzzy systems in industrial applications. What is particularly significant is that in both consumer products and industrial systems, the employment of soft computing techniques leads to systems, which have high MIQ (Machine Intelligence Quotient). In large measure, the high MIQ of SC-based systems account for the rapid growth in the number and variety of applications of soft computing.

In many ways, soft computing represents a significant paradigm shift in the aims of computing - a shift which reflects the fact that the human mind, unlike present day computers, possesses a remarkable ability to store and process information which is pervasively imprecise, uncertain and lacking in categoricity.

### **1-4 Wavelet**

Wavelet analysis is a new development in the area of applied mathematics. They were first introduced in seismology to provide a time dimension to seismic analysis that Fourier analysis lacked. Wavelet analysis allows researchers to isolate and manipulate specific types of patterns hidden in masses of data [Soman'05].

Wavelets are mathematical functions that cut up data into different frequency components, and then study each component with a resolution matched to its 'scale'. They have advantages over traditional Fourier methods in analyzing physical situations where the signal

contains discontinuities and sharp spikes. Wavelets were developed independently in the fields of mathematics, quantum physics, electrical engineering, and seismic geology. Historical perspective of wavelets is as follows:

**Historical Perspective:** In the history of mathematics, wavelet analysis shows many different origins [Meyer'93]. Much of the work was performed in the 1930's, and at that time, the separate efforts did not appear to be parts of a coherent theory.

**Pre-1930:** Before 1930, the main branch of mathematics leading to wavelet began with Joseph Fourier (1807) with his theories of frequency analysis, now often referred to as Fourier synthesis. He asserted that any  $2\pi$  - periodic function  $f(t)$  is the sum of its Fourier series.

$$a_0 + \sum_{k=1}^{\infty} (a_k \cos kt + b_k \sin kt) \quad (1.1)$$

The coefficients  $a_0$ ,  $a_k$  and  $b_k$  are calculated by

$$a_0 = \frac{1}{2\pi} \int_0^{2\pi} f(t)dt, \quad a_k = \frac{1}{\pi} \int_0^{2\pi} f(t)dt, \quad b_k = \frac{1}{\pi} \int_0^{2\pi} f(t)dt \quad (1.2)$$

Fourier's assertion played an essential role in the evolution of the ideas mathematicians had about the functions. He opened up the door to a new functional universe.

After 1807, by exploring the meaning of functions, Fourier series convergence, and orthogonal systems, mathematicians gradually were led from their previous notion of frequency analysis to the notion of scale analysis. That is, analyzing  $f(x)$  by creating mathematical structures that vary in scale. How? Construct a function, shift it by some amount, and change its scale. Apply that structure in approximating a signal. Now repeat the procedure. Take that basic structure, shift it, and scale it again. Apply it to the same

signal to get a new approximation and so on. It turns out that this sort of scale analysis is less sensitive to noise because it measures the average fluctuations of the signal at different scales.

### ***Wavelet multi resolution analysis***

***The 1930s:*** In the 1930s, several groups, working independently, researched the representation of the functions using scale-varying basis functions. By using scale varying basis function, called the Haar basis function, Paul Levy a 1930s physicist, investigated Brownian motion, a type of random signal [Meyer'93]. He found that the Haar basis function is superior to the Fourier basis functions for studying small-complicated details in the Brownian motion.

Another 1930s research effort by Littlewood, Paley, and Stein involved computing the energy of the function  $f(x)$ :

$$energy = \frac{1}{2} \int_0^{2\pi} f(t)^2 dt \quad (1.3)$$

The computation produced different results if the energy was concentrated around a few points or distributed over a larger interval. This result disturbed the scientists because it indicated that energy might not be conserved. The researchers discovered a function that can vary in scale and can conserve energy when computing the functional energy. Their work provided Devid Marr with an effective algorithm for numerical image processing using wavelets in the early 1980s.

***1960-1980s:*** During these years a lot of work has been done. Some of the pioneering works done by Coifman and Morlet are given below:

- ***Guido Weiss and Ronal R. Coifman (1960-1980):*** These two mathematicians studied the simplest element of a function space, called atoms, with the goal of finding the atoms for a common function and finding the “assembly rules” that allows the reconstruction of all elements of the function space using these atoms.
- ***Grossman and Morlet (1980):*** A physicist and an engineer, broadly defined wavelets in context of quantum physics. These two researchers provided a way of thinking for wavelets based on physical intuition.

***1980-1990s:*** in these years, the pioneering work of the Stephane Mallat (1985) on pyramidal algorithm or multi-resolution theory gave the new apex in wavelet era.

- ***Stephane Mallat (1985):*** In 1985, Stephane Mallat gave wavelets an additional jump-start through his work in digital signal processing. He discovered some relationship between quadrature mirror filters, pyramidal algorithms, and orthonormal wavelet bases. Y. Meyer constructed the first non-trivial wavelets. Ingrid Daubechies used Mallat’s work to construct a set of wavelet orthonormal basis functions that are perhaps the most elegant, and have become the corner stone of wavelet applications today.

***Post-1990s:*** During this decade application of wavelets, develop in many branch of science, same as signal processing, identifications, numerical analysis and networks.

## **1-5 Motivation**

During the nineteen century Fourier transform, solved many problems in physics and engineering. This prominence led scientists and engineers to think of them as the preferred way to analyze phenomena of all kinds. This ubiquity forced a close examination of the method. As a result, through the twentieth century, mathematicians, physicists, and engineers came to realize a

drawback of the Fourier transforms: they have trouble reproducing transient signals or signals with abrupt changes, such as the spoken word or the rap of a snare drum [Soman'05].

At the present scenario, wavelet decomposition emerges as a new powerful tool for function approximation due to its multi-resolution property. Recent advances have shown the existence of orthonormal wavelet bases, from which follows the availability of rates of convergence for approximation by wavelet based networks.

Several works has been done and so many works are going on for wavelets. Its application in neural network and neuro-fuzzy model gives tremendous performance for function approximation. However, until this time, selection of parameters and support of wavelet properties are mystery. Due to these discrepancies and multi resolution property of the wavelets, we have motivated to work with wavelet for forecasting and modeling applications of dynamic systems.

## **1-6 Scope of the Thesis**

In recent years, wavelets have become a very active subject in many scientific and engineering research areas. Especially, Wavelet Neural Networks (WNN), inspired by both the feed forward neural networks and wavelet decompositions, have received considerable attention [Q. Zhang'92, Q. Zhang'97, J. Zhang'95] and become a popular tool for function approximation. The main characteristic of WNN is that wavelet functions are used as the nonlinear transformation function in the hidden layer, instead of the usual sigmoid function. Incorporating the time-frequency localization properties of wavelets and the learning ability of the Neural Network (NN), WNN has shown its advantages over the regular methods such as NN for complex nonlinear system modeling.

At present, there are two different kinds of WNN structure. One is with fixed wavelet bases, where the dilation and translation parameters of wavelet basis are fixed, and only the output layer weights are adjustable. Another type is the variable wavelet bases, where the dilation parameters, translation parameters and the output layer weights are adjustable [Billings'05]. For the WNN with fixed wavelets, the main problem is the selection of wavelet bases/frames. The wavelet bases have to be selected appropriately since the choice of the wavelet basis can be critical to approximation performance. Obviously, to improve the approximation accuracy, a large number of wavelet neurons are required for WNN with fixed wavelet bases. This will result in a large complex network structure and cause over-fitting problem.

Since the dilation parameter has explicit physical concept, i.e., resolution, it plays a significant role in wavelet analysis and approximation of a given function. In this thesis, for selection of the wavelet bases/frames, we have used variable wavelet bases for the better accuracy of function approximation though its complexity is increased. In addition, we have presented a comparative study for different types of the wavelet functions. To used approximation of inputs by Sigmoid Activation Function (SAF) and Wavelet Activation Functions (WAF), separately, we have proposed two neuron models to combine them.

In dealing with the modeling of dynamic systems recurrent network have better performance as compared to static behavior of feed-forward network based on proposed sigmoid-wavelet neuron models different types of recurrent neuron models are introduced. These recurrent neurons give us opportunity of comparative study of recurrent neuron models consist of SAF and WAF in feed-forward neural network architecture.

In many complex and ill-defined systems especially with the uncertainty of the systems, the fuzzy models have shown high performance. Motivated by both the theory of multi-

resolution analysis of WNN and the traditional Neuro-fuzzy model, Wavelet Neuro- Fuzzy (WNF) model is introduced. The goal of introducing the WNN in the fuzzy model is improving function approximation accuracy in terms of the dilation and translation parameters of wavelets, meanwhile not increasing the number of wavelet bases. In general, the Takagi-Sugeno-Kang (TSK) fuzzy models consist of a set of rules, and the consequent of each rule acts like a “local model” by using fuzzy set to partition the input space into local fuzzy regions. The consequents of these rules are represented by either a constant or a linear equation. In this work, the consequent part of each fuzzy rule corresponds to sub-WNNs at different resolution levels and used to capture different behaviors (global or local) of the approximated function. Here, the role of the fuzzy set is to determine the region for the contribution of the sub-WNNs to the output of the WNF. As a result, wavelets with different dilation values under these fuzzy rules are fully utilized to capture various essential components of the system.

In addition, in this work, the series-parallel and parallel configurations, which are used in parameter identification of networks\models, are exploited simultaneously for learning the parameters of the premise and the consequent part of the neuro-fuzzy model.

## **1-7 Organization of Thesis**

The organization of the thesis is as follows:

### **Chapter 2: Wavelet Networks**

A comparative study of two existing wavelet networks namely Wavelet Synapses Neural Network (WSNN) and Wavelet Activation Neural Network (WANN), based on three different



wavelet function is presented in this chapter. Feed-forward neural networks show the ability to deal with complex problems and especially in input-output data systems. In addition, wavelet transformation has the ability of representing a function and revealing the properties of the function in the localized regions of the joint time frequency space. The chapter covers some basic concept of wavelet same as wavelet transform, Continuous Wavelet Transform (CWT) and Discrete Wavelet Transform (DWT). Three types of non-orthogonal wavelet are introduced in this section. These wavelets when used in feed-forward network give wavelet network.

### **Chapter 3: Generalized Wavelet Networks**

The main objective of this thesis is to improve existing one layer feed-forward network with SAF and WAF. Feed-forward neural networks show the ability to deal with complex problems and especially in input-output data systems. In addition, wavelet transformation has the ability of representing a function and revealing the properties of the function in the localized regions of the joint time frequency space. Due to above ability, in this chapter, combination of sigmoid and wavelet activation function is proposed. It has shown that a smart combination of these not only decreases the size of the network, it also increases the accuracy of the network. Two proposed wavelet neural network namely Summation Sigmoid-Wavelet (SS-W) and Multiplication Sigmoid-Wavelet (MS-W) neuron model are discussed in details. One method for structure identification of the model is introduced. General approximation capability of the network has also been presented in this chapter with different theorems.

#### **Chapter 4: Neuro-Fuzzy Model**

This chapter serves as an introduction into the basic concept of parameter identification for neuro-fuzzy models. Two parameter identification schemes, namely Parallel (P) and the Series-Parallel (S-P) configurations, are described in this chapter. A combination of these two configurations is proposed for neuro-fuzzy models. Modified mountain clustering is applied to neuro-fuzzy models for structure determination and initialization of the neuro-fuzzy models. An algorithm with adaptive learning rate is used to learn learning parameters of the model. Convergence of the learning procedure is guaranteed by Lyapunov stability theorem.

#### **Chapter 5: Wavelet Neuro-Fuzzy Model**

This chapter discusses about the wavelet neuro-fuzzy model. The proposed network in chapter 3 with better performance is used in the consequent part of each fuzzy rule in TSK neuro-fuzzy model that results WNF model. A hybrid of Genetic Algorithm and Gradient Descent has been employed to learn the model parameters.

#### **Chapter 6: Recurrent Wavelet Networks**

In this chapter, recurrent neuron models are introduced. Due to the dynamic behavior of recurrent networks, they are suitable in dealing with the modeling of dynamic systems as compared to static behavior of feed-forward network. The quantitative behavior of the sigmoid and wavelet activation functions for dealing with and saving the dynamic of systems are considered. The general approximation properties of the recurrent neuron models are also evaluated. Since the convergence analysis plays an important role in the recurrent networks, the Lyapunov stability approach is employed to guarantee the convergence of network.

## **Chapter 7: Case study, Indian Monsoon Rain-Fall**

The agricultural economy of India is closely linked to the performance of summer monsoon rainfall all over India. The ability to understand and predict circulation and rainfall during the Asian summer monsoon on various time-scales is of prime importance to the economy of several nations of this region because of its affect on agriculture, drinking water, transportation, health, power, and the very livelihood of billions people living in the monsoon region. Due to these reason, in this chapter, all the proposed networks are tested on rainfall data.

## **Chapter 8: Conclusion**

Finally, conclusions of the thesis and suggestions for the future work have been covered in chapter 8.

# **1-8 Description of Some Dynamic Systems**

Six different classes of dynamic systems are described in the following examples for validation of the proposed work. Among them the selected four dynamic examples are different nonlinear differential equations with different order [Narendra'90]. Example five is a general benchmark problem of gas furnace data [Box'70], whereas, example six is an action performed by the operator at chemical plant [Sugeno'93].

### **Example 1: Linear regression with nonlinear input**

The system is a non-linear second order dynamical model [Narendra'90]. The function ' $f$ ' is a polynomial of current input  $u(k)$  of degree three whereas the input  $u(k)$  is a sum of two sinusoids given in (1.6).

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + f[u(k)] \quad (1.4)$$

Where

$$f[u(k)] = [u(k)]^3 + 0.3[u(k)]^2 - 0.4u(k) \quad (1.5)$$

$$u(k) = \sin(2\pi k/250) + \sin(2\pi k/25) \quad (1.6)$$

In this example, 500 input-output data are generated. First three hundred data are used for learning procedure and remaining 200 data are for prediction.

### **Example 2: Non-linear regression with random input**

This system expressed as second order nonlinear function that is presented by (1.7). The input  $u(k)$  is a random variable uniformly distributed in the interval  $[-1, 1]$ . Five hundred input-output data are generated by the second order difference equation [Narendra'90]: three hundred data are used to train the model and remaining two hundred data are used for validation of the model.

$$y(k+1) = f[y(k), y(k-1)] + u(k) \quad (1.7)$$

where function  $f$  is:

$$f[y(k), y(k-1)] = \frac{y(k)y(k-1)[y(k)+2.5]}{1+y^2(k)+y^2(k-1)} \quad (1.8)$$

### **Example 3: Non-Linear Regression with Non-Linear Input**

A system described by difference equation [Narendra'90] and expressed as (1.9).

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \quad (1.9)$$

$$u(k) = \sin(2\pi k/25) + \sin(2\pi k/10) \quad (1.10)$$

This system is having first order nonlinear dynamic. Hundred input-output data are produced by input  $u(k)$  as given in (1.10). Eighty data is used for training and 20 remaining data are used for testing and validation.

#### **Example 4: Non-linear Regression of Input and output**

In this example, a nonlinear plant with third delay in output and with two delays in inputs has been taken from [Narendra'90, Lee'00] and describes as:

$$y(k+1) = f(y(k), y(k-1), y(k-2), u(k), u(k-1)) \quad (1.11)$$

Where  $f$  is:

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 \cdot x_2 \cdot x_3 \cdot x_5 (x_3 - 1) + x_4}{1 + x_3^2 + x_2^2} \quad (1.11)$$

The reference [Narendra'90] has used five input to predict next output but [Lee'00] used only  $u(k)$  and  $y(k)$  to predict next output  $y(k+1)$ . Here we also used these two variables to predict the output  $y(k+1)$ . One thousand input-output data are produced by using the input expressed by (1.12) to identify the models. The input  $u(k)$  is selected same as equation (1.12) for data 1 to 1000.

**THESIS**

$$u(k) = \begin{cases} \sin\left(\frac{\pi k}{25}\right), & k < 250 \\ 1.0 & 250 \leq k < 500 \\ -1.0 & 500 \leq k < 750 \\ 0.3 \sin\left(\frac{\pi k}{25}\right) + 0.1 \sin\left(\frac{\pi k}{32}\right) \\ + 0.6 \sin\left(\frac{\pi k}{10}\right), & 750 \leq k < 1000 \end{cases} \quad (1.12)$$

### Example 5: Gas Furnace data

A benchmark problem of system identification is considered [Box'70]. The process in this example is a gas furnace with single input  $u(t)$ , i.e., gas flow rate and single output  $y(t)$ , i.e.,  $CO_2$  concentration. Here we supposed there are three inputs:  $y(t-1)$ ,  $u(t-3)$  and  $u(t-4)$  to the model [Sugeno'93]. Total 290 data are utilized which can be found in [Box'70]. First 250 data are used to train the models and remaining 40 data are used for testing and validation of the model.

### Example 6: Human Operation at a Chemical Plant

We deal with a model of an operator's control action of a chemical plant [Sugeno'93]. The plant is for producing a polymer by the polymerization of some monomers. Since the start-up of the plant is very complicated, a man has to make the manual operation at the plant. As shown in Fig. 1.1 there are five input candidates  $(u_1, u_2, \dots, u_5)$  whom a human operator might refer at the start up of chemical plant to take control action  $y$ , for production of polymer.

- $u_1$  : monomer concentration.
- $u_2$  : change in monomer concentration.
- $u_3$  : monomer flow rate.
- $u_4, u_5$  : local temperatures inside the plant.
- $y$  : set point for monomer flow rate.

Here  $u_1$  and  $u_3$  are employed to model the control action [Azeen'03]. Out of 70 data points of the above six variable from the actual plant, first 60 data are used for training the model and remaining 10 data are used for prediction.

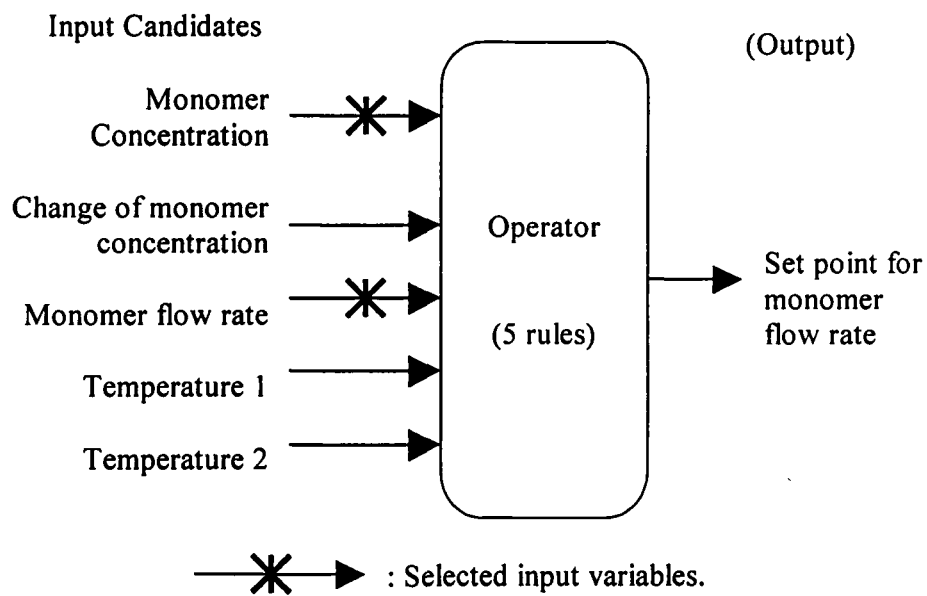


Fig. 1.1. Control action of an operator

# Chapter 2

## Wavelet Networks

### 2-1 Introduction

The approximation of general continuous functions by nonlinear networks such as those discussed in [Zhang'92, Poggio'90a,b] is very useful for system modeling and identification. Such approximation methods can be used for example in black-box identification of non-linear systems. Feed Forward Neural Networks (FFNN) has been established as a general approximator for fitting nonlinear models from input-output data [Hornik'89, Funahashi'89, Hartman'90, and Blum'91].

In addition, wavelet transformation has the ability of representing a function and revealing the properties of the function in the localized regions of the joint time frequency space. The wavelet with coarse resolution can capture the global (low frequency) behavior easily, while the wavelet with fine resolution can capture the local behavior (higher frequency) of the function accurately. These distinguished characteristics give wavelet based neural networks with the advantages of fast convergence, easy training, and high accuracy [Ho'01]. In view of the similarity between wavelet transformation and feed-forward neural networks, the idea of augmenting both, Zhang and Beneviste [Zhang'92] have proposed Wavelet Neural Networks



(WNN). WNN instigate a superior system model for complex and seismic application in comparison to Neural Network (NN) with sigmoid activation function. The application of wavelet is mostly limited to small dimension [Benveniste'94], though WNN can handle large dimension problem [Zhang'97].

Basis function networks have been investigated by many researchers, employing various kinds of basis function. e.g. hyper basis function [Poggio'90a], splines [Poggio'90, Friedman'91], polynomial [Sanger'90, Sanger'91] and radial basis function [Poggio'90b, Moody'89].

Due to the above advantages of wavelets over other basis functions, Boubez and Peskin [Boubez'93] used wavelet functions as basis functions. They adopted orthonormal sets of wavelets and verified that network weights can be computed directly and independently. Yamakawa [Yamakawa'94] has proposed to adopt the over-complete system of non-orthogonal smooth wavelet bases in order to approximate a nonlinear function with a smooth function. He proposed two types of neuron models and used a simple cosine function as a compactly supported wavelet function. Later some authors used the Yamakawa's models with non-orthogonal wavelet functions like Mexican [Zhang '95, Ho'01]. But none of the reported work caters a comparative study for different types of the wavelets. In this chapter, two types of wavelet network architectures introduced by Yamakawa [Yamakawa'94], namely, Wavelet Synapses Neural Network (WSNN) and Wavelet Activation function Neural Network (WANN) are described. Different wavelet activation functions are applied and the networks are tested by eight different dynamic examples.

This chapter is organized as follow: In section 2-2, a brief discussion of artificial neural network is presented. In section 2-3, wavelet and wavelet transform are discussed. Section 2-4

proposes Wavelet Neural Network (WNN) models and describes their convergence analysis. The learning algorithm based on Gradient Descent describes in section 2-5. Experimental results are revealed in section 2-6 and, finally conclusions are relegated to section 2-7.

## 2-2 Artificial Neural Network

*Artificial Neural Network (ANN) was introduced in the middle of the last century to reproduce learning and generalization of the human brain [Bernieri'94, Patterson'96, Schalkoff'97]. Ability of ANN to deal with complex problems, generalization of the result from known situation to unforeseen situation and ability to carry out classifications of the elements of a given set make them one of the most useful technique in functional approximation, nonlinear system identification and control, pattern recognition and classification, and optimization.*

The architecture of the single hidden-layer neural network is given in the Fig. 2.1. The operation performed by each layer is also described below:

1. *Input Layer:* Each node of this layer passes the input signal with its connection weights to the hidden layer neurons of the network.
2. *Hidden Layer:* Hidden layer perform two operations: in the first operation, all the signals coming from the input nodes multiplied with their connection weights and summed up; and in the second operation these summed quantities passes through the activation (logistic sigmoid) function that maps the signal and gives the output in between the range  $[-1, 1]$  or  $[0, 1]$ .
3. *Output Layer:* In this layer, the output coming from the hidden layer multiplied with its connection weights and finally the linear summation of all the signal, which gives the output of the network, takes place.

The mathematical expression for the output of the neural network is given in (2.1).

$$\hat{Y} = \sum_{j=1}^M \sum_{i=1}^n \Phi_j(C_{ji} \cdot X_i + b_j) \cdot W_j \quad (2.1)$$

Where  $b_j$ , for  $j=1,2,\dots,M$ , denotes the bias for each hidden neuron,  $C_{ji}$ ; for  $j=1,2,\dots,M$  and  $i=1,2,\dots,n$  denotes the weight to signal  $X_i$  going to the  $j^{\text{th}}$  neuron.  $W_j$  for  $j=1,2,\dots,L$  is the weight to the output of  $j^{\text{th}}$  neuron for output of the network,  $\Phi_j$  is the activation function for the  $j^{\text{th}}$  neuron.

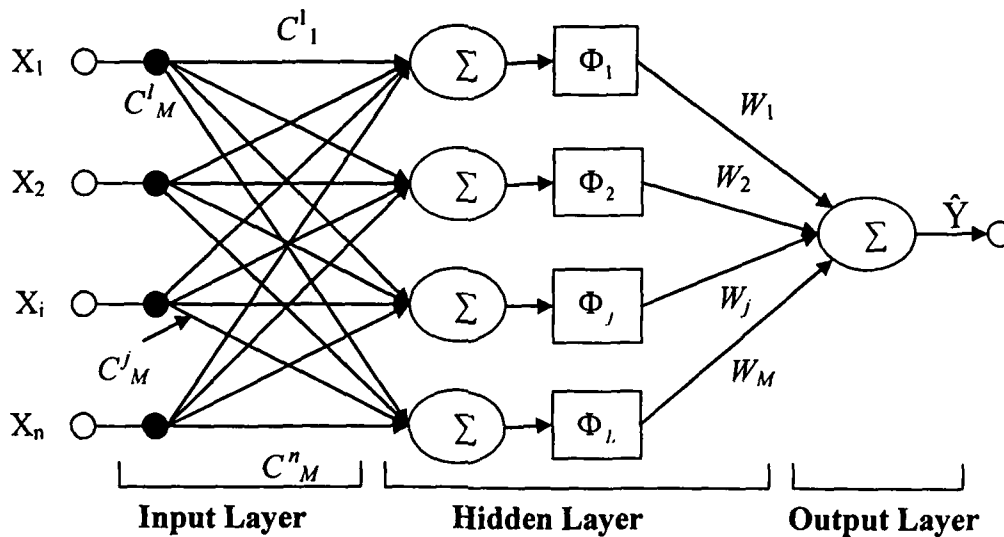


Fig. 2.1. Architecture of the single layer neural network

## 2-3 Wavelet

A wave is usually defined as an oscillating function of time or space, such as a sinusoid. A wavelet is a “small wave”, which has its energy concentrated in time to give a tool for the analysis of transient, nonstationary, or time varying phenomena. The function  $\psi(t)$  is a wavelet or mother wavelet if it satisfies these two properties:

$$\int_{-\infty}^{+\infty} \psi(t) dx = 0 \quad (2.2)$$

$$\int_{-\infty}^{+\infty} |\psi(t)|^2 dx < \infty \quad (2.3)$$

First property is suggestive of a function i.e. oscillatory or that which has a wavy appearance and second property implies that most of the energy in it is confined to a finite duration. We will take wavelet and use them in a series expansion of signals or functions in the same way as Fourier series uses the wave or sinusoid to represent a signal or function.

### **2-3.1 Continuous time Wavelet Transform (CWT)**

The Wavelet Transform (WT) in its continuous form provides a flexible time-frequency window, which narrows when observing high frequency phenomena and widens when analyzing low frequency behavior. Thus, time resolution becomes arbitrarily good at high frequencies, while the frequency resolution becomes arbitrarily good at low frequencies. This kind of analysis is suitable for signals composed of high frequency components with short duration and low frequency components with long duration, which is often the case in practical situations. Here, a brief review from the theory of wavelets is described that gives basic idea about the wavelets and the related work. Wavelets are divided in the two parts: Continuous Wavelet Transform (CWT) and Discrete Wavelet Transform (DWT) [Rao'04, Daubechies'92, Burrus'97, Stark'05, Soman'05, Cgui'95]. Historically the CWT was the first studied wavelet transform:

Let  $f(t)$  be any square integratable function. The CWT of  $f(t)$  with respect to a wavelet  $\psi(t)$  is defined as:

$$W(a, b) = \int f(t) \psi_{a,b}^*(t) dt \quad (2.4)$$

Where

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-b}{a}\right) \quad (2.5)$$

$\psi(t)$  is the mother wavelet, 'a' is a scaling factor, 'b' is shifting parameter and \* denotes complex conjugation. The family of functions can be obtained by scaling and shifting of  $\psi(t)$ . Thus, the wavelet transform is a function of two variables. Both  $f(t)$  and  $\psi(t)$  belong to  $L^2(\mathfrak{R})$ , the set of square integrable function, also called the set of energy signals.

The signal or function  $f(t)$  can be expressed as (2.6).

$$f(t) = \iint W(a, b) \cdot \psi\left(\frac{t-b}{a}\right) db da \quad (2.6)$$

The mother wavelet has the property that the set  $[\psi_{a,b}(t)]_{a,b \in Z}$  forms an orthogonal basis in  $L^2(\mathfrak{R})$ . This implies that the mother wavelet can, in turn, generate any function in  $L^2(\mathfrak{R})$ . The mother wavelet has to satisfy the following admissibility condition:

$$C_{\psi} = \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{\omega} d\omega < \infty \quad (2.7)$$

Where  $\Psi(\omega)$  is the Fourier transform of  $\psi(t)$ .

In practice  $\Psi(\omega)$  will have sufficient decay, so that the admissibility condition is reduced to:

$$\int_{-\infty}^{\infty} \psi(t) dt = \Psi(0) = 0 \quad (2.8)$$

### 2-3.2 Discrete time Wavelet Transform (DWT)

The CWT has the drawbacks of redundancy and impracticability with digital computers. As parameters  $(a, b)$  are of continuous values, the resulting CWT is a very redundant representation, and impracticable as well. This impracticability is the result of redundancy. Therefore, the scale and shift parameters are evaluated on a discrete grid of time-scale leading to a discrete set of continuous basis functions. The continuous inverse wavelet transform (2.6) is discretized as:

$$f(t) = \sum_i W_i \cdot a_i^{-1/2} \psi\left(\frac{t - b_i}{a_i}\right) \quad (2.9)$$

To analyze discrete time signals, it is convenient to take integer values for 'a' and 'b' in defining this basis: if  $a = 2^j$  and  $b = n \cdot 2^j$  (where  $j$  and  $n$  are integers) then, via translations and dilations:

$$\{\psi_{j,n}(t)\}_{j,n \in \mathbb{Z}} = \left\{ 2^{-j/2} \psi\left(\frac{t - n \cdot 2^j}{2^j}\right) \right\} \quad (2.10)$$

Equation (2.10) forms a sparse orthonormal basis of  $L^2(\mathfrak{R})$ . This means that the wavelet basis induces an orthogonal decomposition of any function in  $L^2(\mathfrak{R})$ .

### 2-3.3 Types of the wavelet

The difference between wave (sinusoids) and wavelet is shown in Fig. 2.2. Waves are smooth, predictable and everlasting, whereas wavelets are of limited duration, irregular and may be asymmetric. Waves are used as deterministic basis functions in Fourier analysis for the expansion of the functions (signals), which are time invariant, or stationary.

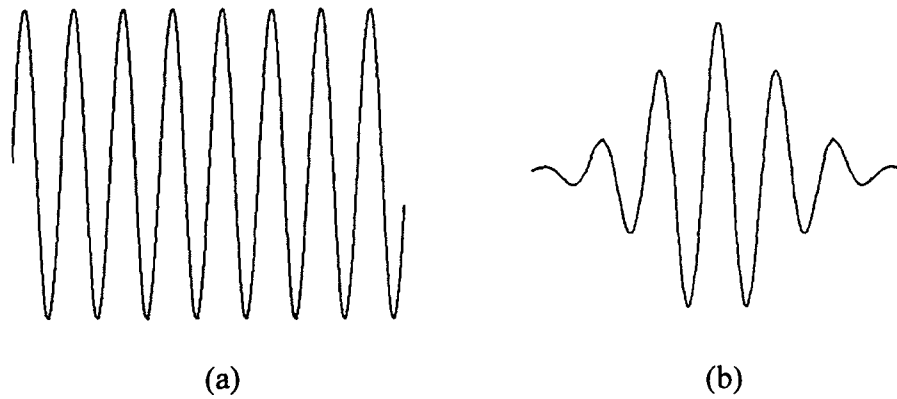


Fig. 2.2. Representation of (a) a wave and (b) a wavelet

The important characteristic of wavelets is that they can serve as deterministic or non-deterministic basis for generation and analysis for the most natural signals to provide better time frequency representation, which is not possible with waves using conventional Fourier analysis. The selection of basic (or mother) wavelet depends very much on the nature of the signals and the goal of the signal processing. These basis vectors have the following important properties.

- **Vanishing moments:** The higher the degrees of vanishing moments a basis has, the better it models the smooth part of the signal.
- **Regularity:** This property is important in signal compression if high ratios are desired; the shapes of the basis vectors become “visible” under these circumstances. The larger the regularity, the smoother the basis vector becomes. Low regularity might result in fractal- like shapes in the reconstructed signals or images.
- **Compact Support:** This property is important for efficient and exact numerical implementation [Daubechies92].

Some wavelets are better than others are for specific applications. In general however, because of these properties, wavelet bases generate very efficient and simple representations for

piecewise smooth signals and images. The manner in which vanishing moments, regularity and compact support affect the wavelet's efficacy as a basis for signal classification is not clear. One would expect that a wavelet that "looks like" the elemental components of the signals under consideration would be most appropriate. More important however, is the ability of the wavelet basis to generate a Time-Frequency Representation (TFR) that clearly distinguishes signals in different classes. This requires that the wavelet functions appropriately model the signal, and that they be localized and well behaved in the time-frequency plane.

In this thesis, three types of wavelet function, namely Mexican hat, Morlet and Sinc are introduced.

#### **a) Mexican hat wavelet**

This wavelet is derived from a function, which is proportional to the second derivative function of the Gaussian probability density function. It is non-orthogonal, with infinite support and has maximum energy around origin with the narrow band. The expression for Mexican hat wavelet is given by (2.11) and it is shown in Fig. 2.3.

$$\psi(x) = (1 - 2x^2) \cdot \exp(-x^2) \quad (2.11)$$

#### **b) Morlet wavelet**

This wavelet is derived from a function that is proportional to the cosine function and Gaussian probability density function. It is non-orthogonal, infinite support and its maximum energy lies around origin with the narrow band. The Morlet wavelet is expressed as (2.12) and shown in Fig. 2.4.

$$\psi(x) = \exp(-x^2) \cdot \cos(5x) \quad (2.12)$$



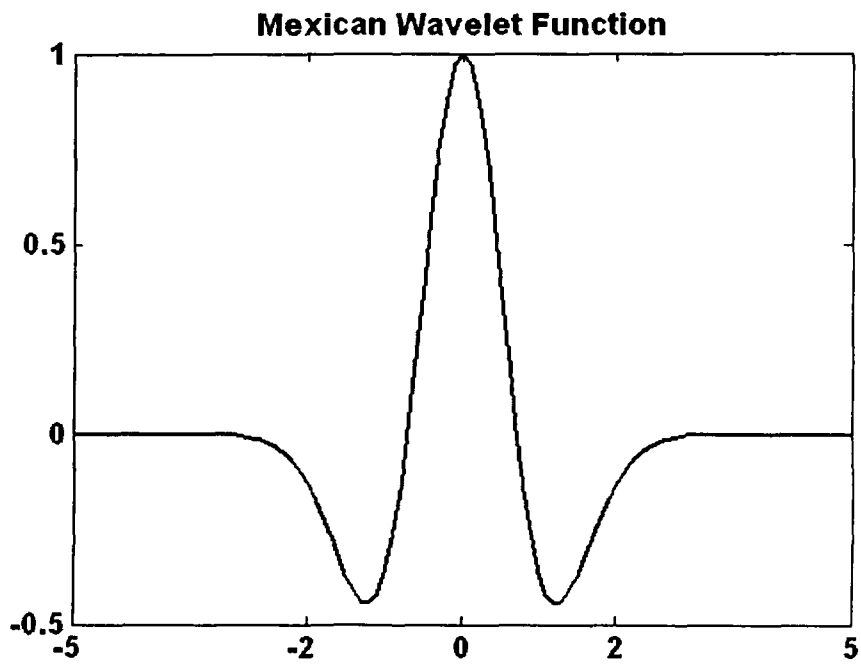


Fig. 2.3. Mexican hat wavelet functions

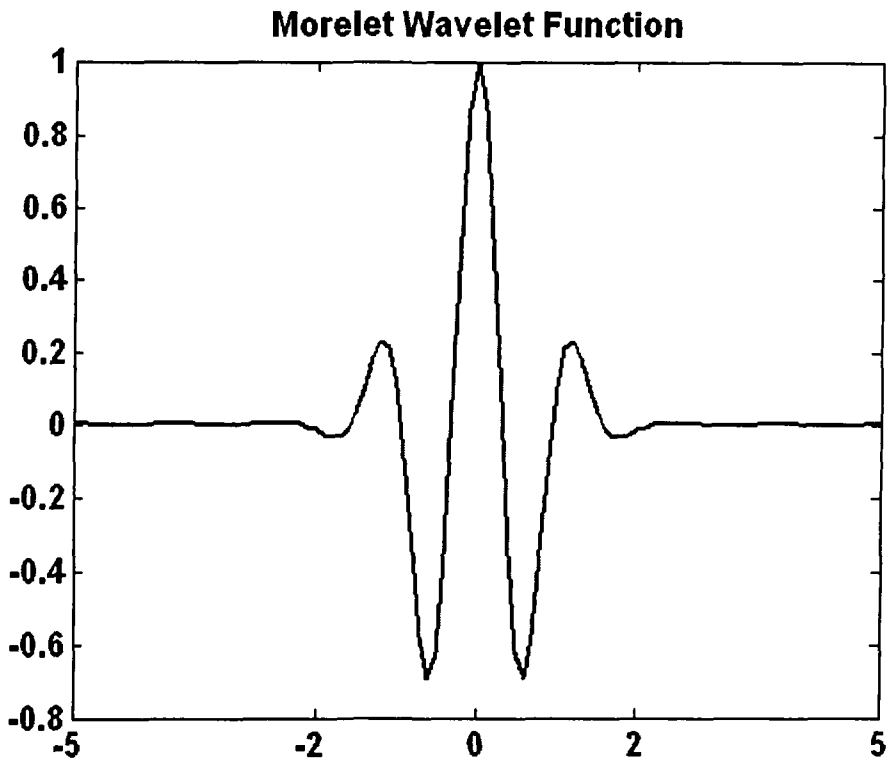


Fig. 2.4. Morlet wavelet functions

### c) Sinc (Shannon) wavelet

This wavelet is derived from a function that is proportional to the cosine function. This wavelet is also non-orthogonal with infinite support and maximum energy occupies wider band around origin as compared to the above two wavelets. The Sinc wavelet is specified as (2.13) and shown in Fig. 2.5.

$$\psi(x) = \sin(\pi x) / (\pi x) \quad (2.13)$$

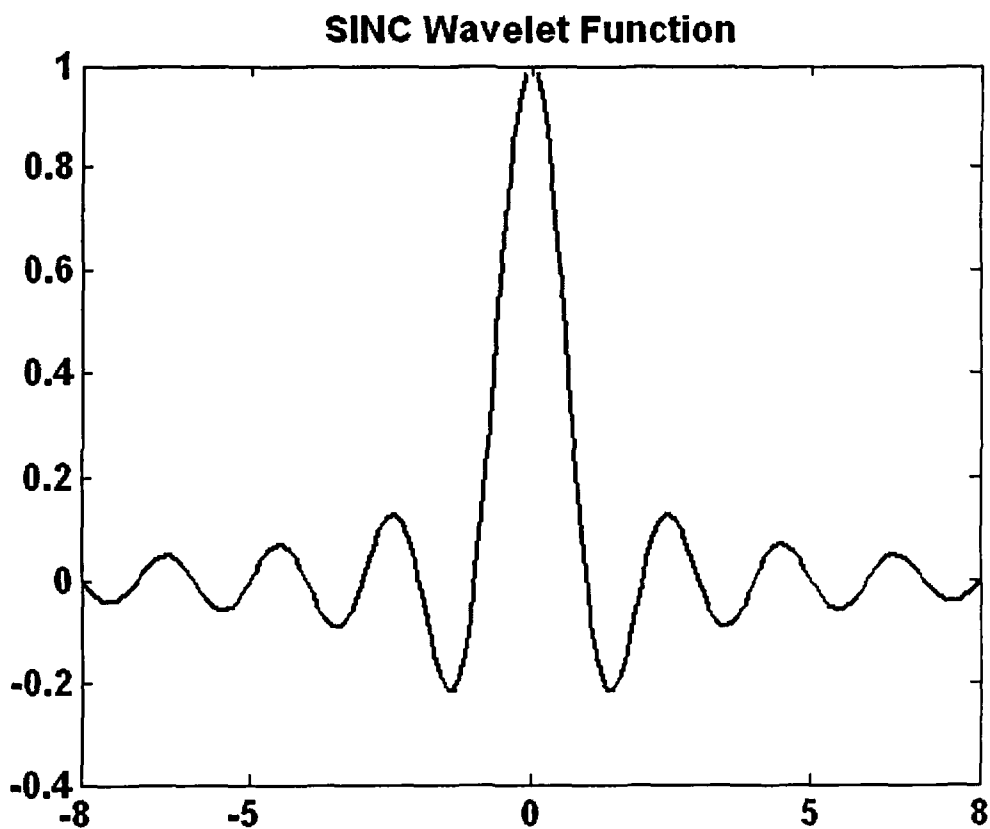


Fig. 2.5. Sinc wavelet functions

## 2-4 Wavelet Neural Network

The applications of orthonormal wavelet bases and wavelet frames are usually limited to problems of small dimension [Zhang'97]. The main reason is that they are composed of regularly dilated and translated wavelet. For practical implementations, infinite basis and frames are always truncated. The number of wavelets in a truncated bases or frames drastically increases with the dimension, therefore, constructing and storing wavelet bases or frames of large dimension are with prohibitive cost.

In most practical situation of large dimension, the available data are sparse [Zhang'97]. If the inverse wavelet transform is discretized according to the distribution of the data, there are expectations to reduce the number of wavelets needed in the reconstruction. It is thus possible to handle problem of large dimension with such adaptive discretization of the inverse wavelet transform.

The adaptive discretization consists of determining the parameters  $w_i$ ,  $a$ , and  $b$  in (2.6) according to data sample  $(x, y)$ . This problem is very similar to neural network training. In fact, formula (2.6) can be viewed as a one hidden layer of neural network with  $\psi$  as the activation function of the hidden neuron and with a linear neuron in the output layer. For this reason, we refer to the adaptively discretized inverse wavelet transform as wavelet network.

A basic McCulloch and Pitts neuron model is characterized by weighted sum (linear sum) of inputs and a sigmoid activation function. Two wavelet neuron models were proposed in [Yamakawa'94] by modifying the basic neuron model. We have used wavelet synapses and wavelet activation function neuron model of Yamakawa in this chapter, which is shown in the Figure 2.6.

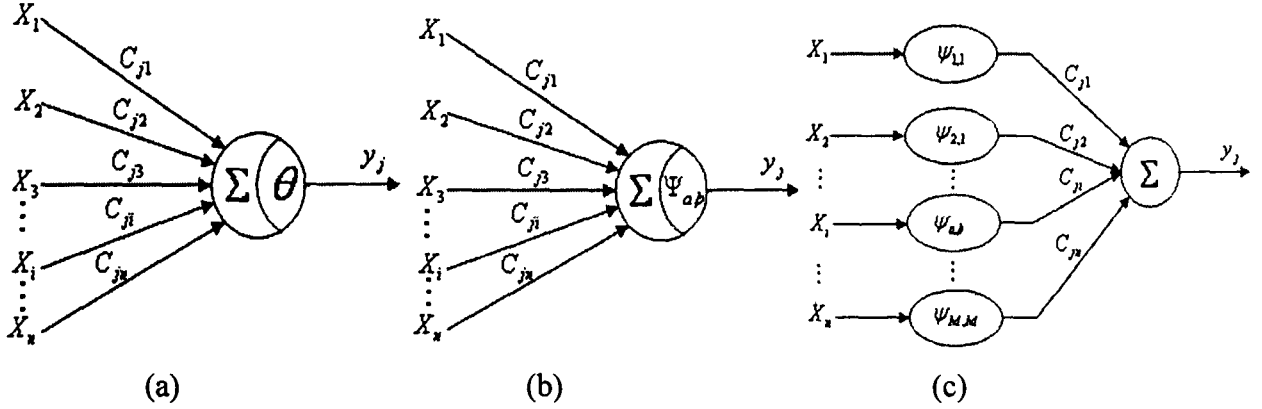


Fig. 2.6: (a) Simple neuron model (b) Wavelet activation function neuron model and (c) Wavelet synapses neuron model

### 2-4.1 Wavelet Synapses Neural Network

The architecture of WSNN is shown in Fig. 2.7. Suppose  $M$  is the total number of wavelet functions selected. If  $\Psi_{a,b}$  is used as nonlinear transformation function of  $M$  number of hidden units and  $C_{a,b}$  is the connection weights, then layer wise analysis of the architecture is given below:

- Layer 1 (*Input layer*): In this layer, each input is directly applied to every wavelet function  $\Psi_{a,b}$ .
- Layer 2 (*Hidden layer*): This layer performs two operations. Firstly, the output of input layer multiplied with connection weights  $C_{a,b}^i$ , and then linear summation takes place in second operation (2.14).

$$y_i = \sum_{a=1}^M \sum_{b=1}^{a-1} C_{a,b}^i \Psi_{a,b}(x_i) \quad (2.14)$$

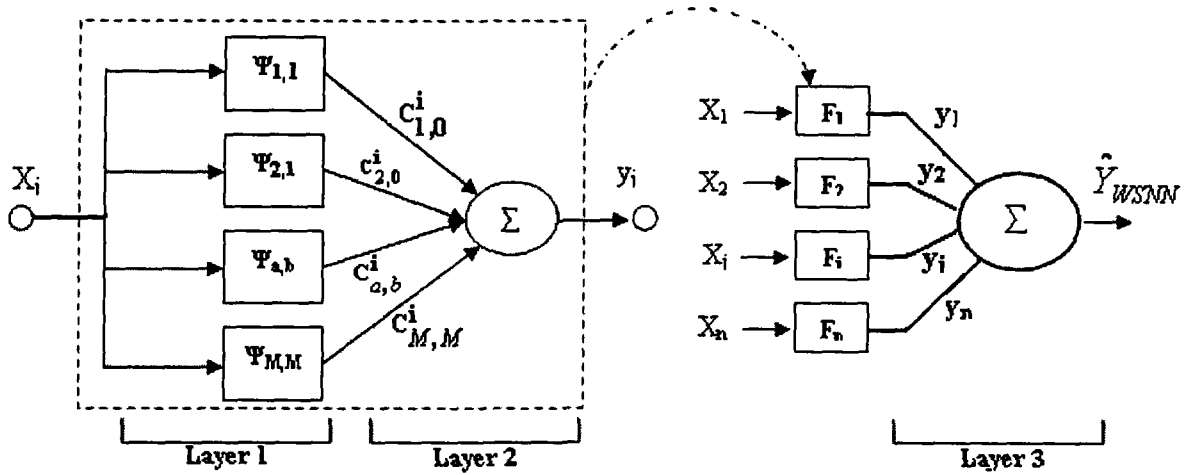


Fig. 2.7. Wavelet Synapses Neural Network (WSNN)

- **Layer 3 (Output layer):** In this layer, the outputs of hidden layer are linearly summed that gives the output of the model. The output of WSNN network can be calculated by:

$$\hat{Y}_{WSNN} = \sum_{i=1}^n y_i \quad (2.15)$$

Where  $i = 1, 2, \dots, n$ ; denotes the number of inputs applied to the network.

For a WSNN network with scaling factor  $M$ , there is  $M \cdot (M + 1)/2$  learning parameter  $C$  for each input. Therefore, the total number of learning parameters is  $n \cdot M \cdot (M + 1)/2$ , where  $n$  is the number of inputs.

### 2-4.2 Wavelet Activation function Neural Network

In this network, wavelet functions are used as activation function. The architecture of WANN is shown in Fig. 2.8. Layer wise description of the network architecture has been given below:

- Layer 1 (*Input layer*): In this layer, all the inputs are multiplied with their connection weights  $C_{a,b}^i$ .
- Layer 2 (*Hidden layer*): Two operations are performed in this layer. First, output of input layer are summed linearly and applied to each wavelet function  $\psi_{a,b}$  in second operation as given in (2.16).

$$y_{a,b} = \psi_{a,b} \left( \sum_{i=1}^n C_{a,b}^i x_i \right) \quad (2.16)$$

- Layer 3 (*Output layer*): In this layer, also two operations are performed. The outputs of hidden layer multiplied by their connection weights  $W_{a,b}$  in first operation, and then linear summation takes place in second that gives the output of WANN. The mathematical expression for the output of WANN is given in (2.17).

$$\hat{Y}_{WANN} = \sum_{a=1}^M \sum_{b=1}^{a-1} W_{a,b} \cdot y_{a,b} \quad (2.17)$$

For a WANN network with scaling factor  $M$ , there is  $M \cdot (M+1)/2$  hidden neuron. Therefore, the total number of learning parameters  $C$  in Layer 1 is equal to  $n \cdot M \cdot (M+1)/2$ , where  $n$  is the number of inputs. The number of learning parameters  $W$  in Layer 3 is also equal to  $M \cdot (M+1)/2$ . Therefore, the total number of learning parameter in this model is equal to  $(n+1) \cdot M \cdot (M+1)/2$ .

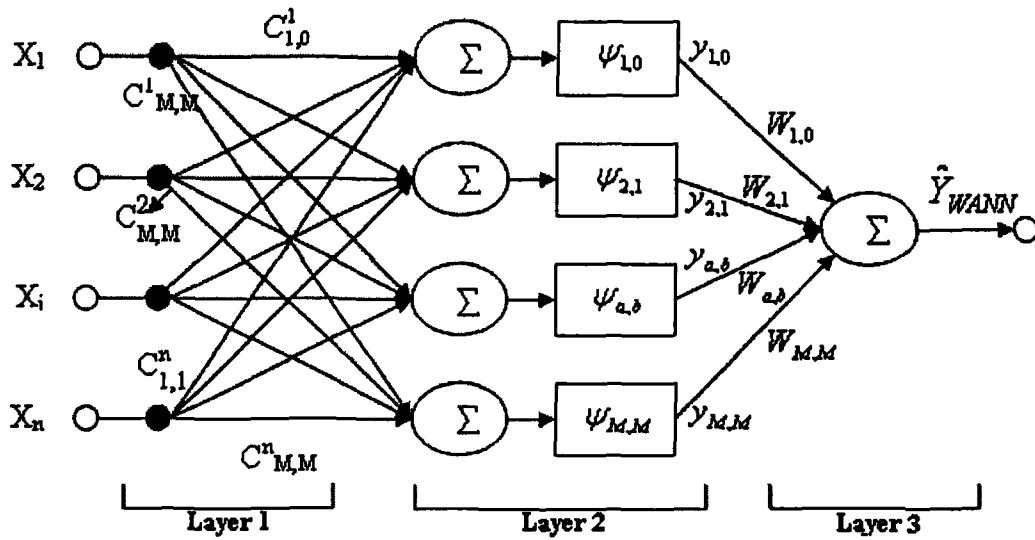


Fig. 2.8. Wavelet Activation Function Neural Network (WANN)

## 2-5 Gradient Descent learning of parameters

The Gradient Descent (GD) learning can be achieved by minimizing the performance index  $J$  as follows:

$$J = \frac{1}{2 \cdot P \cdot y_r^2} \cdot \sum_{p=1}^P (Y(p) - \hat{Y}(p))^2 \quad (2.18)$$

where  $y_r = \left( \max_{p=1}^P Y(p) - \min_{p=1}^P Y(p) \right)$ ,  $\hat{Y}$  is output of network and  $Y$  is actual data,  $P$  is the number of dataset. The reason for using normalized mean square error is that it provides a universal platform for model evaluation irrespective of application and target value specification while selecting an input to the model [Azeem'00a].

In the batch-learning scheme employing  $P$ -data set, change in any parameter is covered by the equation

$$\Delta v(q) = \sum_{p=1}^P \Delta_p v(q) + \alpha_m \cdot \Delta v(q-1) - \gamma_f \cdot v(q) \quad (2.19)$$

and the parametric update equation is;

$$v(q+1) = v(q) + \Delta v(q) \quad (2.20)$$

where  $\Delta v(q) = -\partial J / \partial v$  and  $v$  may stand for any of the parameters  $C'_{a,b}$  or  $W_{a,b}$ , and  $C'_{a,b}$  in WSNN or WANN networks, respectively.  $q$  is  $q^{\text{th}}$  epoch,  $\alpha_m$  is a momentum update coefficient in the limits  $0 \leq \alpha_m < 1$  (typically  $\alpha_m = 0.9$ ),  $\gamma_f$  is a decay factor (typically in the range of  $10^{-3}$  to  $10^{-6}$ ).

We apply gradient descent technique to modify the parameter  $C'_{a,b}$  in WSNN. The parameter update formula for  $p^{\text{th}}$  data set is as follows:

$$\Delta_p C'_{a,b}(q) = -\eta \frac{\partial J}{\partial C'_{a,b}} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial C'_{a,b}} \right|_p \quad (2.21)$$

where  $\eta$  is adaptive learning rate. By applying gradient descent technique to modify the parameters  $W_{a,b}$  and  $C'_{a,b}$  in WANN, the parameter update formulas for  $p^{\text{th}}$  data set are derived as follows:

$$\Delta_p C'_{a,b}(q) = -\eta \frac{\partial J}{\partial C'_{a,b}} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial C'_{a,b}} \right|_p \quad (2.22)$$

$$\Delta_p W_{a,b}(q) = -\eta \frac{\partial J}{\partial W_{a,b}} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial W_{a,b}} \right|_p \quad (2.23)$$

where  $e = Y - \hat{Y}$  is the error between the actual output and the model output. By applying chain method to the above equation  $\frac{\partial \hat{Y}}{\partial C'_{a,b}}$  or  $\frac{\partial \hat{Y}}{\partial W_{a,b}}$  and  $\frac{\partial \hat{Y}}{\partial C'_{a,b}}$  for WSNN or WANN networks are derived as follows:



a) For WSNN

$$\frac{\partial \hat{Y}(k)}{\partial C_{a,b}^i} = \psi_{a,b}(x_i) \quad (2.24)$$

b) For WANN

$$\frac{\partial \hat{Y}}{\partial W_{a,b}} = y_{a,b} = \psi_{a,b} \left( \sum_{i=1}^n C_{a,b}^i \cdot x_i \right) \quad (2.25)$$

$$\frac{\partial \hat{Y}}{\partial C_{a,b}^i} = x_i \cdot W_{a,b} \cdot \psi'_{a,b} \left( \sum_{i=1}^n C_{a,b}^i \cdot x_i \right) \quad (2.26)$$

WAF  $\psi$  for different wavelet function is given in (2.11-2.13).  $\psi'$  in (2.26) is differential functions for the wavelet functions. Derivative for Mexican hat, Morlet and Sinc function are given in (2.27-2.29), respectively.

$$\psi'_{a,b}(z) = -\frac{2}{a^{1.5}} \cdot z \cdot e^{-z^2} \cdot [2 + (1 - 2 \cdot z^2)] \quad (2.27)$$

$$\psi'_{a,b}(z) = \frac{-2}{a} \cdot z \cdot e^{-z^2} \cdot \cos(5 \cdot z) - e^{-z^2} \cdot \frac{5}{a} \cdot \sin(5z) \quad (2.28)$$

$$\psi'_{a,b}(z) = \frac{1}{a} \cdot \frac{\pi \cdot z \cdot \cos(\pi \cdot z) - \sin(\pi \cdot z)}{\pi \cdot z^2} \quad (2.29)$$

Where  $z = \frac{\sum_{i=1}^n C_{a,b}^i \cdot x_i - b}{a}$ .

## 2-6 Simulation Results

In this section, different types of dynamic systems that are discussed in chapter 1 have been considered. WSNN and WANN networks are tested for three different types of wavelet functions, namely, Mexican hat wavelet, Sinc wavelet and Morlet wavelet function. A comparative study of performance index  $J$ , for WSNN and WANN networks with different wavelet function and scaling factor  $M$ , for all examples, is shown in Table 2.1.

### Revisited Example 1: *Linear regression with nonlinear input*

In this example, WSNN with Mexican hat wavelet function and WANN with Morlet wavelet function, yield better performance and WSNN network with Mexican hat is the best with performance index  $J=1.1576 \times 10^{-6}$ . However, WSNN with Sinc wavelet function dose not converge at all. The learning parameter  $C$  of this network is as follows. The number of column in learning parameter  $C$  shows the number of inputs and the number of rows shows the number of hidden neurons in WSNN network. Figure 2.9 shows actual and network output of WSNN network with Mexican hat wavelet function. The error also is shown in this figure.

$$C = \begin{bmatrix} -0.0036 & -0.2941 & 0.4620 \\ -0.1986 & 0.1560 & -0.1829 \\ 0.7855 & 0.0446 & 0.3546 \\ 0.2880 & -0.1814 & 0.3043 \\ 0.2875 & -0.2797 & -0.1977 \\ 0.5777 & 0.4240 & -0.0771 \\ 0.1967 & 0.4679 & -0.4133 \\ 0.3233 & -0.2516 & 0.0074 \\ 0.5139 & 0.3298 & 0.2538 \\ 0.2669 & 0.2097 & 0.0594 \\ -0.1192 & 0.1209 & 0.3820 \\ 0.2816 & -0.1469 & 0.3937 \\ 0.6003 & 0.3829 & 0.6085 \\ 0.0883 & -0.0911 & 0.6319 \\ 0.4213 & 0.0687 & 0.6646 \end{bmatrix}$$

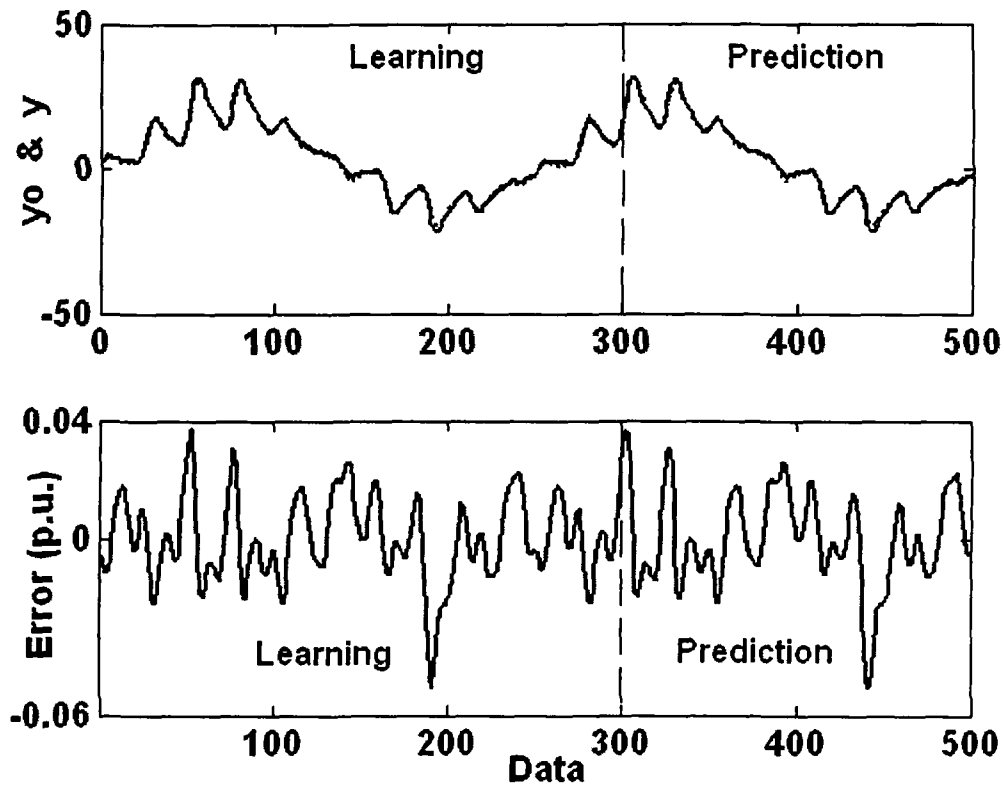


Fig. 2.9. Actual output and network output with WSNN (Mexican hat) network and the error for Example 1

### Revisited Example 2: *Non-linear regression with random input*

In this example, WSNN and WANN networks with Morlet function have better performance and WANN with Morlet function is the best model with performance index  $J=1.147 \times 10^{-5}$ . The learning parameters  $C$  &  $W$  of this network are as follows. The number of rows in  $C$  and the size of  $W$  show the number of hidden neuron and the number of column in  $C$  shows the number of inputs in WANN network. WSNN network with Sinc wavelet function, for this example, does not converge. Figure 2.10 shows actual output and network output of WANN network with Morlet function and the error.

$$C = \begin{bmatrix} 0.3201 & 0.2279 & 0.3832 \\ 0.6301 & 0.6295 & 0.5978 \\ 0.8352 & 0.5727 & 0.0735 \\ 0.0139 & 0.7949 & 1.0723 \\ 0.2697 & 0.4789 & 0.1077 \\ 0.6200 & 0.0414 & 0.1698 \\ 0.5601 & 0.3121 & 0.8287 \\ 0.1206 & -0.3583 & 1.4104 \\ 0.8266 & 1.0795 & 0.6907 \\ 0.3806 & 0.4967 & 0.3247 \\ 0.8865 & 0.3628 & 0.8807 \\ 0.6964 & -0.0012 & 0.8033 \\ 0.5681 & 0.0387 & 0.0777 \\ 0.9238 & 0.1711 & 0.1124 \\ 0.1527 & 0.8065 & 0.9189 \end{bmatrix} \quad W = \begin{bmatrix} 0.1787 \\ 0.0826 \\ 0.1689 \\ 0.6186 \\ 0.4415 \\ 0.4237 \\ 0.0696 \\ 0.8107 \\ 0.5939 \\ 0.8456 \\ 0.5364 \\ 0.9113 \\ 0.7424 \\ 0.5382 \\ 0.9793 \end{bmatrix}$$

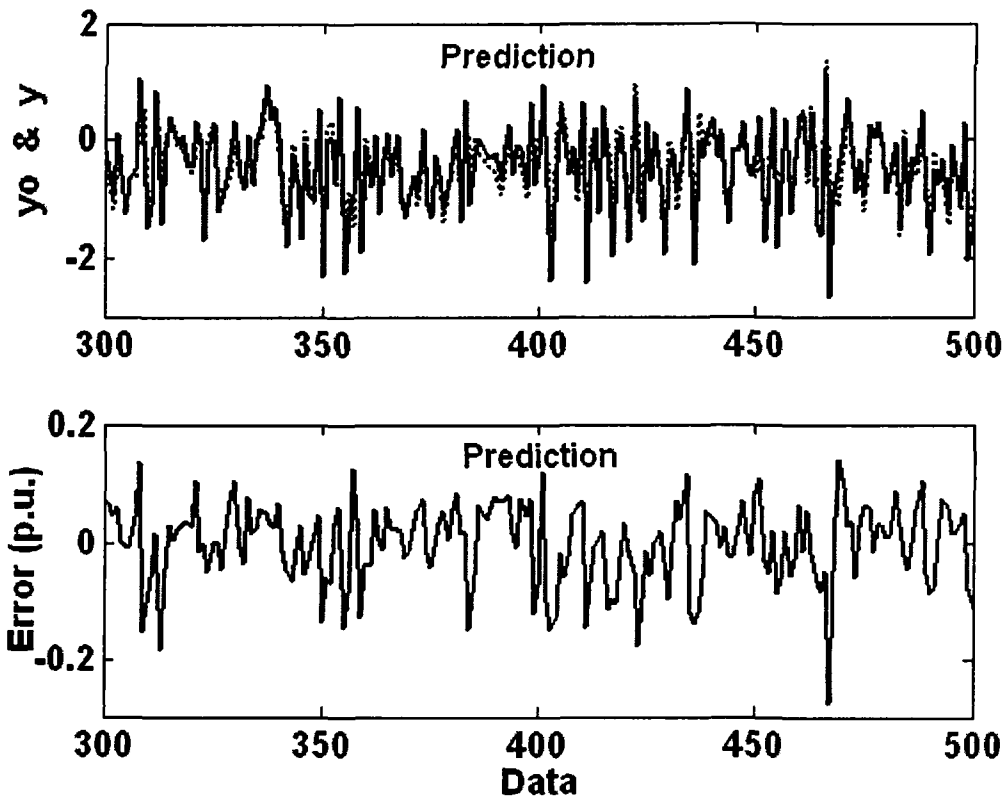


Fig. 2.10. Actual output and network output with WANN (Mexican hat) network and the error for Example 2

### Revisited Example 3: *Non-Linear Regression with Non-Linear Input*

In this example, WSNN with Morlet and WANN network with Mexican hat wavelets have better performance while WANN network with Mexican hat and performance index  $J=1.436 \times 10^{-4}$  is the best. WSNN with Sinc wavelet function does not convergence. Actual output and the output of the WANN network with Mexican hat wavelet function and the error are shown in Fig. 2.11. The learning parameters  $C$  &  $W$  for this network are as follows.

$$C = \begin{bmatrix} 1.0575 & -0.2304 \\ 0.6299 & 0.6032 \\ 0.7022 & 0.5383 \\ 0.9972 & 0.6570 \\ 0.3040 & 0.8453 \\ 0.1794 & -0.0317 \\ 0.9202 & 0.6102 \\ 0.2894 & 0.4761 \\ 0.3859 & 0.6415 \\ 0.6504 & 0.6816 \\ 0.3331 & 0.7914 \\ 0.2020 & 0.5176 \\ 0.2421 & 0.3185 \\ 0.2264 & 0.0371 \\ 0.0812 & 0.2415 \end{bmatrix} \quad W = \begin{bmatrix} 0.5725 \\ -0.0644 \\ 0.5778 \\ 0.4333 \\ 0.3929 \\ 0.4022 \\ -0.3507 \\ 0.5410 \\ 0.3716 \\ 1.0131 \\ 0.5824 \\ 0.1109 \\ 0.2274 \\ 0.3756 \\ 0.5056 \end{bmatrix}$$

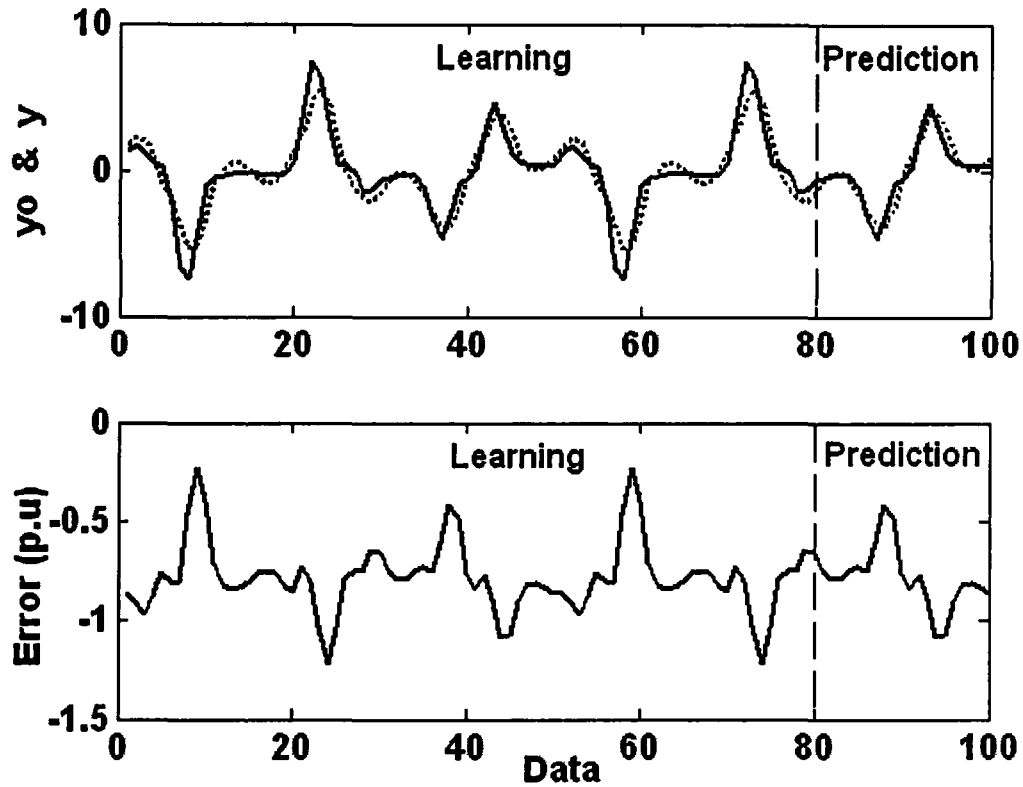


Fig. 2.11. Actual output and network output with WANN (Mexican hat) network and the error for Example 3

#### Revisited Example 4: *Non-linear Regression of Input and output*

In this example, Morlet and Mexican hat wavelet give better learning pattern for WSNN and WANN models, respectively. However, WANN with Mexican hat wavelet function yields best performance index with  $J=1.229 \times 10^{-5}$ . WSNN with Sinc function does not converge. Actual output and the output of the WANN network with Mexican hat wavelet function and the error are shown in Fig. 2.12. The learning parameters  $C$  &  $W$  of this network are as follows.

$$C = \begin{bmatrix} 0.2983 & 0.9195 \\ 0.8244 & 0.8071 \\ 0.7990 & 0.5669 \\ 0.5296 & 0.5529 \\ 0.5257 & 0.1979 \\ 0.6817 & 0.5770 \\ 0.0682 & 0.3274 \\ 0.6209 & 0.9769 \\ -0.0034 & 0.6931 \\ 0.3570 & 0.3717 \\ 0.3079 & 0.7465 \\ 0.8802 & 0.2760 \\ -0.0225 & 0.4158 \\ 0.7148 & 0.9028 \\ 0.9337 & 0.6596 \end{bmatrix} \quad W = \begin{bmatrix} 0.0620 \\ 0.3012 \\ -0.0032 \\ 0.3721 \\ 0.3190 \\ 0.4128 \\ -0.0985 \\ -0.0528 \\ 0.3878 \\ 0.4114 \\ 0.0006 \\ 0.3154 \\ 0.3611 \\ 0.5669 \\ 0.3047 \end{bmatrix}$$

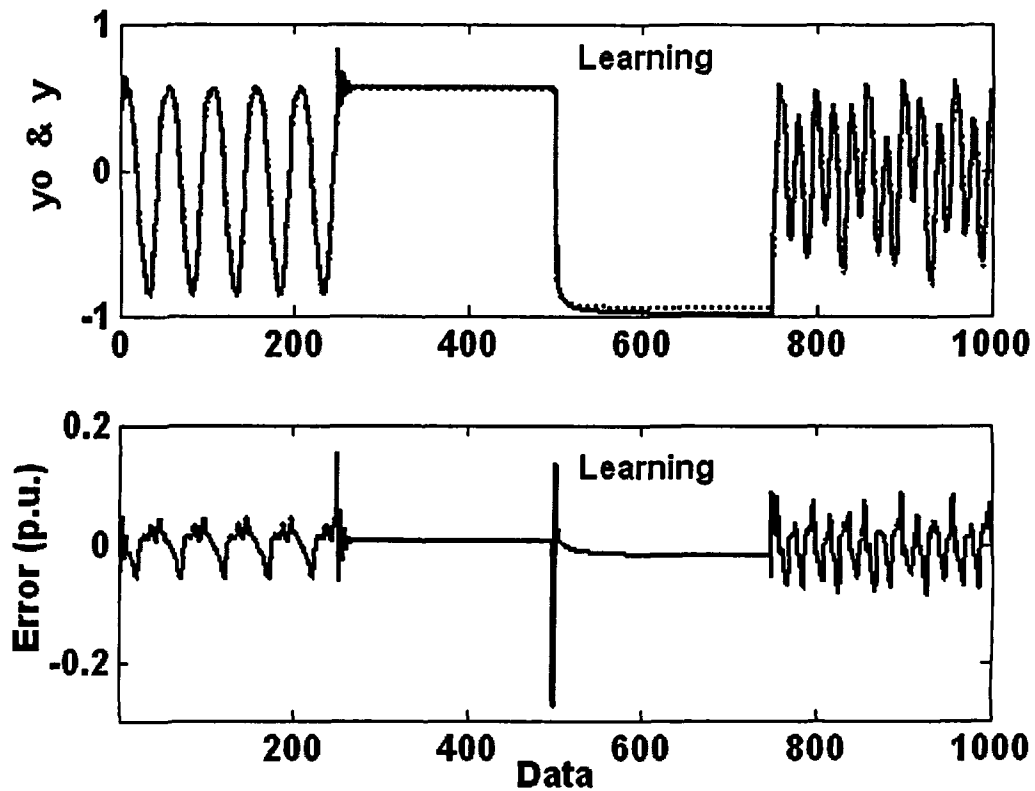


Fig. 2.12. Actual output and network output with WANN (Mexican hat) network and the error for Example 4

### Revisited Example 5: Gas Furnace Data

In this example, Morlet and Mexican hat wavelet give better learning pattern for WSNN and WANN models, respectively. However, Mexican hat in WANN is the best with performance index  $J=2.632 \times 10^{-7}$  while WSNN learning pattern for Sinc does not converge. Figure 2.13 shows the actual output and WANN network with Mexican hat wavelet output. The error is presented in this figure. The learning parameters  $C$  &  $W$  of this network are as follows.

$$C = \begin{bmatrix} 0.7684 & 0.5781 & 1.0060 \\ 0.7315 & 1.0590 & 0.5302 \\ 0.8833 & 0.0239 & 0.6560 \\ 0.2034 & 0.7845 & 0.8445 \\ 0.4194 & 0.0268 & 0.1347 \\ 0.2129 & 0.5195 & 0.0224 \\ 0.0350 & 0.1921 & 0.2246 \\ 0.0811 & 0.7156 & 0.1165 \\ 0.8505 & 0.2506 & 0.0693 \\ 0.3402 & 0.9338 & 0.8529 \\ 0.4661 & 0.1371 & 0.1802 \\ 0.9137 & 0.5216 & 0.0324 \\ 0.2285 & 0.8952 & 0.7339 \\ 0.8620 & 0.9423 & 0.5365 \\ 0.6566 & 0.3350 & 0.2760 \end{bmatrix} \quad W = \begin{bmatrix} 0.6476 \\ -0.5161 \\ 0.2141 \\ 0.1751 \\ 0.4582 \\ 0.7032 \\ 0.5793 \\ 0.5092 \\ 0.0742 \\ 0.1932 \\ 0.3796 \\ 0.2764 \\ 0.7708 \\ 0.3139 \\ 0.6381 \end{bmatrix}$$



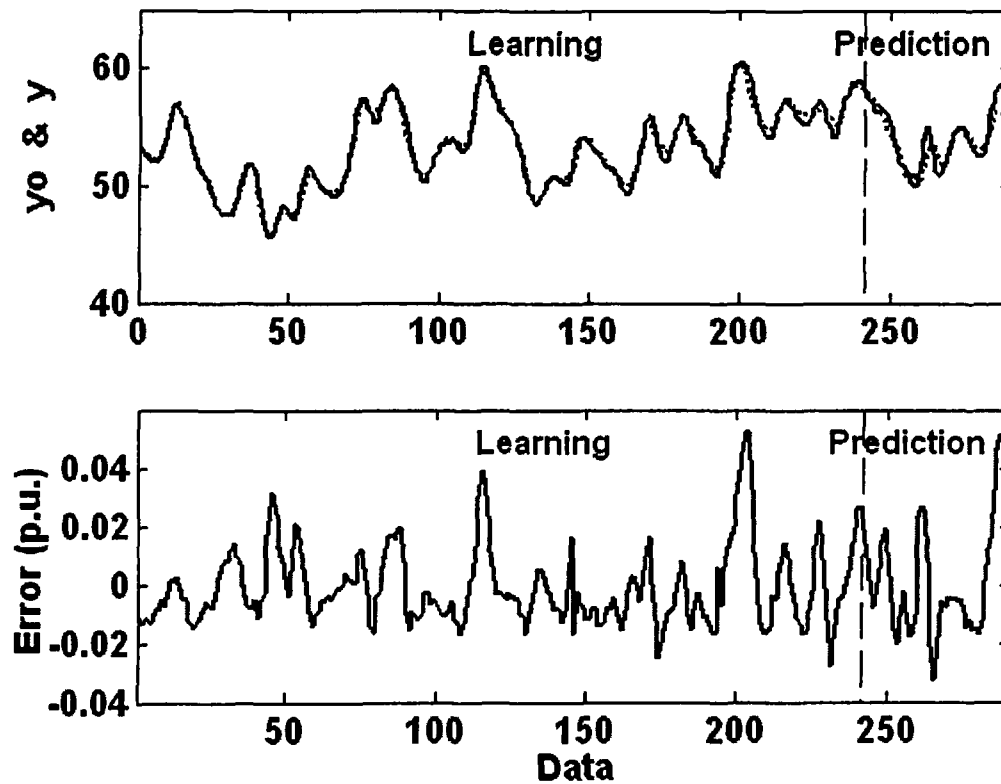


Fig. 2.13. Actual output and network output with WANN (Mexican hat) network and the error for Example 5

### Revisited Example 6: Human Operation at a Chemical Plant

In this example, WSNN and WANN networks with Morlet function yield better performance, while WANN with Morlet wavelet function is best with performance index  $J=8.228 \times 10^{-6}$ . WSNN network for Sinc wavelet function does not convergence. Actual output and the output of the WANN with Morlet function and error are shown in Fig. 2.14. The learning parameters  $C$  &  $W$  of this network are as follows.

$C =$	0.4057	0.0235	$W =$	0.8845
	0.9355	0.6898		0.0877
	0.9166	0.5639		0.6509
	0.4103	0.8558		0.3865
	0.8934	0.5141		0.8861
	0.0579	0.4043		0.8016
	0.3529	0.8283		0.3885
	0.8131	0.5315		-0.0089
	0.0097	0.1842		0.8592
	0.1389	0.6628		0.4738
	0.2028	0.8312		0.5520
	0.1986	0.0096		0.7564
	0.6036	0.6847		0.9346
	0.2721	0.3739		0.7825
	0.1988	0.8302		0.1992

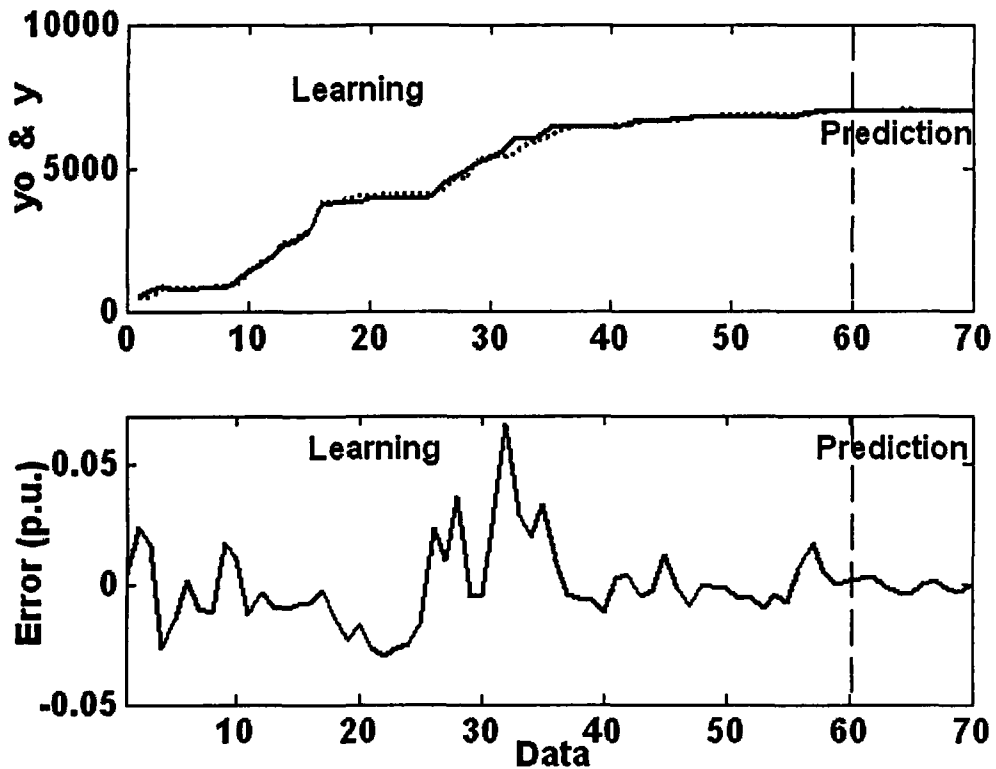


Fig. 2.14. Actual output and network output with WANN (Mexican hat) network and the error for Example 6

### Example 7: Human Operation at a Chemical Plant

A test signal is generated by the following dynamical system [Yamakawa'94]:

$$X_{n+1} = \frac{5X_n}{1+X_n^2} - 0.5X_n - 0.5X_{n-1} + 0.5X_{n-2} \quad (2.30)$$

with initial values of  $X_0 = 0.2$ ,  $X_1 = 0.3$  and  $X_2 = 1.0$ . This signal is chaotic and difficult to predict. A time series of 120 data are produced. First 100 data are used for training of the model and remaining 20 data are used for prediction.

In this example, WSNN and WANN networks with Morlet and Mexican hat wavelet function, respectively, yield better performance. WANN network with Mexican hat is best with  $J=3.4516 \times 10^{-5}$ , while WSNN network with Sinc wavelet function does not convergence. Figure 2.15 shows system output and the output of the WANN network with Mexican hat function. The error also is shown in Fig. 2.15. The learning parameters  $C$  &  $W$  of this network are as follows.

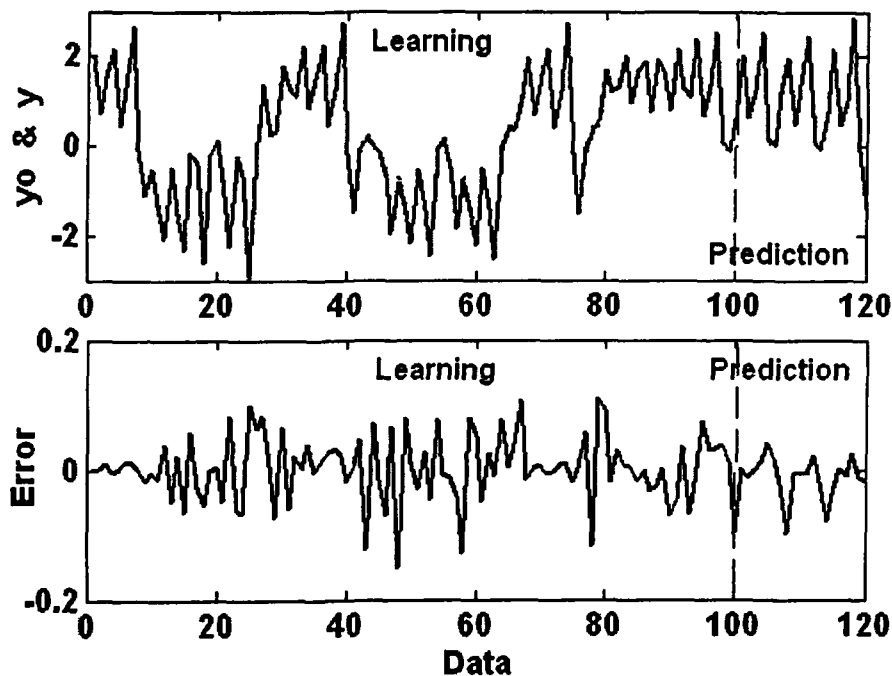


Fig. 2.15. Actual output and network output with WANN (Mexican hat) network and the error for Example 7

$$C = \begin{bmatrix} -0.3596 & 0.9389 & -0.0108 \\ 0.6207 & 0.6815 & 0.9756 \\ 0.2815 & 0.6812 & -0.4890 \\ -0.0350 & 0.0981 & 2.7240 \\ 1.0787 & 0.0695 & 0.2231 \\ 0.3849 & 1.2978 & -1.1089 \\ 0.9713 & 1.2888 & 0.2822 \\ -0.1619 & 0.5269 & 0.4099 \\ 0.4246 & 0.0813 & 0.6710 \\ -0.0163 & -0.1119 & 1.3133 \\ 0.7904 & -0.3032 & 1.3394 \\ 0.7639 & 0.3584 & 0.1869 \\ -0.9237 & 0.9736 & 0.3086 \\ -0.0934 & 0.0283 & 3.1322 \\ 0.4308 & 0.8169 & -0.0579 \\ 0.9412 & 0.3422 & 0.4453 \\ -0.1112 & 0.8273 & 0.6658 \\ 0.1214 & 0.5225 & 0.7317 \\ 0.9567 & -0.5422 & 1.0359 \\ 0.9259 & -0.1694 & 0.7421 \\ 0.9002 & 0.4819 & 0.6735 \\ 0.8594 & 0.5305 & -0.1662 \\ 0.6908 & 0.9054 & 0.7609 \\ 0.6603 & 0.4248 & 1.9629 \\ 0.2899 & 0.5815 & 0.4761 \\ 0.2742 & 0.6083 & 1.0177 \\ 0.6395 & 0.7989 & 0.2813 \\ 0.2006 & 0.1383 & 0.3538 \end{bmatrix}$$

$$W = \begin{bmatrix} 0.0128 \\ 0.0447 \\ -1.1800 \\ 2.2251 \\ -0.2346 \\ 1.3796 \\ -0.2380 \\ -0.8916 \\ 0.3217 \\ 1.4936 \\ 0.4532 \\ 0.0418 \\ -1.6513 \\ 4.3237 \\ -0.3870 \\ 0.0195 \\ -1.0231 \\ -0.1762 \\ 1.6222 \\ 0.8967 \\ 0.3455 \\ 0.6990 \\ -0.2268 \\ 0.7616 \\ -0.0580 \\ 0.3231 \\ 0.1512 \\ 0.6458 \end{bmatrix}$$

### Example 8: Human Operation at a Chemical Plant

The time series used in this example is generated by the chaotic Mackey-Glass differential delay equation defined below:

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \quad (2.31)$$

The data of above equation is available in MATLAB (mgdata.dat) and are produced by:  $x(0)=1.2$ ,  $\tau=17$  and  $x(t)=0$  for  $t < 0$ . The variables  $x(t-18)$ ,  $x(t-12)$ ,  $x(t-6)$  and  $x(t)$  are inputs and  $x(t+6)$  is the output of the model. The number of data set produced for validity test of network is 1000. Out of that 500 data are used for training and the remaining 500 are testing the networks.

WSNN network with Morlet and WANN network with Mexican hat wavelet functions have better performance whereas WANN with Mexican hat is best with performance  $J=6.2074 \times 10^{-6}$ . In this example, WSNN network does not convergence with Sinc wavelet function. Actual output and the output of the WANN network with Mexican hat are shown in Fig. 2.16. The error is presented in this figure. The learning parameters  $C$  &  $W$  of this network are as follows.

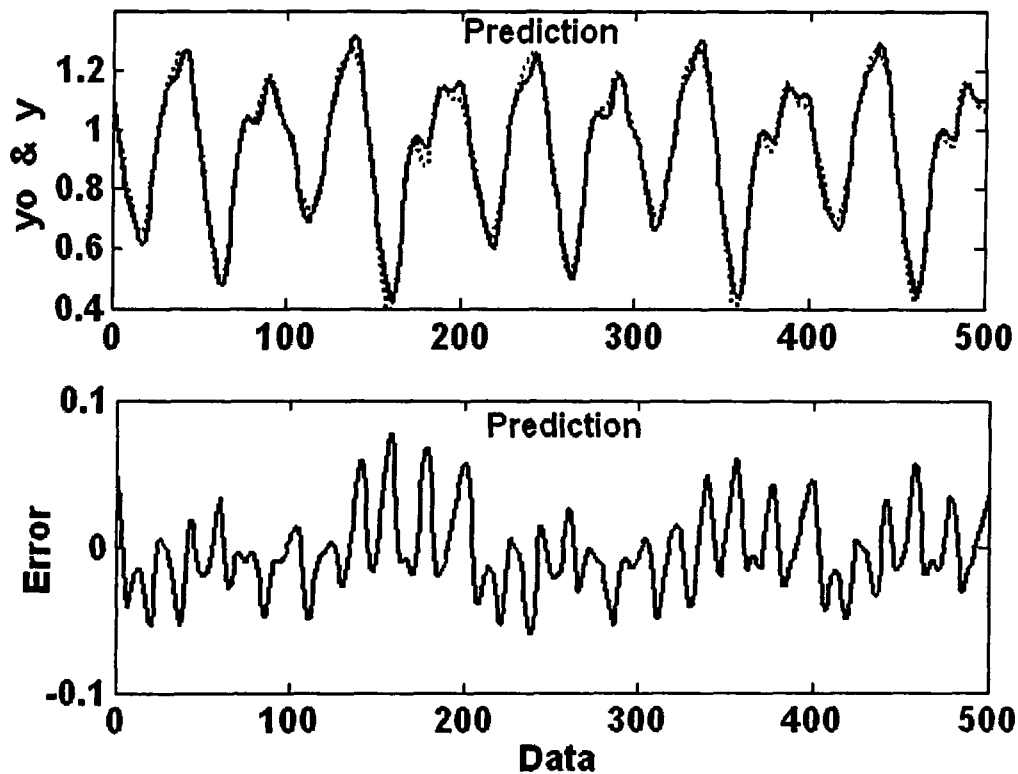


Fig. 2.16. Actual output and network output with WANN (Mexican hat) network and the error for Example 8

$$C = \begin{bmatrix} -1.7687 & -2.3279 & -2.4718 & -1.9171 \\ 0.0954 & -0.0658 & 0.3538 & 0.2268 \\ 1.6205 & 2.4201 & 0.1788 & 0.1756 \\ 0.4907 & 0.2756 & -0.0730 & 0.1768 \\ 0.4255 & 0.4772 & 0.8384 & 0.7852 \\ 0.8360 & 1.3197 & 0.5969 & -0.2602 \\ 0.3593 & 0.6888 & 0.0098 & 0.4698 \\ 0.5352 & 1.0094 & 0.3123 & 0.2134 \\ 0.9296 & 0.9555 & 1.0148 & 0.5380 \\ 0.3852 & 0.4586 & 0.0748 & 0.2369 \\ 0.1562 & 0.2770 & 0.2386 & 0.8289 \\ 0.7094 & 0.2880 & 0.6958 & 0.1025 \\ 0.8224 & 0.1423 & 0.2983 & 0.9269 \\ 0.0272 & 0.2723 & 0.5418 & 1.2093 \\ 0.7140 & 0.7101 & 0.0809 & 0.9921 \\ 0.3412 & 0.2555 & 0.9391 & 0.7334 \\ 0.8663 & 0.8206 & 0.5403 & 0.4366 \\ 0.4887 & 0.4449 & 0.4361 & 0.6267 \\ 0.7337 & 0.3865 & 0.5506 & 0.2544 \\ 0.3297 & 0.4204 & 0.3151 & 0.7794 \\ 0.2796 & 0.5065 & 0.4061 & 0.3002 \\ 0.1768 & 0.4292 & 0.2114 & 0.9432 \\ 0.1810 & 0.6450 & 0.5559 & 0.7471 \\ 0.7401 & 0.7606 & 0.8463 & 0.4224 \\ 0.3809 & 0.9226 & 0.6033 & 0.7688 \\ 0.5824 & 1.0455 & 0.6702 & 0.2624 \\ 0.1289 & 0.4668 & 0.1988 & 0.4574 \\ 0.6313 & 0.7213 & 0.3652 & 0.9892 \end{bmatrix}$$

$$W = \begin{bmatrix} -1.0867 \\ 0.4760 \\ 0.4742 \\ -0.0190 \\ -0.0887 \\ -0.7065 \\ -0.0143 \\ -0.5303 \\ -0.1470 \\ 0.0951 \\ 0.1671 \\ 0.2572 \\ 0.6524 \\ 0.7984 \\ 0.1243 \\ -0.0697 \\ 0.4345 \\ 0.5252 \\ 0.1026 \\ 0.7279 \\ 0.0478 \\ -0.0267 \\ 0.3977 \\ -0.4437 \\ -0.1951 \\ -0.2055 \\ 0.1677 \\ 0.5594 \end{bmatrix}$$

Table 2.1: Performance Index of WSNN and WANN networks for different wavelet activation functions

Examples	Networks	N.H. ( $M$ )	Performance Index ( $J$ )		
			Mexican hat	Morlet	Sinc
Example 1	WSNN	15 ( $M=5$ )	$1.1576 \times 10^{-6}$	$1.339 \times 10^{-6}$	-----
	WANN		$2.817 \times 10^{-6}$	$1.993 \times 10^{-6}$	$4.388 \times 10^{-6}$
	NN	20	$9.01 \times 10^{-6}$		
Example 2	WSNN	15 ( $M=5$ )	$4.3 \times 10^{-5}$	$3.2 \times 10^{-5}$	-----
	WANN		$2.762 \times 10^{-5}$	$1.147 \times 10^{-5}$	$3.319 \times 10^{-5}$
	NN	13	$4.019 \times 10^{-5}$		
Example 3	WSNN	15 ( $M=5$ )	$1.508 \times 10^{-4}$	$1.452 \times 10^{-4}$	-----
	WANN		$1.436 \times 10^{-4}$	$3.591 \times 10^{-4}$	$2.042 \times 10^{-4}$
	NN	15	$2.769 \times 10^{-4}$		
Example 4	WSNN	15 ( $M=5$ )	$2.6536 \times 10^{-5}$	$2.0679 \times 10^{-5}$	-----
	WANN		$1.229 \times 10^{-5}$	$9.737 \times 10^{-5}$	$2.280 \times 10^{-4}$
	NN	18	$4.849 \times 10^{-5}$		
Example 5	WSNN	15 ( $M=5$ )	$1.0976 \times 10^{-5}$	$2.6546 \times 10^{-7}$	-----
	WANN		$2.632 \times 10^{-7}$	$7.530 \times 10^{-6}$	$6.837 \times 10^{-6}$
	NN	16	$1.04 \times 10^{-5}$		
Example 6	WSNN	15 ( $M=5$ )	$9.0252 \times 10^{-6}$	$8.3505 \times 10^{-6}$	-----
	WANN		$8.668 \times 10^{-6}$	$8.228 \times 10^{-6}$	$2.622 \times 10^{-5}$
	NN	20	$1.096 \times 10^{-5}$		
Example 7	WSNN	28 ( $M=5$ )	$7.7 \times 10^{-4}$	$1.9 \times 10^{-4}$	-----
	WANN		$3.4516 \times 10^{-5}$	$3.482 \times 10^{-5}$	$9.9988 \times 10^{-5}$
	NN	12	$8.0 \times 10^{-4}$		
Example 8	WSNN	28 ( $M=5$ )	$4.0988 \times 10^{-5}$	$2.3 \times 10^{-4}$	-----
	WANN		$6.2074 \times 10^{-6}$	$3.0 \times 10^{-3}$	$1.1721 \times 10^{-5}$
	NN	12	$8.23 \times 10^{-5}$		

In Table 2.1, different networks namely Wavelet Activation function Neural Network (WANN), Wavelet Synapses Neural Network (WSNN) and Neural Network (NN) are shown in second column. In this column, the network with better performance is Bold. Third column shows the Number of Hidden neurons (N.H.). The maximum number of scaling factor ( $M$ ) for WANN and WSNN is shown in bracket. The last column is also shows the performance index  $J$  for different wavelet functions namely Mexican hat, Morlet and Sinc. The best performance index is Bold.

## 2-7 Conclusions

In this chapter, a comparative study of Wavelet Synapses Neural Network (WSNN) and Wavelet Activation Neural Network (WANN) networks is studied. Three types of wavelet activation functions, namely Mexican hat, Morlet and Sinc are tested in WSNN and WANN networks. The comparative result of different wavelets shows that Mexican activation function yield better performance in WANN network however in WSNN network most the times Morlet activation function is better. WSNN does not convergence with Sinc wavelet function. WANN network generally yields better performance than WSNN network.



# Chapter 3

## Generalized Wavelet Networks

### 3-1 Introduction

At the present scenario, wavelet decomposition emerges as a new powerful tool for function approximation due to its multi-resolution property. Wavelet Neural Networks (WNN) inspired by both the feed forward neural networks and wavelet decompositions have received considerable attention [Q. Zhang'92, 97] [J. Zhang'95] and become a popular tool for function approximation.

The main characteristic of WNN is that some kinds of wavelet functions are used as the nonlinear transformation function in the hidden layer, instead of the usual sigmoid function. Incorporating the time-frequency localization properties of wavelets and the learning of the general Neural Network (NN), WNN has shown its advantages over the regular methods such as NN for complex nonlinear system modeling.

In this chapter, two types of WNN namely Summation Sigmoid-Wavelet (SS-W) and Multiplication Sigmoid-Wavelet (MS-W) are proposed [Banakar'06a]. Literature survey indicates that all studies show the efficacy of wavelets when used in wavelet network. But none of the reported work caters a comparative study for different types of the wavelets. The presented

work is an attempt to propose a comparative study for three types of wavelet used in WNN, namely, Mexican hat, Morlet and Sinc wavelet function. The idea of this work is to use approximation of inputs by Sigmoid Activation Function (SAF) and Wavelet Activation Functions (WAF) separately and then to combine them. The SAF in NN can modulate low frequency section of signal and the WAF in WNN can modulate high frequency section especially sharp section of signal. Conjunction of SAF and WAF combines the localize approximation property of wavelets with functional approximation properties of neural network. The temporal change in dynamic system, particularly when the changes are sharp, can be accumulated in wavelets. The output of every neuron in SS-W is summation of SAF and WAF and output of each neuron in MS-W is the product of these two.

The result of these two models are compared with a Localized WNN (LWNN) that proposed in [Banakar'06b]. A local model is used in WNN to approximate output of each wavelet. It means that localization of wavelet is approximated by a linear function of inputs (i.e., local model) then precise output of the WNN hopes to be improved. By LWNN the precision of the results increase but complexity of network is increased while in two proposed SS-W and MS-W networks precision increases where as complexity decreases.

This chapter is organized as follow: In section 3-2, Localized wavelet Neural Network (LWNN) is discussed. Section 3-3 proposes sigmoid-wavelet neuron networks. SS-W & MS-W neuron networks are described under this section. Universal approximation of the proposed SS-W & MS-W neuron networks are described in section 3-4. The learning algorithm based on Gradient Descent describes in section 3-5. Structure determination of the proposed networks is derived in section 3-6. Experimental results are revealed in section 3-7 and, finally conclusions are relegated to section 3-8.

## 3-2 Localized Wavelet Neural Network (LWNN)

The structure of LWNN is shown in Fig. 3.1. Same as WANN the network is framed into four layers. Layer wise description of the network architecture has been given below:

- **Layer 1 (Input layer):** The neurons in this layer only transmit the inputs to the hidden neuron.
- **Layer 2:** This layer hold two inner sections. First the inputs are weighted with  $C$ , and then they are passed through wavelet activation function  $\psi_{a,b}$ . The mathematical expression for the output of this layer is given in (3.1).

$$O_{a,b}^2 = \psi_{a,b} \left( \sum_{i=1}^n C_{a,b}^i \cdot x_i \right) \quad (3.1)$$

- **Layer 3 (Localized Layer):** In this layer, the local models  $W_{a,b}(\mathbf{X})$  exert on outputs of second layer. It's mean that localization of wavelet is approximated by a local function to increase the precision. The local model  $W_{a,b}(\mathbf{X})$  expressed as linear function of input  $X$  as follows:

$$W_{a,b}(\mathbf{X}) = W_{a,b}^0 + W_{a,b}^1 \cdot x_1 + W_{a,b}^2 \cdot x_2 + \dots + W_{a,b}^n \cdot x_n \quad (3.2)$$

Output of nodes in third layer is:

$$O_{a,b}^3 = W_{a,b}(\mathbf{X}) \cdot O_{a,b}^2 \quad (3.3)$$

- **Layer 4 (Output Layer):** The final output of the network is:

$$\hat{Y} = \sum_{a=1}^M \sum_{b=0}^{a-1} O_{a,b}^3 \quad (3.4)$$

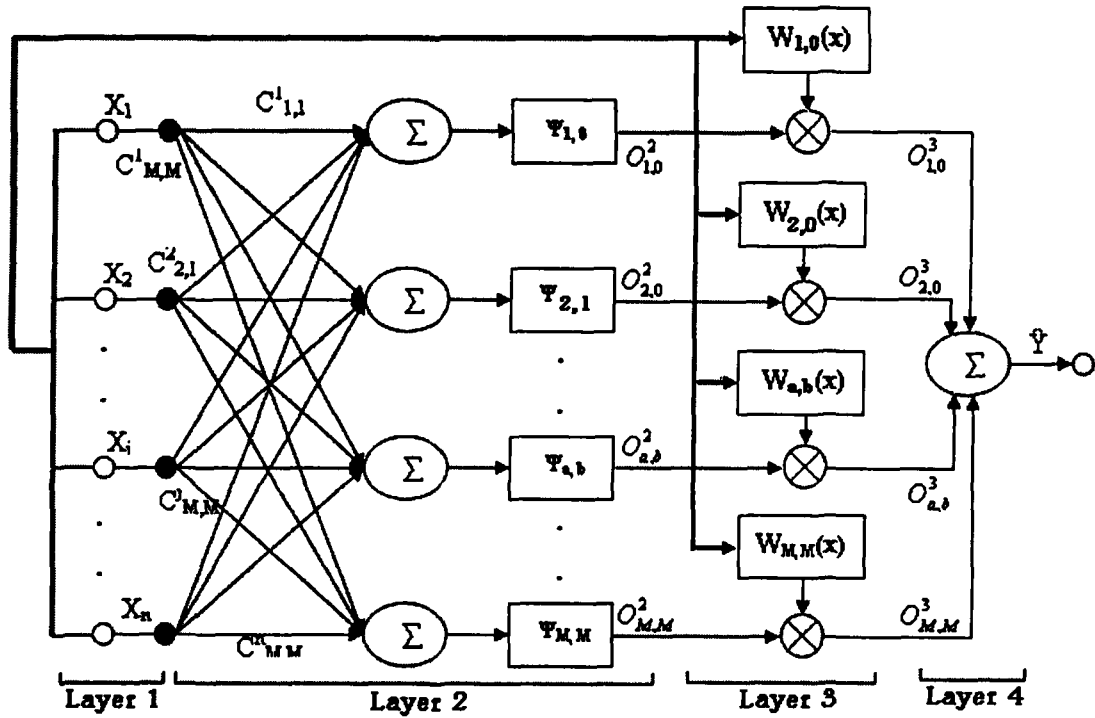


Fig. 3.1. Localized Wavelet Neural Network (LWNN)

For a WANN network with scaling factor  $M$ , there is  $M \cdot (M + 1) / 2$  hidden neuron. Therefore, the total number of learning parameters  $C$  in Layer 1 is equal to  $n \cdot M \cdot (M + 1) / 2$ , where  $n$  is the number of inputs. The number of learning parameters  $W$  in Layer 3 is also equal to  $(n + 1) \cdot M \cdot (M + 1) / 2$ . Therefore, the total number of learning parameter in this model is equal to  $(n + 2) \cdot M \cdot (M + 1) / 2$ .

### 3-3 Sigmoid-Wavelet Neuron Networks

In this section, we introduce a feed-forward network. Each neuron in this network is a combination of SAF and WAF. In both network a Sigmoid-Wavelet (S-W) neuron is used in hidden layer. If the summation operator combines SAF and WAF, that results a Summation Sigmoid-Wavelet (SS-W) neuron, Whereas with a product operator results a Multiplication Sigmoid-Wavelet (MS-W) neuron.

### 3-3.1 Feed-forward network

Feed-forward network with different type S-W neuron in the hidden layer is proposed. Figure 3.2, shows a feed-forward network. In the hidden layer, neurons represented by 'G' are S-W neurons. The output of feed-forward network is given in (3.5)

$$\hat{Y}_{WNN} = \sum_{i=1}^L W_i \cdot y_i \quad (3.5)$$

where  $y_i$  is the output of S-W neurons,  $W_i$  is the weights between hidden neuron and output neurons and  $L$  is the number of hidden neuron.

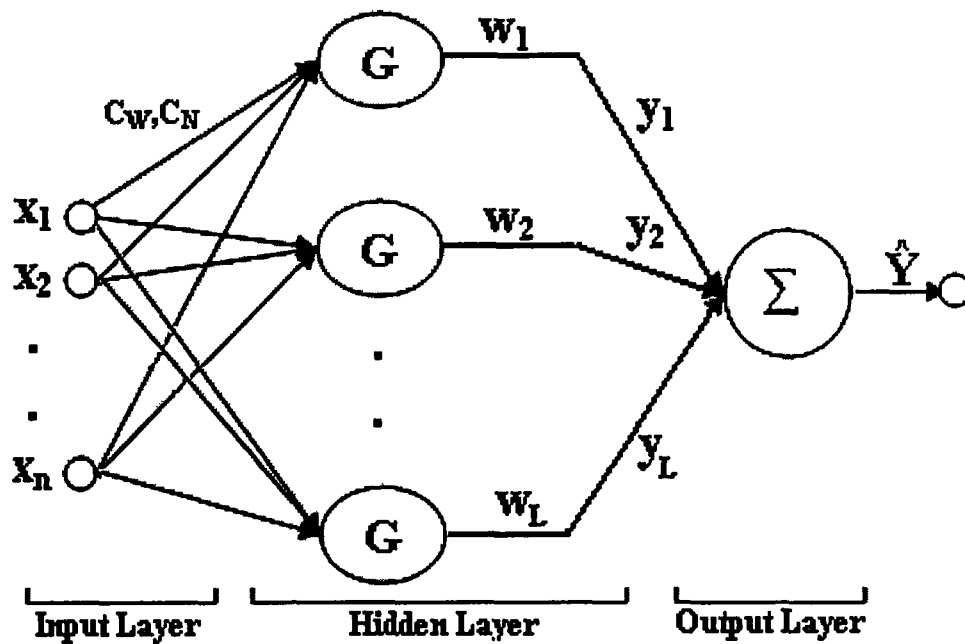


Fig. 3.2. Feed-Forward Neural Network

### 3-3.2 Summation Sigmoid-Wavelet (SS-W) Neuron

The detailed structure of S-W neuron is shown in Fig. 3.3. The output of each S-W neuron is summation of the output from SAF and WAF and is given by (3.5).

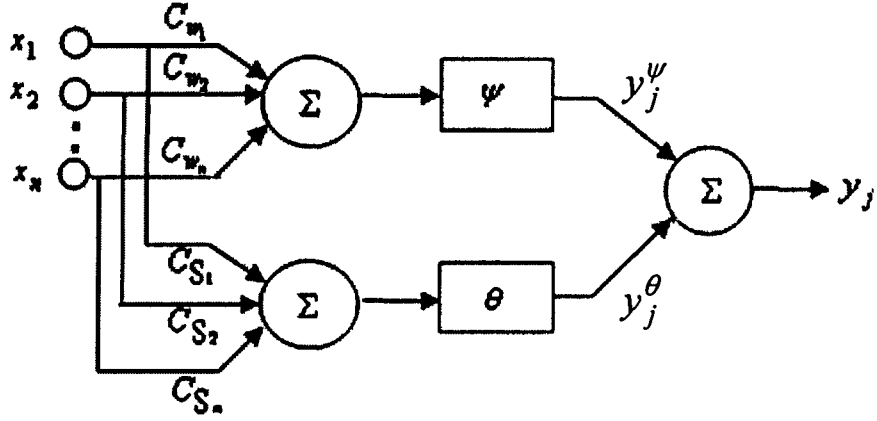


Fig. 3.3. Summation Sigmoid-Wavelet (SS-W) neuron network

$$y_j(k) = y_j^\theta(k) + y_j^\psi(k) \quad (3.6)$$

The function  $y_j^\theta$  and  $y_j^\psi$  are output of SAF and WAF for  $j^{\text{th}}$  S-W neuron, in the hidden layer, respectively. The function  $y_j^\theta$  and  $y_j^\psi$  are expressed as (3.7-3.8)

$$y_j^\theta(k) = \theta \left( \sum_{i=1}^n C_{S_i}^j \cdot x_i(k) \right) \quad (3.7)$$

$$y_j^\psi(k) = \psi \left( \sum_{i=1}^n C_{W_i}^j \cdot x_i(k) \right) \quad (3.8)$$

$x_i$  is  $i^{\text{th}}$  input.  $C_S$  and  $C_W$  are weights to inputs signal for SAF and WAF, in each hidden neuron, respectively.

### 3-3.3 Multiplication Sigmoid-Wavelet (MS-W) Neuron

The detailed structure of MS-W neuron is shown in Fig. 3.4. The output of each S-W neuron is product of the output from SAF and WAF and is given by (3.9)

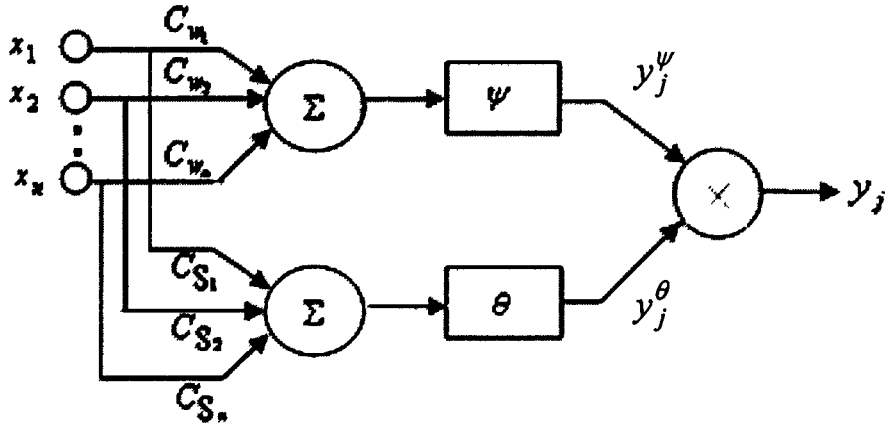


Fig. 3.4. Multiplication Sigmoid-Wavelet (MS-W) neuron network

$$y_j(k) = y_j^\theta(k) \cdot y_j^\psi(k) \quad (3.9)$$

The function  $y_j^\theta$  and  $y_j^\psi$  are outputs of SAF and WAF for  $j^{\text{th}}$  S-W neuron, in the hidden layer, respectively. The function  $y_j^\theta$  and  $y_j^\psi$  are expressed as discussed in (3.7-3.8)

### 3-4 Universal approximation of the S-W neuron networks

For system identification, the *Universal Approximation* means that for any given continuous output trajectory  $y(t)$  of any nonlinear dynamic system over any compact time-interval  $t \in [t_0, T]$ , the output  $\hat{Y}(t)$  of the SS-W and MS-W networks can approximate  $y(t)$  uniformly with arbitrarily high precision. The proposed SS-W and MS-W networks can be shown to be a universal approximate for continuous functions over compact set if it satisfies some certain conditions. The conditions for different wavelet functions are described in following theorems. Prove of the theorems have been given in appendix A.

Theorem 3.1: *Universal approximation theorem of SS-W neuron network*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an SS-W network  $f$ , with Mexican hat, Morlet or Sinc WAF, such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

Theorem 3.2: *Universal approximation theorem of MS-W neuron network with Mexican hat WAF*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an MS-W network  $f$ , with Mexican hat WAF, that satisfies condition (3.10), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

$$C_w X \neq b \pm a \frac{\sqrt{2}}{2} \quad (3.10)$$

where  $C_w = \{C_{w_1}, C_{w_2}, \dots, C_{w_n}\}$ ,  $X = \{x_1, x_2, \dots, x_n\}$ .

Theorem 3.3: *Universal approximation theorem of MS-W neuron network with Morlet WAF*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an MS-W network  $f$ , with Morlet WAF, that satisfies condition (3.11), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

$$C_w X \neq b + a(2\rho + 1) \frac{\pi}{2} \quad (3.11)$$

where  $C_w = \{C_{w_1}, C_{w_2}, \dots, C_{w_n}\}$ ,  $X = \{x_1, x_2, \dots, x_n\}$  and  $\rho$  is any integer value.

Theorem 3.4: *Universal approximation theorem of MS-W neuron network with Sinc WAF*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and



for any given  $\varepsilon > 0$  there is an MS-W network  $f$ , with Sinc WAF, that satisfies condition (3.12), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

$$C_w X \neq b + \rho a \quad (3.12)$$

where  $C_w = \{C_{w_1}, C_{w_2}, \dots, C_{w_n}\}$ ,  $X = \{x_1, x_2, \dots, x_n\}$  and  $\rho$  is any integer value.

### 3-5 Gradient Descent learning of parameters

The gradient descent learning can be achieved by minimizing the performance index  $J$  and using the parametric update equation as given in (2.18) and (2.20), respectively.

Applying gradient descent technique to modify the parameters  $W_{a,b}$  &  $C_{a,b}$  in LWNN results the following parameter update formulas for  $p^{\text{th}}$  data set:

$$\Delta_p W_{a,b}(q) = -\eta \frac{\partial J}{\partial W_{a,b}} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial W_{a,b}} \right|_p \quad (3.13)$$

$$\Delta_p C_{a,b}(q) = -\eta \frac{\partial J}{\partial C_{a,b}} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial C_{a,b}} \right|_p \quad (3.14)$$

where  $e = y - \hat{y}$  is the error between the actual output and the model output. By applying chain method to the above equations,  $\frac{\partial \hat{Y}}{\partial W_{a,b}}$  and  $\frac{\partial \hat{Y}}{\partial C_{a,b}}$  for LWNN network are derived as

follows:

$$\frac{\partial \hat{Y}}{\partial W_{a,b}^0} = \frac{\partial O_{a,b}^3}{\partial W_{a,b}^0} = \frac{\partial W_{a,b}(X)}{\partial W_{a,b}^0} \cdot O_{a,b}^2 = \psi_{a,b} \left( \sum_{i=1}^n C_{a,b}^i \cdot x_i \right) \quad (3.15)$$

$$\frac{\partial \hat{Y}}{\partial W_{a,b}^i} = \frac{\partial O_{a,b}^3}{\partial W_{a,b}^i} = \frac{\partial W_{a,b}(X)}{\partial W_{a,b}^i} \cdot O_{a,b}^2 = x_i \cdot \psi_{a,b} \left( \sum_{i=1}^n C_{a,b}^i \cdot x_i \right) \quad (3.16)$$

$$\frac{\partial \hat{Y}}{\partial C_{a,b}^i} = \frac{\partial O_{a,b}^3}{\partial C_{a,b}^i} = W_{a,b}(X) \cdot \frac{\partial O_{a,b}^2}{\partial C_{a,b}^i} = W_{a,b}(X) \cdot x_i \cdot \psi'_{a,b} \left( \sum_{i=1}^n C_{a,b}^i \cdot x_i \right) \quad (3.17)$$

where  $O_{a,b}^2$  and  $O_{a,b}^3$  are outputs of second and third layers in LWNN network as shown in Fig. 3.1.

In S-W neuron networks by applying gradient descent technique to modify the parameters  $W$ ,  $C_w$  &  $C_s$ , following parameter update formulas for  $p^{\text{th}}$  data set are resulted:

$$\Delta_p W_j(q) = -\eta \frac{\partial J}{\partial W_j} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \frac{\partial \hat{Y}}{\partial W_j} \Big|_p \quad (3.18)$$

$$\Delta_p C_{w_j}(q) = -\eta \frac{\partial J}{\partial C_{w_j}} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \frac{\partial \hat{Y}}{\partial C_{w_j}} \Big|_p \quad (3.19)$$

$$\Delta_p C_{s_j}(q) = -\eta \frac{\partial J}{\partial C_{s_j}} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \frac{\partial \hat{Y}}{\partial C_{s_j}} \Big|_p \quad (3.20)$$

Applying chain method to  $\frac{\partial \hat{Y}}{\partial W_j}$ ,  $\frac{\partial \hat{Y}}{\partial C_{w_j}}$  and  $\frac{\partial \hat{Y}}{\partial C_{s_j}}$  in above equation for SS-W and MS-

W neuron networks the following equations are derived:

#### a) For SS-W neuron network

$$\frac{\partial \hat{y}(k)}{\partial W_j} = y_j(k) = y_j^\theta(k) + y_j^\psi(k) \quad (3.21)$$

$$\frac{\partial \hat{y}(k)}{\partial C_{w_i}^j} = x_i(k) \cdot W_j \cdot \psi' \left( \sum_{i=1}^n C_{w_i}^j \cdot x_i(k) \right) \quad (3.22)$$

$$\frac{\partial \hat{y}(k)}{\partial C_{s_i}^j} = x_i(k) \cdot W_j \cdot \theta' \left( \sum_{i=1}^n C_{s_i}^j \cdot x_i(k) \right) \quad (3.23)$$

**b) For MS-W neuron network**

$$\frac{\partial \hat{y}}{\partial W_j} = y_j(k) = y_j^\theta(k) \cdot y_j^\psi(k) \quad (3.24)$$

$$\frac{\partial \hat{y}(k)}{\partial C_{w_i}^j(k)} = x_i(k) \cdot W_j(k) \cdot y_j^\theta(k) \cdot \psi' \left( \sum_{i=1}^n C_{w_i}^j(k) \cdot x_i(k) \right) \quad (3.25)$$

$$\frac{\partial \hat{y}(k)}{\partial C_{s_i}^j(k)} = x_i(k) \cdot W_j(k) \cdot y_j^\psi(k) \cdot \theta' \left( \sum_{i=1}^n C_{s_i}^j(k) \cdot x_i(k) \right) \quad (3.26)$$

$\theta$  and  $\psi$  in (3.16-3.21) are SAF and WAF, respectively. SAF  $\theta$  is given in (3.22).

$$\theta(s) = \frac{1}{1 + e^{-s}} \quad (3.22)$$

where  $s = \sum_{i=1}^n C_{s_i}^j \cdot x_i$ . Differential functions for the SAF is  $\theta'$  and expressed as follows:

$$\theta'(s) = \theta(s) \cdot (1 - \theta(s)) \quad (3.23)$$

WAF  $\psi$  and  $\psi'$  the differential functions of the WAF's, for Mexican hat, Morlet and Sinc function wavelet functions are given in (2.11-2.13) and (2.27-2.29), respectively.

### **3-6 Structure determination of S-W neuron networks**

There are two methods for selecting the number of hidden layer neurons in feed-forward neural network. In one method, initially large numbers of hidden layer neurons are selected. As the training progress, the neurons output are monitored to remove the redundant and the inactive neurons from the hidden layer. Redundant neurons are those whose output is a linear combination of the rest of the two or more neurons for all the data set. Inactive neurons are those whose output remains constant for all the data set, they add a bias to the next layer neurons. Second method employs, in the beginning, a fewer number of neurons in the hidden layer, as the learning progresses the number of hidden layer neuron is increased. While increasing the number of neurons in the hidden layer, redundancy and inactiveness should be checked.

In this thesis, method for structure determination of the network is devised for feed-forward S-W neuron network. Each SS-W and MS-W neuron is a parallel combination of WAF and SAF; it means that the number of WAF and SAF is the same. Since wavelet parameter is highly dependent on the nature of the input-output signal, initially the scaling parameter is selected with the minimum possible value ( $a=1$  for normalize I/O signal) and shifting parameter is chosen by appropriate positioning of wavelet (i.e.,  $b=0$ ). This results in a single hidden layer neuron. Later on, by gradually increasing the scaling factor and appropriate positioning of wavelets the number of neuron in the hidden layer goes on increasing, resulting in the growth of network. A criterion is specified to stop the growth of the network. The decomposition of the input signal space by the wavelets is shown in Fig. 3.5. As shown in Fig. 3.5, we select minimum number of WAF by using scaling factor  $a=1$  and shifting  $b=0$ . Therefore, the number of sigmoid activation function is one. In total, for  $a=1$  there is only one neuron constituting one WAF and SAF. In the next step the WAF with scaling factor  $a=2$  is added to previous network. For  $a=2$

shifting parameter  $b$  is change from 0 to 1. Therefore, in this stage there are three WAF along with three SAF. In this stage the numbers of neuron increase to 3. In the same way for  $a=3$  shifting parameter  $b$  is change from 0 to 2 and the number of neurons increase to 6. In this manner, the network grows itself until the specified criterion, for stopping of this growth, is accomplished.

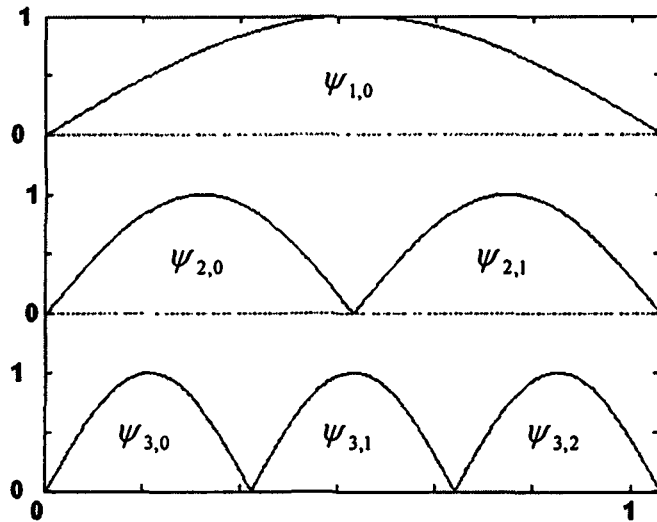


Fig. 3.5. Wavelet with different scaling factor and shifting

Various methods for correct selection of these parameters, in more effective way, have been proposed in [Zhang'95, Oussar'00]. Since the data are normalized, the number of wavelet function with scale 'a' needed to cover normalized range is no more than  $a+1$ . Let  $a$  the value for scaling factor, the value of shifting parameter  $b$  change from 0 to  $a-1$ . Here, in proposed method, firstly, the number of WAF is to be fixed and then the same number of SAF is added. To select the number of WAF, we increase scaling factor from one to higher value, in step of one, until we obtain the desired accuracy. For a value of scaling factor  $a$ , the number of hidden S-W neurons in network is equal to  $a(a+1)/2$ . Therefore, the total number of learning parameters in this model is equal to  $a(a+1) \cdot (2n+1)/2$ .

For every scaling factor  $a$ , the network is initialized and trained. The number of scaling factor and therefore number of hidden neurons continued to be increased until this increase in hidden neurons improves the model performance index  $J$  expressed in (2.18). Figure 3.6 shows the algorithm of structure determination.

### 3-7 Simulation Results

In this section, different types of dynamic systems that are discussed in chapter 1 have been considered. The proposed SS-W and MS-W neuron networks in feed-forward network have been tested with three different types of wavelet functions, namely, Mexican hat wavelet, Sinc wavelet and Morlet wavelet function.

#### **Revisited Example 1: *Linear regression with nonlinear input***

Figures 3.7 and 3.37, show the procedure of structure determination for SS-W neuron network. The scaling factor ' $a$ ' has been increased one by one. For ' $a=1$ ' there is only one WAF with one SAF that constitutes one neuron. The performance of this model is shown in Fig. 3.7 with solid line. For ' $a=2$ ' there is three WAF. First wavelet is for ' $a=1$ ' and the rest two are corresponds to ' $a=2$ ' with shifting parameter ' $b=0$  &  $1$ '. Therefore, three SAF are also added to form three SS-W neuron networks in the hidden layer of feed-forward network. In Fig. 3.7 the performance of the model with ' $a=2$ ' is shown with dashed line. In the next step increasing the scaling factor is increased to ' $a=3$ ' the number of neuron in the hidden layer is increased to six. This increase is due to addition of three WAF with shifting " $b=0, 1, \& 2$ ". In Fig. 3.7 the learning pattern of the model with ' $a=3$ ' is shown with dotted black color line. Since the performance reduces due to this increase of ' $a$ ' from 2 to 3, further increase of its value is stopped so the model with ' $a=2$ ' is to be considered as model with best performance.

Table 3.1 shows performance index  $J$  for Mexican hat, Morlet and Sinc wavelet activation functions. In this table, number of Hidden Neuron (H.N.) for each model also has been shown. Initialization of the learning parameters  $W$ ,  $C_S$  and  $C_W$  for all wavelet and the learning parameters for Mexican hat, Morlet and Sinc wavelet function corresponding to the learned SS-W neuron network are as follows:

Initialization of the learning parameters:

$$W = \begin{bmatrix} 0.1934 \\ 0.6822 \\ 0.3027 \end{bmatrix} \quad C_S = \begin{bmatrix} 0.8216 & 0.6602 & 0.3411 \\ 0.6449 & 0.3419 & 0.5340 \\ 0.8179 & 0.2897 & 0.7271 \end{bmatrix} \quad C_W = \begin{bmatrix} 0.5416 & 0.3783 & 0.5935 \\ 0.1508 & 0.8600 & 0.4965 \\ 0.6979 & 0.8536 & 0.8997 \end{bmatrix}$$

For SS-W neuron network with Mexican hat wavelet function:

$$W^f = \begin{bmatrix} 0.7254 \\ -0.5889 \\ 1.1855 \end{bmatrix} \quad C_S^f = \begin{bmatrix} 0.4502 & 0.4534 & 0.9474 \\ 0.7798 & 0.5004 & 0.0121 \\ 0.3642 & 0.4942 & 0.4880 \end{bmatrix} \quad C_W^f = \begin{bmatrix} 0.5796 & 0.0727 & 0.3838 \\ 0.6658 & 0.2252 & 0.4201 \\ 0.4610 & 0.1697 & 1.1744 \end{bmatrix}$$

For SS-W neuron network with Morlet wavelet function:

$$W^f = \begin{bmatrix} 0.4984 \\ -0.0850 \\ 1.2565 \end{bmatrix} \quad C_S^f = \begin{bmatrix} 0.1321 & 0.0556 & 0.9209 \\ 0.2035 & 0.7568 & 0.5316 \\ 0.1970 & 0.5095 & 0.3451 \end{bmatrix} \quad C_W^f = \begin{bmatrix} 1.1391 & 0.2984 & 0.0283 \\ 0.1046 & 0.8329 & 0.9810 \\ 0.5440 & -0.0482 & 0.5068 \end{bmatrix}$$

For SS-W neuron network with Sinc wavelet function:

$$W^f = \begin{bmatrix} 0.7805 \\ -0.6753 \\ 0.8946 \end{bmatrix} \quad C_S^f = \begin{bmatrix} 0.5208 & 0.2643 & 0.9944 \\ 0.0287 & 0.4157 & 0.5242 \\ 0.0940 & 0.5112 & 0.4782 \end{bmatrix} \quad C_W^f = \begin{bmatrix} 0.0416 & 0.4875 & 0.4077 \\ 1.0471 & 0.9260 & 0.4550 \\ 0.6568 & 0.5245 & 0.7591 \end{bmatrix}$$

In this example, Morlet wavelet yields better result with  $J=1.734 \times 10^{-6}$ . Figure 3.8 shows learning pattern of SS-W neuron network with different types of wavelet function.

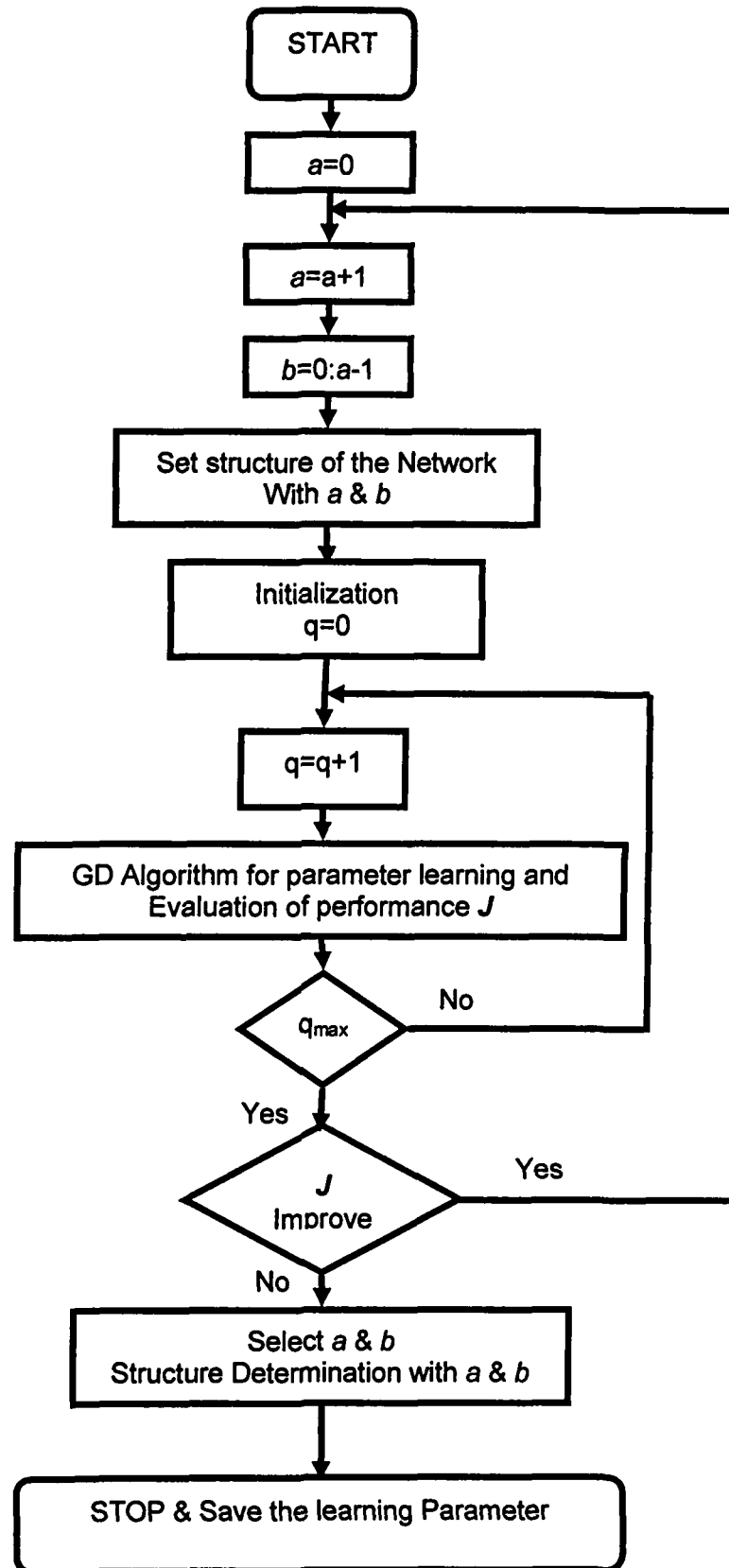


Fig. 3.6. Algorithm for Structure Determination



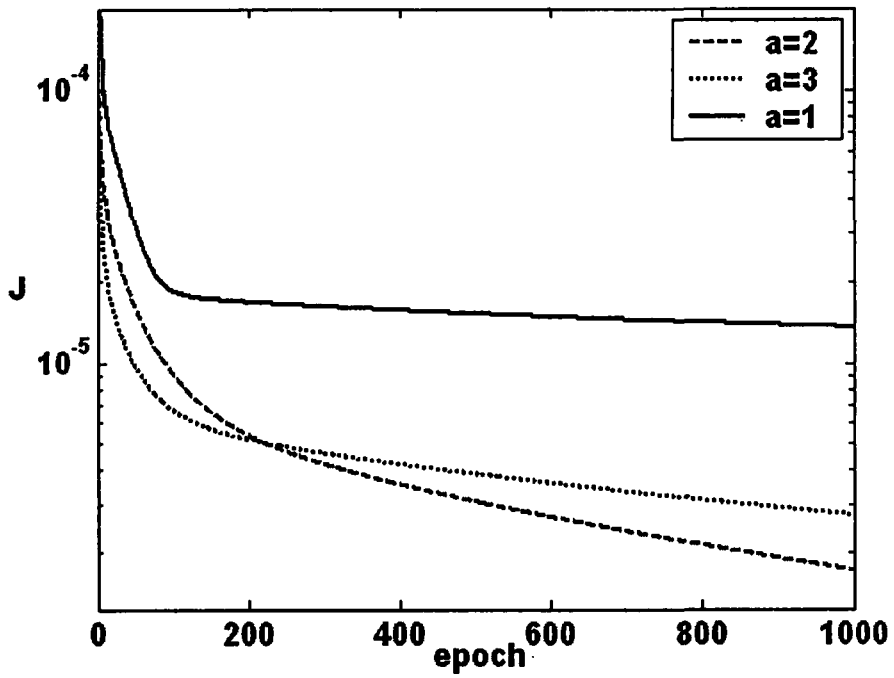


Fig. 3.7. Learning pattern of feed-forward network with SS-W neuron network using Morlet activation function with scaling factor  $a=1, 2$  &  $3$  for Example 1

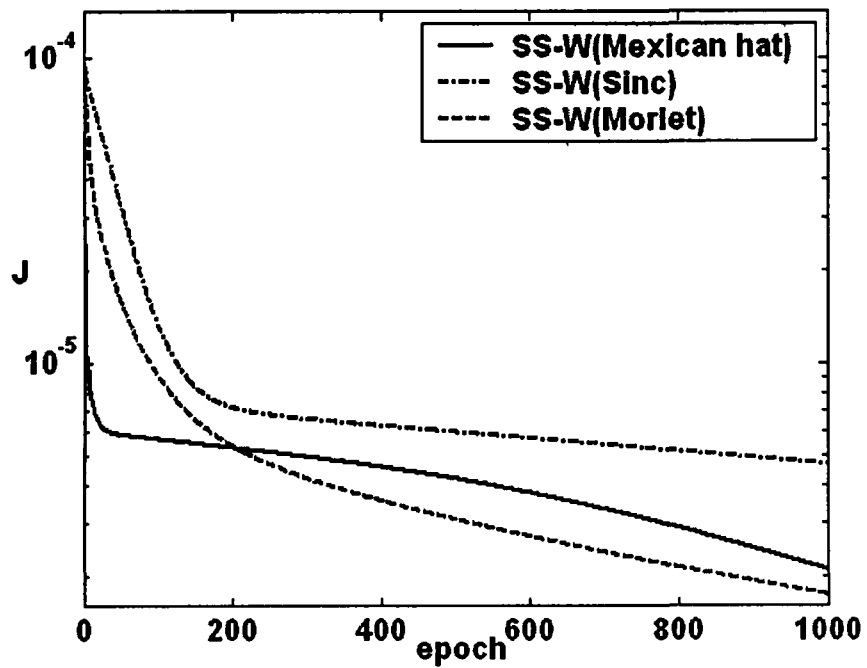


Fig. 3.8. Learning pattern of SS-W neuron network with all wavelet functions for Example 1

Figures 3.9 and 3.38, show the procedure of structure determination for MS-W neuron network. The scaling factor 'a' has been increased one by one. For 'a=1' there is only one WAF with one SAF that constitutes one neuron. The performance of this network is shown in Fig. 3.9 with solid line. For 'a=2' there is three WAF. First wavelet is for 'a=1' and the rest two corresponds to 'a=2' with shifting parameter 'b=0 & 1'. Therefore, three SAF are also added to form three MS-W neuron networks in the hidden layer of feed-forward network. In Fig. 3.9 the performance of the network with 'a=2' is shown with dashed line. Further increasing the scaling factor, i.e., 'a=3' the number of neuron in the hidden layer increased to six. This increase is due to addition of three WAFs with shifting "b=0, 1, & 2". In Fig. 3.9 the learning pattern of the network with 'a=3' is shown with dotted black color line. Since the performance reduces due to this increase of 'a' from 2 to 3, further increase of its value is stopped so the network with 'a=2' is to be considered as network with best performance. In this example, MS-W network has three hidden neuron. Initialization of the learning parameters  $W$ ,  $C_s$  and  $C_w$  for all wavelet functions is same as SS-W neuron network. Figure 3.10 shows learning pattern of MS-W neuron network with different type of wavelets. Learning parameters for Mexican hat, Morlet and Sinc wavelet function corresponding to the learned MS-W neuron network are as follows.

For MS-W neuron network with Mexican hat wavelet function:

$$W^f = \begin{bmatrix} 1.2325 \\ 0.3809 \\ 0.1769 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.5197 & 0.3334 & 0.3799 \\ 0.8730 & 0.1681 & 0.6703 \\ 0.9363 & 0.8708 & 0.9645 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.7042 & -0.1224 & 0.6061 \\ 0.4457 & 0.4127 & 0.2055 \\ 0.3613 & 0.4217 & 0.1096 \end{bmatrix}$$

For MS-W neuron network with Morlet wavelet function:

$$W^f = \begin{bmatrix} -0.0602 \\ 0.5548 \\ 1.1146 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.8185 & 0.6608 & 0.3415 \\ 0.6340 & 0.3407 & 0.5837 \\ 0.8862 & 0.3314 & 0.8063 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.6088 & 0.3229 & 0.5377 \\ -0.1648 & 0.4196 & -0.0864 \\ 0.4203 & -0.0865 & 0.6054 \end{bmatrix}$$

For MS-W neuron network with Sinc wavelet function:

$$W^f = \begin{bmatrix} 0.9520 \\ -0.2201 \\ 0.7252 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.4317 & 0.9389 & 0.3218 \\ 0.1141 & 0.4661 & 0.8700 \\ 0.8605 & 0.9715 & 0.8332 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.4475 & -0.4184 & 0.7518 \\ 0.8709 & 0.7142 & -0.0132 \\ 0.6003 & 0.7018 & 0.7514 \end{bmatrix}$$

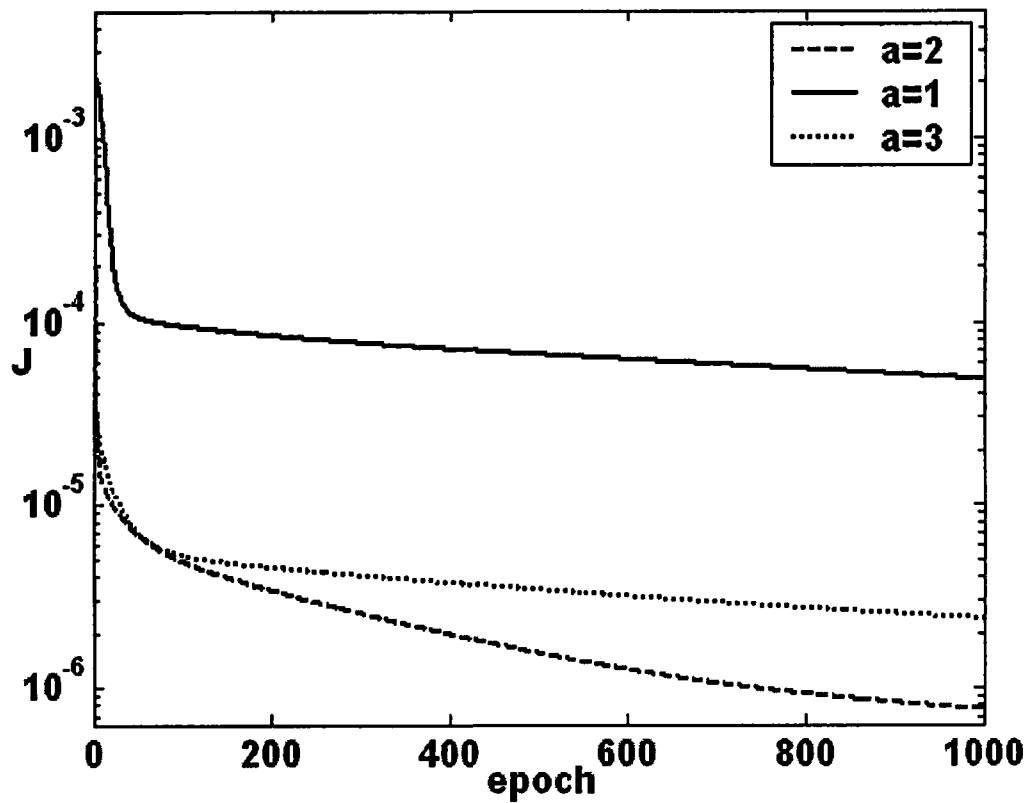


Fig. 3.9. Learning pattern of feed-forward network with MS-W neuron network using Morlet activation function with scaling factor  $a=1, 2$  &  $3$  for Example 1

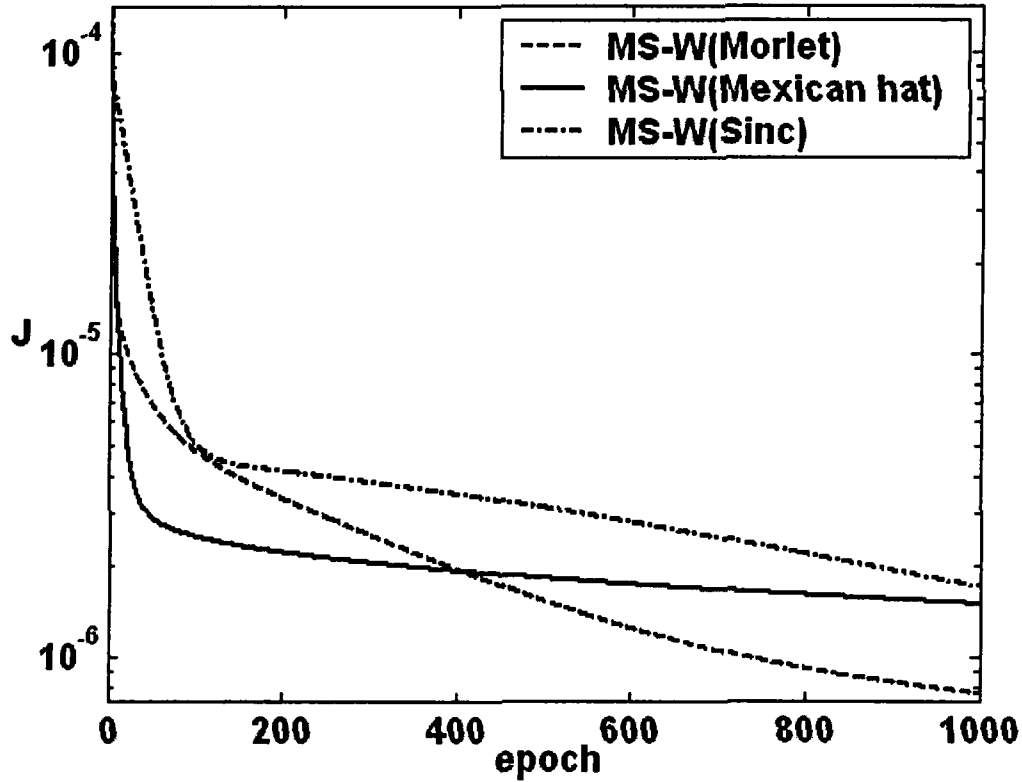


Fig. 3.10. Learning pattern of MS-W neuron network with all wavelet functions for Example 1

Table 3.1 shows performance index with different network. For each network, Wavelet function with the better performance is Bold and the best is Bold-Italic. MS-W neuron network with Morlet activation function yields better performance. Actual output & network output for MS-W with Morlet wavelet function and error between them have been shown in Fig. 3.11.

Table 3.1: Performance index ( $J$ ) with different networks and wavelet functions for Example 1

Model →	SS-W	MS-W	WNN	LWNN	NN
H.N. →	6	6	15	15	20
Mexican hat	$2.763 \times 10^{-6}$	$1.505 \times 10^{-6}$	$2.817 \times 10^{-6}$	<b><math>1.843 \times 10^{-6}</math></b>	$9.01 \times 10^{-6}$
Morlet	<b><math>1.734 \times 10^{-6}</math></b>	<i><math>7.585 \times 10^{-7}</math></i>	<b><math>1.993 \times 10^{-6}</math></b>	$5.806 \times 10^{-6}$	
Sinc	$5.566 \times 10^{-6}$	$1.714 \times 10^{-6}$	$4.388 \times 10^{-6}$	$6.750 \times 10^{-6}$	

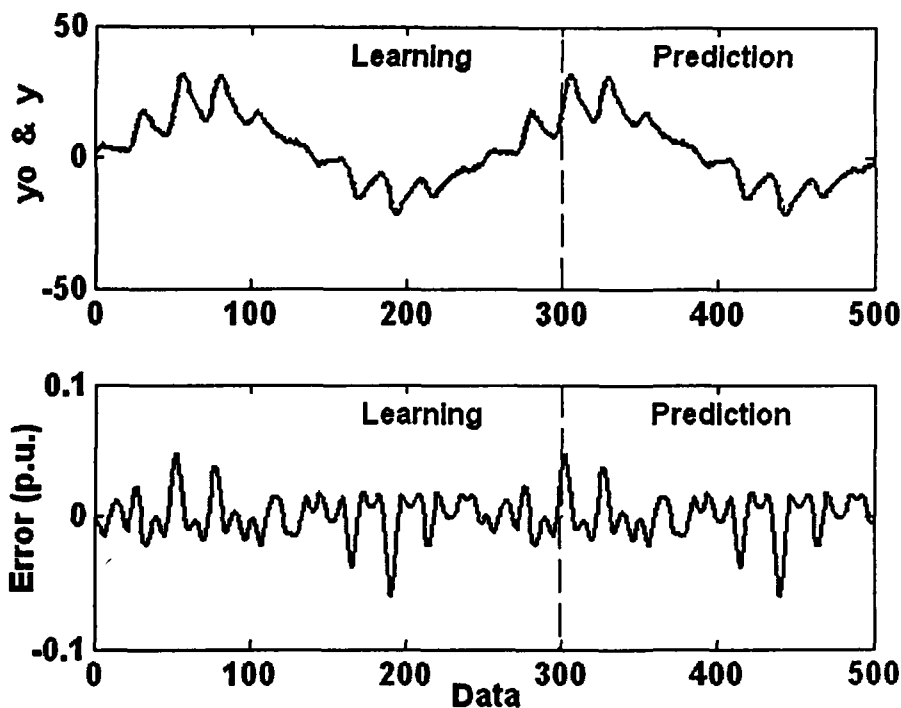


Fig. 3.11. Actual output and network output with MS-W (Morlet) neuron network and the error for Example 1

**Revisited Example 2: Non-linear regression with random input**

The structure determination of the SS-W neuron network is started from ' $\alpha=1$ '. We increase scaling factor one by one. For every scaling factor the model is being learned. Figures 3.12 and 3.37 show that an increase in scaling factor decreases the performance index. In Fig. 3.12, the performance of the network with ' $\alpha=4$ ' is shown with dashed-dot line. By increasing scaling factor from 3 to 4 the performance index does not improve. So scaling factor " $\alpha=3$ " is selected. Table 3.2 shows that SS-W model with Morlet wavelet function yield better result with performance index  $J=1.648 \times 10^{-5}$ . Figure 3.13 shows the Learning pattern of SS-W neuron network with all wavelet functions. SS-W neuron network with Morlet wavelet function yields better result. Initialization of the learning parameters  $W$ ,  $C_S$  and  $C_W$  for all wavelet functions and learning parameters for Mexican hat, Morlet and Sinc wavelet function corresponding to the learned SS-W neuron network are as follows.

Initialization of the learning parameters:

$$W = \begin{bmatrix} 0.2844 \\ 0.4692 \\ 0.0647 \\ 0.9883 \\ 0.5827 \\ 0.4235 \end{bmatrix} \quad C_s = \begin{bmatrix} 0.6945 & 0.1729 & 0.1365 \\ 0.6213 & 0.9797 & 0.0117 \\ 0.7948 & 0.2714 & 0.8939 \\ 0.9568 & 0.2523 & 0.1991 \\ 0.5225 & 0.8757 & 0.2987 \\ 0.8801 & 0.7373 & 0.6614 \end{bmatrix} \quad C_w = \begin{bmatrix} 0.4965 & 0.3419 & 0.8385 \\ 0.8997 & 0.2897 & 0.5680 \\ 0.8216 & 0.3411 & 0.3704 \\ 0.6449 & 0.5340 & 0.7027 \\ 0.8179 & 0.7271 & 0.5465 \\ 0.6602 & 0.3092 & 0.4448 \end{bmatrix}$$

For SS-W neuron network with Mexican hat wavelet function:

$$W^f = \begin{bmatrix} -0.5873 \\ 0.7124 \\ 1.1610 \\ -0.0349 \\ -0.6710 \\ 0.16116 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.2910 & 0.5654 & 0.6631 \\ 0.3976 & 0.6184 & 0.6739 \\ 0.8655 & 0.6638 & 0.0673 \\ 0.7986 & 0.9035 & 0.8843 \\ 0.3401 & 0.9343 & 0.6251 \\ 0.0450 & 0.3517 & 0.6497 \end{bmatrix} \quad C_w^f = \begin{bmatrix} -0.6071 & 0.1894 & 0.9035 \\ 0.9502 & 0.6577 & -0.3634 \\ 1.4262 & -0.1030 & 1.0077 \\ 0.7930 & 0.3450 & 0.5887 \\ 0.4494 & 1.1507 & 1.1337 \\ 0.8773 & 0.0386 & 0.5877 \end{bmatrix}$$

For SS-W neuron network with Morlet wavelet function:

$$W^f = \begin{bmatrix} 0.5257 \\ 0.2387 \\ -0.3102 \\ 1.1563 \\ 0.1041 \\ 0.5969 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.7220 & 0.1751 & 0.0541 \\ 0.6647 & 0.9902 & -0.0926 \\ 0.7962 & 0.2844 & 0.9160 \\ 1.0360 & 0.2470 & -0.0480 \\ 0.5490 & 0.8723 & 0.2637 \\ 0.9188 & 0.7393 & 0.5517 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.4195 & -0.0524 & 1.5715 \\ 1.4692 & 0.8091 & 0.8285 \\ 0.6935 & 0.3470 & 0.6080 \\ -0.0476 & 1.0244 & 1.2828 \\ 0.7678 & 0.7572 & 0.3325 \\ 1.2763 & 0.6754 & 0.3700 \end{bmatrix}$$

For SS-W neuron network with Sinc wavelet function:

$$W^f = \begin{bmatrix} -1.0267 \\ -0.5687 \\ 0.6634 \\ 0.6977 \\ 0.0858 \\ 1.0382 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.5016 & 0.3292 & 0.4022 \\ 0.5665 & 1.0474 & 0.2288 \\ 0.9160 & 0.2490 & 0.9086 \\ 1.3856 & 0.0917 & -0.1136 \\ 0.6364 & 0.8915 & 0.2880 \\ 1.1642 & 0.6049 & 0.4129 \end{bmatrix} \quad C_w^f = \begin{bmatrix} -0.2107 & 0.7048 & 0.7356 \\ 0.8408 & 0.7642 & 0.7181 \\ 1.3764 & 0.4185 & 0.3031 \\ 0.6284 & 0.2844 & 0.6090 \\ 0.9498 & 0.7066 & 0.4947 \\ 1.5183 & 0.3564 & 0.1966 \end{bmatrix}$$

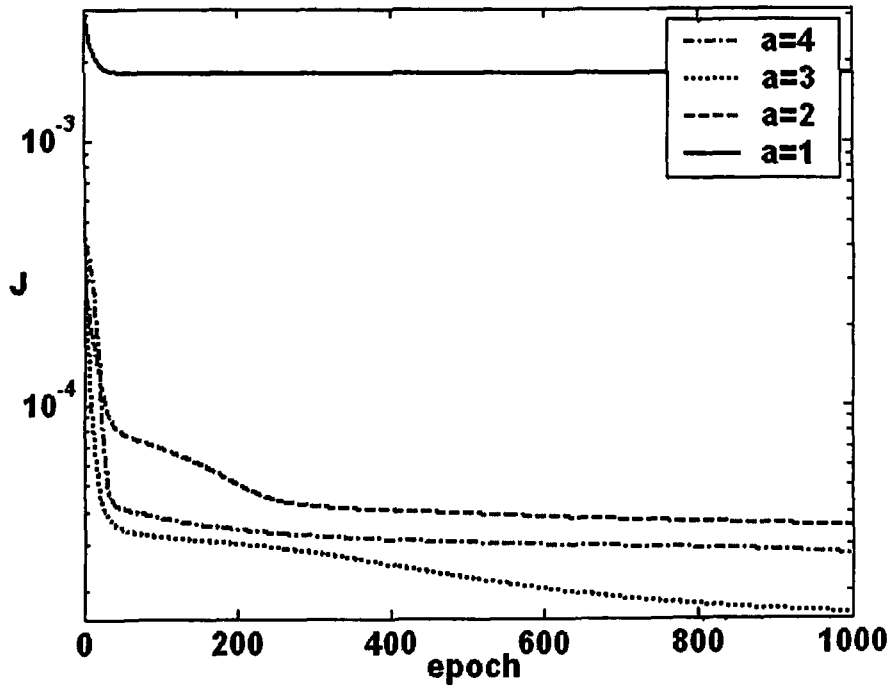


Fig. 3.12. Learning pattern of feed-forward network with SS-W neuron network using Morlet activation function with scaling factor  $a=1, 2, 3$  &  $4$  for Example 2

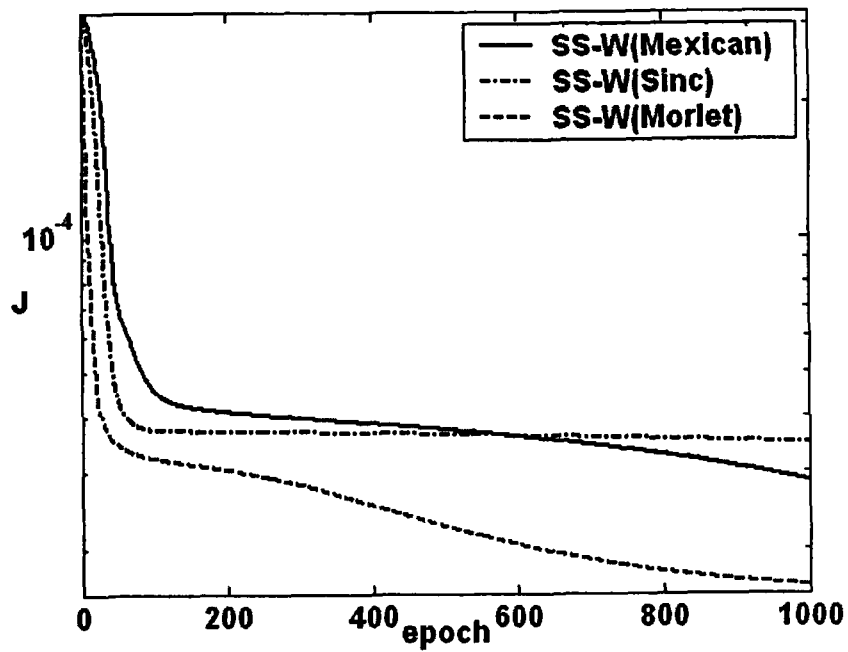


Fig. 3.13. Learning pattern of SS-W neuron network with all wavelet functions for Example 2

The structure determination of the MS-W neuron network is shown in Fig. 3.14 and 3.38. We increase scaling factor one by one. For every scaling factor, the model is being learned. Figure 3.14 shows that an increase in scaling factor from 3 to 4 decreases the performance index. So scaling factor " $\alpha=3$ " is selected. Figure 3.15 shows MS-W neuron network with Morlet wavelet function and performance index  $J=1.14 \times 10^{-5}$  yields better result than other wavelet function. Initialization of the learning parameters  $W$ ,  $C_s$  and  $C_w$  for all wavelet functions is the same as SS-W neuron network. Learning parameters for Mexican hat, Morlet and Sinc wavelet function corresponding to the learned MS-W neuron network are as follows.

For MS-W neuron network with Mexican hat wavelet function:

$$W^f = \begin{bmatrix} 0.3663 \\ 0.3697 \\ 1.2722 \\ 0.0115 \\ 0.3907 \\ 0.3525 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.8239 & 0.4427 & 0.3504 \\ 0.6466 & 0.7398 & 0.4643 \\ 0.1086 & 0.6310 & 0.4287 \\ 0.7272 & 0.9670 & 0.5741 \\ 0.4846 & 0.8960 & -0.0742 \\ 0.1493 & 0.0599 & 0.5516 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.4605 & 1.4307 & 1.3181 \\ 0.8790 & 0.2679 & 0.3782 \\ 1.4699 & 0.2401 & 0.0996 \\ 0.4654 & 0.5797 & 0.7328 \\ 0.8827 & 0.4866 & 0.9822 \\ 1.0165 & 0.3759 & 0.6195 \end{bmatrix}$$

For MS-W neuron network with Morlet wavelet function:

$$W^f = \begin{bmatrix} 0.3819 \\ 0.5532 \\ 0.4300 \\ 1.1513 \\ 1.0745 \\ 0.3216 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.6968 & 0.2118 & 0.1425 \\ 0.6278 & 0.9899 & 0.0559 \\ 0.8312 & 0.2727 & 0.9200 \\ 0.9930 & 0.1459 & 0.2559 \\ 0.6058 & 0.8981 & 0.4131 \\ 0.8711 & 0.7548 & 0.6410 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.5980 & -0.0123 & 1.2977 \\ 0.9313 & 0.7336 & 0.4126 \\ 0.7562 & 0.2912 & 0.3371 \\ 0.0433 & 0.9934 & 0.9832 \\ 0.5472 & 0.9982 & -0.0975 \\ 0.9107 & 0.4966 & 0.5188 \end{bmatrix}$$



For MS-W neuron network with Sinc wavelet function:

$$W^f = \begin{bmatrix} -0.8328 \\ -0.1318 \\ 0.4229 \\ 1.2157 \\ -0.0080 \\ 0.8076 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.6599 & 0.3699 & 0.3790 \\ 0.5568 & 1.0286 & 0.1483 \\ 0.8306 & 0.2402 & 0.9583 \\ 1.5580 & -0.0885 & -0.3822 \\ 0.5983 & 0.8568 & 0.2690 \\ 0.9680 & 0.7346 & 0.6711 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.0662 & 0.8344 & 0.7076 \\ 0.8341 & 0.4317 & 0.5897 \\ 1.1880 & 0.4777 & 0.2306 \\ 0.7968 & 0.2937 & 0.9282 \\ 0.8891 & 0.6864 & 0.4932 \\ 1.2417 & 0.0826 & 0.0906 \end{bmatrix}$$

Table 3.2 shows SS-W and MS-W neuron networks with Morlet wavelet function have better performance index than WNN and NN. Actual output & predicted output of MS-W neuron network, that has best performance, and error are shown in Fig. 3.16.

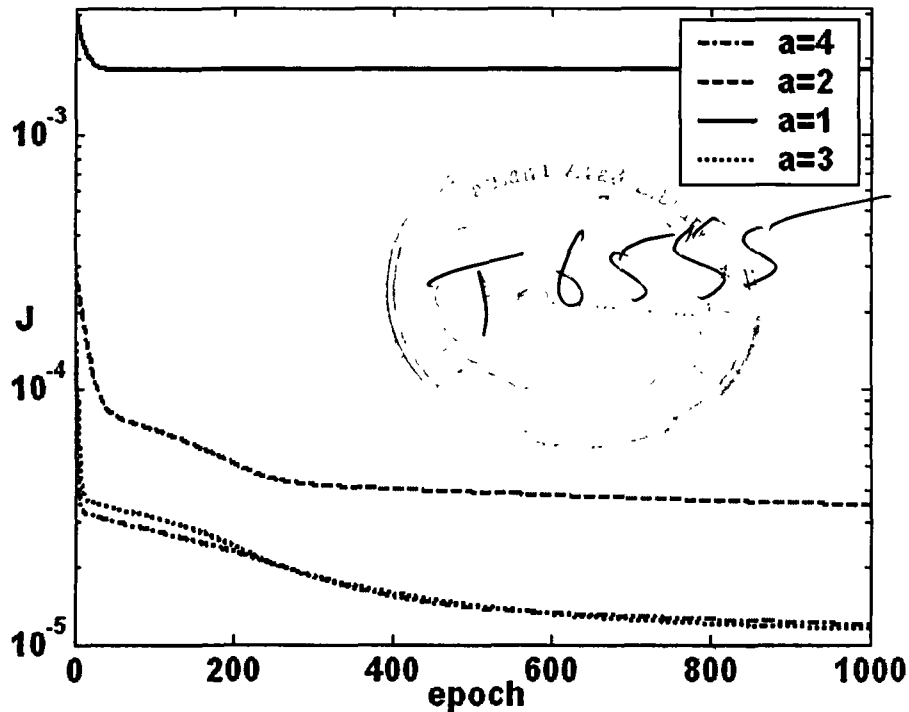


Fig. 3.14. Learning pattern of feed-forward network with MS-W neuron network using Morlet activation function with scaling factor  $a=1, 2, 3$  &  $4$  for Example 2

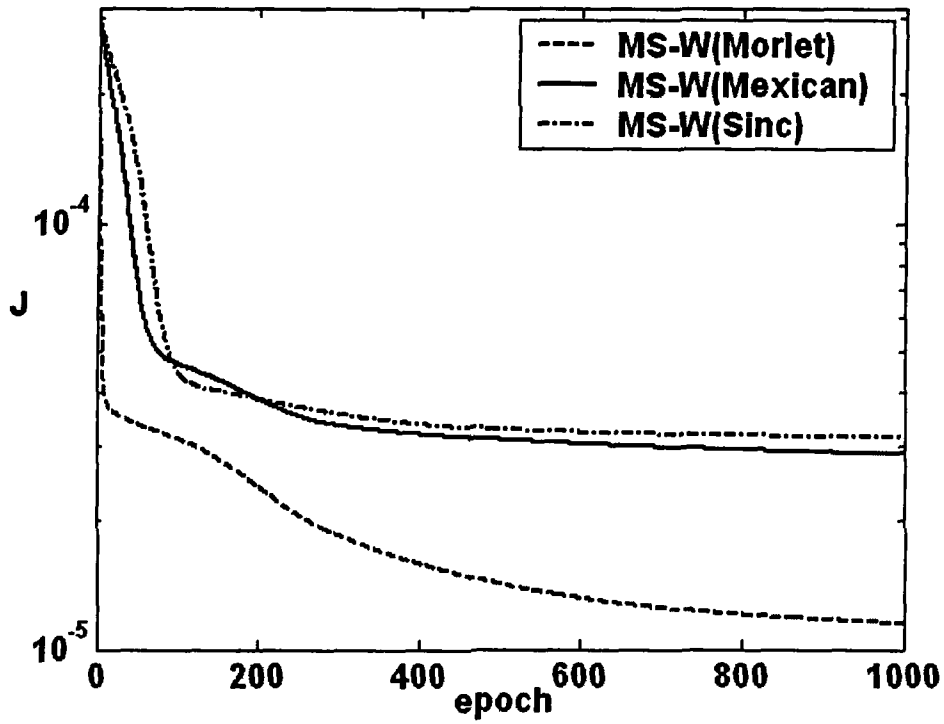


Fig. 3.15. Learning pattern of MS-W neuron network with all wavelet functions for Example 2

Table 3.2: Performance index ( $J$ ) with different networks and wavelet functions for Example 2

Model →	SS-W	MS-W	WNN	LWNN	NN
H.N. →	12	12	15	15	13
Mexican hat	$2.838 \times 10^{-5}$	$2.861 \times 10^{-5}$	$2.762 \times 10^{-5}$	<b><math>1.425 \times 10^{-5}</math></b>	$4.019 \times 10^{-5}$
Morlet	<b><math>1.648 \times 10^{-5}</math></b>	<b><math>1.145 \times 10^{-5}</math></b>	<b><math>1.147 \times 10^{-5}</math></b>	$2.730 \times 10^{-5}$	
Sinc	$3.457 \times 10^{-5}$	$3.131 \times 10^{-5}$	$3.319 \times 10^{-5}$	$1.403 \times 10^{-4}$	

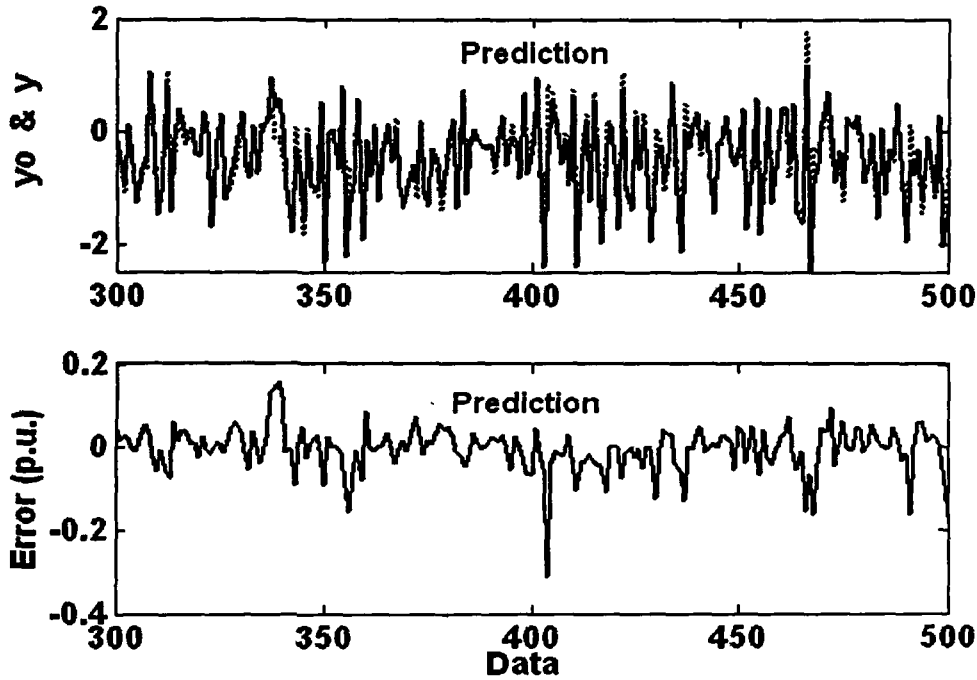


Fig. 3.16. Actual output & predicted output of MS-W (Morlet) neuron network and the error for Example 2

### Revisited Example 3: *Non-Linear Regression with Non-Linear Input*

To determine structure of the SS-W model scaling factor is increased. As shown in Fig. 3.17 and 3.37 scaling factor ' $\alpha=2$ ' is a good selection for this model because there is a decrease in performance index by further increasing the scaling factor from ' $\alpha=2$ ' to ' $\alpha=3$ '. Figure 3.18 shows the learning pattern of SS-W neuron network with all wavelet functions. Morlet wavelet function with performance index  $J= 1.2831 \times 10^{-4}$  yields better result. Initialization of the learning parameters for all wavelet functions and learning parameters for Mexican hat, Morlet and Sinc wavelet function corresponding to the learned SS-W neuron network are as follows.

Initialization of the learning parameters:

$$W = \begin{bmatrix} 0.0575 \\ 0.3675 \\ 0.6314 \end{bmatrix} \quad C_s = \begin{bmatrix} 0.1536 & 0.7275 \\ 0.6756 & 0.4783 \\ 0.6992 & 0.5548 \end{bmatrix} \quad C_w = \begin{bmatrix} 0.7176 & 0.4543 \\ 0.6926 & 0.4418 \\ 0.0840 & 0.3532 \end{bmatrix}$$

For SS-W neuron network with Mexican hat wavelet function:

$$W^f = \begin{bmatrix} 1.4043 \\ -1.1737 \\ 0.8438 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.5219 & 0.3848 \\ 0.8660 & 0.0500 \\ 0.0877 & 0.7753 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 1.1255 & 0.1423 \\ 0.6189 & 0.6988 \\ 1.1825 & 0.7516 \end{bmatrix}$$

5

For SS-W neuron network with Morlet wavelet function:

$$W^f = \begin{bmatrix} -0.1404 \\ -0.6101 \\ 1.4617 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.4138 & 0.1439 \\ 0.8885 & 0.2143 \\ 0.1535 & 0.2371 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.6539 & 0.9401 \\ -0.1573 & 0.8090 \\ 1.4455 & -0.0524 \end{bmatrix}$$

For SS-W neuron network with Sinc wavelet function:

$$W^f = \begin{bmatrix} 1.6235 \\ -1.2487 \\ 0.4890 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.6378 & 0.8295 \\ 0.6725 & 0.4529 \\ 0.6619 & 1.0008 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 1.2126 & 0.0908 \\ 1.4398 & 1.0214 \\ 0.7364 & 0.8657 \end{bmatrix}$$

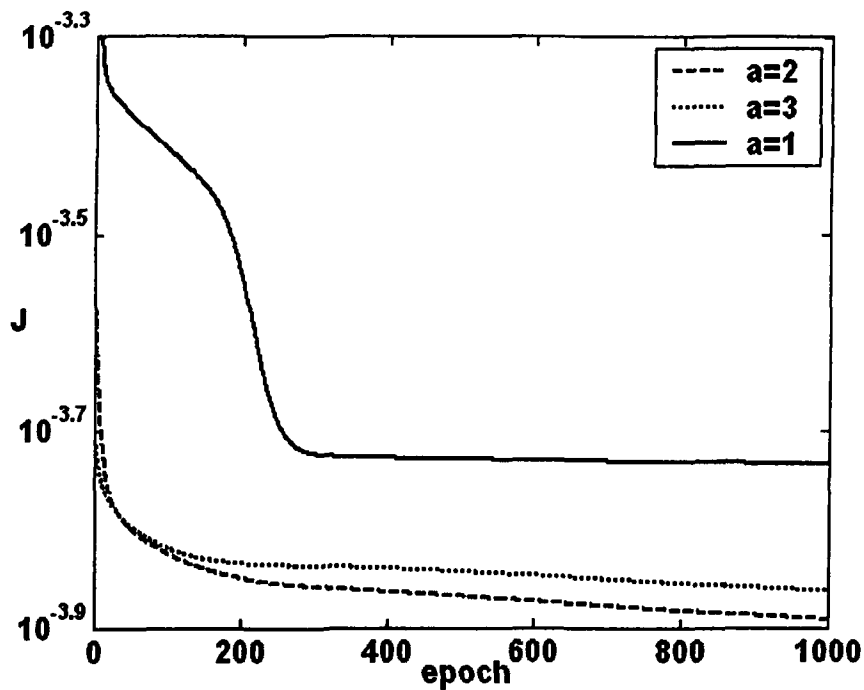


Fig. 3.17. Learning pattern of feed-forward network with SS-W neuron network using Morlet activation function with scaling factor a=1, 2 & 3 for Example 3

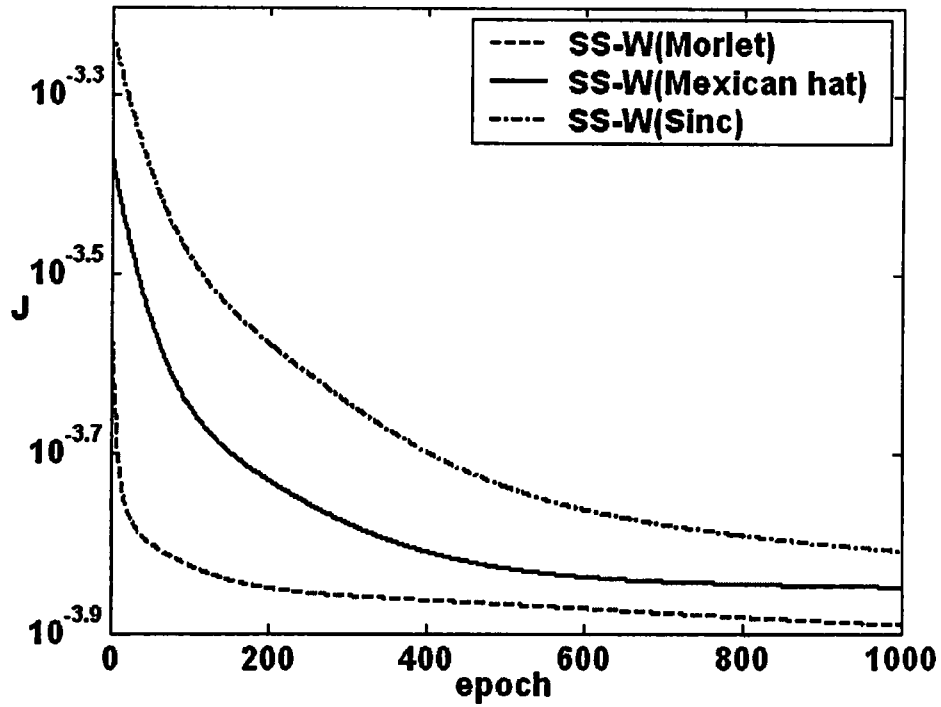


Fig. 3.18. Learning pattern of SS-W neuron network with all wavelet functions for Example 3

To determine structure of the MS-W neuron network, scaling factor is increased. As shown in Fig. 3.19 and 3.38 scaling factor ' $a=2$ ' is a good selection for this model because there is a decrease in performance index by further increasing the scaling factor from ' $a=2$ ' to ' $a=3$ '. Figure 3.20 shows MS-W neuron network with Morlet wavelet function yields better result. Initialization of the learning parameters for all wavelet functions is the same as SS-W neuron network. Learning parameters for Mexican hat, Morlet and Sinc wavelet function corresponding to the learned MS-W neuron network are as follows.

For MS-W neuron network with Mexican hat wavelet function:

$$W^f = \begin{bmatrix} 0.9291 \\ 0.0907 \\ 0.7180 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.4274 & 0.6136 \\ 0.8978 & 0.3487 \\ 0.7241 & 0.1487 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 1.2520 & 0.0636 \\ 0.6861 & 0.1496 \\ 0.8687 & 0.7700 \end{bmatrix}$$

For MS-W neuron network with Morlet wavelet function:

$$W^f = \begin{bmatrix} -0.7266 \\ 0.2420 \\ 0.7478 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.1789 & 0.7075 \\ 0.7362 & 0.4613 \\ 0.6930 & 0.5831 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 1.0280 & -0.0306 \\ 0.1226 & 0.0151 \\ 0.6264 & 0.2613 \end{bmatrix}$$

For MS-W neuron network with Sinc wavelet function:

$$W^f = \begin{bmatrix} 0.8351 \\ -0.7163 \\ 1.4378 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.4455 & 0.6482 \\ 0.5390 & 0.7465 \\ 0.4525 & 1.1529 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 1.0189 & -0.1066 \\ 0.9876 & 0.3791 \\ 1.4656 & 0.5487 \end{bmatrix}$$

Table 3.3 shows that MS-W neuron network have better performance with  $J=1.361 \times 10^{-4}$ . The actual output & output of MS-W model with Morlet function and error are shown in Fig. 3.21.

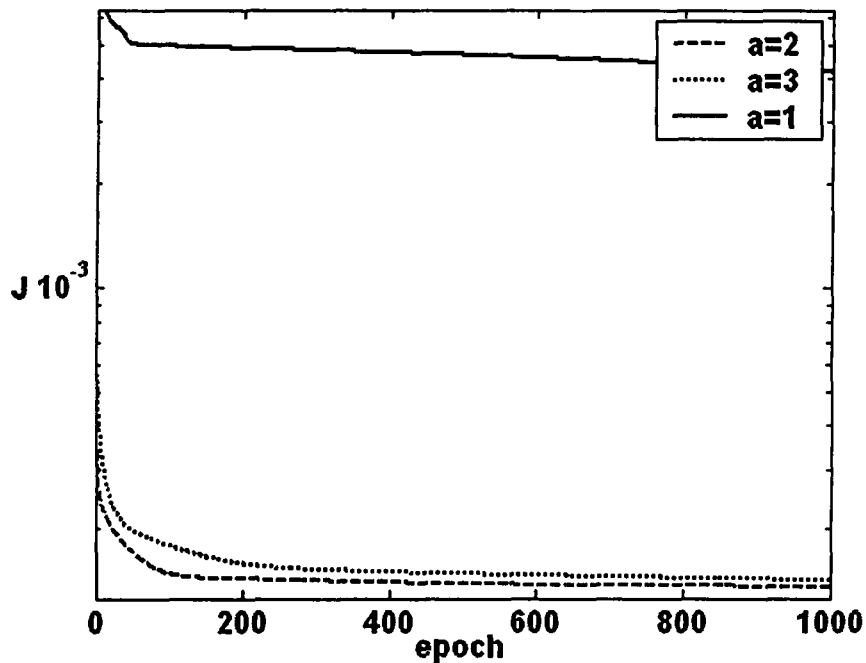


Fig. 3.19. Learning pattern of feed-forward network with MS-W neuron network with scaling factor  $a=1, 2$  &  $3$  for Example 3

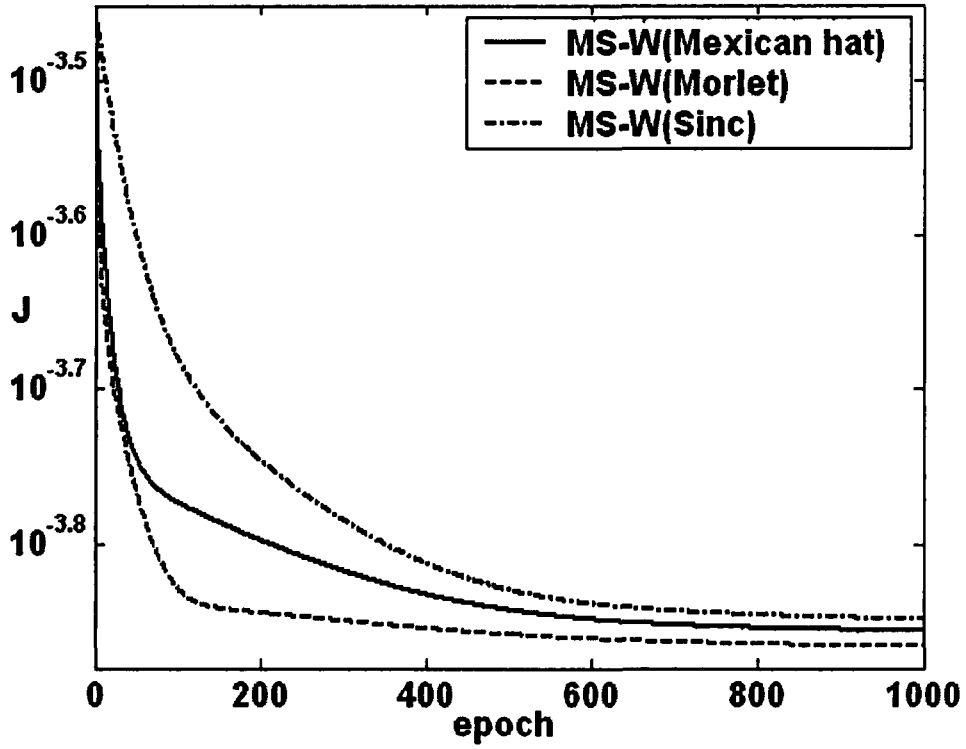


Fig. 3.20. Learning pattern of MS-W neuron network with all wavelet functions for Example 3

Table 3.3: Performance index ( $J$ ) with different networks and wavelet functions for Example 3

Model →	SS-W	MS-W	WNN	LWNN	NN
H.N. →	6	6	15	15	15
Mexican hat	$1.415 \times 10^{-4}$	$1.381 \times 10^{-4}$	$1.436 \times 10^{-4}$	$1.412 \times 10^{-4}$	$2.769 \times 10^{-4}$
Morlet	$1.283 \times 10^{-4}$	$1.361 \times 10^{-4}$	$3.591 \times 10^{-4}$	$1.753 \times 10^{-3}$	
Sinc	$1.553 \times 10^{-4}$	$1.418 \times 10^{-4}$	$2.042 \times 10^{-4}$	$1.403 \times 10^{-4}$	

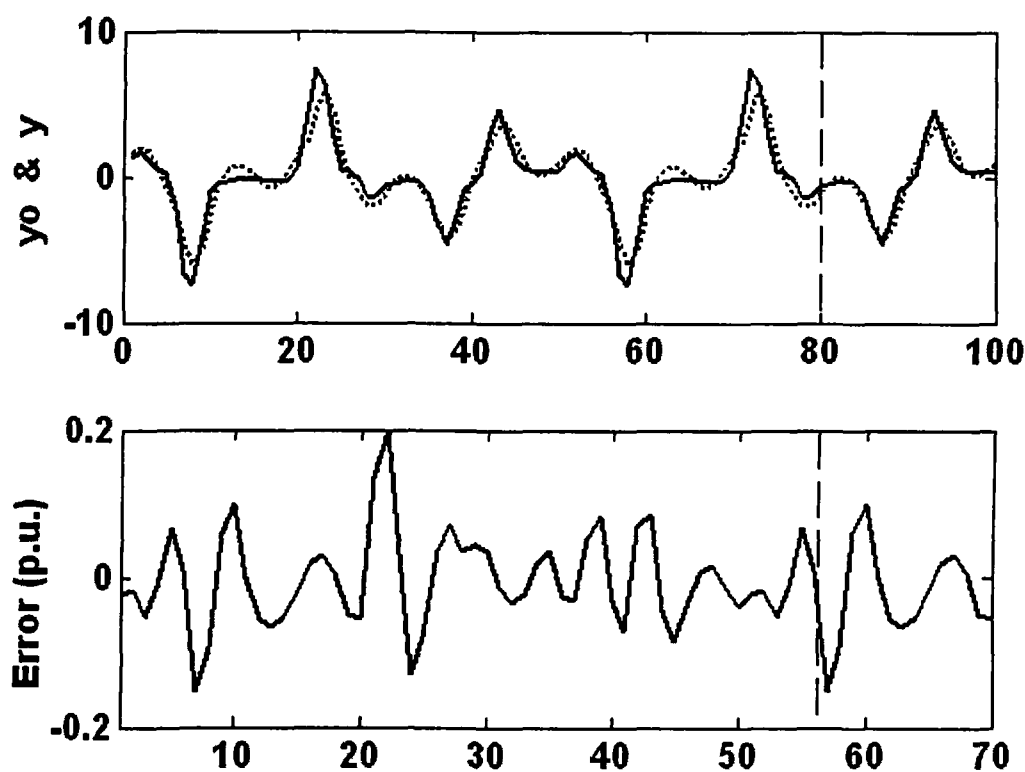


Fig. 3.21. Actual output & predicted output of MS-W (Morlet) neuron network and the error for Example 3

#### **Revisited Example 4: *Non-linear Regression of Input and output***

The number of hidden neuron for SS-W neuron network is selected by evaluating the performance index of the model in each step with an increase in scaling factor. Figures 3.22 and 3.37 show the performance index  $J$  for scaling factor ' $a=1$ ', ' $a=2$ ' and ' $a=3$ ' with one, three and six WAF, respectively. Scaling factor ' $a=3$ ' is selected and therefore in SS-W neuron network there is six WAF in parallel with six SAF. Figure 3.23 shows that Morlet wavelet function yield better result than Mexican hat and Sinc wavelet function. The performance index of SS-W neuron network with Morlet function is  $J=5.8144 \times 10^{-6}$ . Initialization of the learning parameters  $W$ ,  $C_S$  and  $C_W$  for all wavelet functions and the learning parameters for Mexican hat, Morlet and Sinc wavelet function corresponding to the learned MS-W neuron network are as follows.



Initialization of the learning parameters

$$W = \begin{bmatrix} 0.4057 \\ 0.9354 \\ 0.9169 \end{bmatrix} \quad C_s = \begin{bmatrix} 0.1388 & 0.6037 \\ 0.2027 & 0.2721 \\ 0.1987 & 0.1988 \end{bmatrix} \quad C_w = \begin{bmatrix} 0.4102 & 0.3528 \\ 0.8936 & 0.8131 \\ 0.0578 & 0.0098 \end{bmatrix}$$

For SS-W neuron network with Mexican hat wavelet function:

$$W^f = \begin{bmatrix} 0.5871 \\ -0.1212 \\ 0.5307 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.5852 & 1.0646 \\ 0.9196 & 0.3019 \\ 0.2273 & 0.3373 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.6828 & 0.5328 \\ 0.4357 & 0.7834 \\ 0.7960 & 1.0929 \end{bmatrix}$$

For SS-W neuron network with Morlet wavelet function:

$$W^f = \begin{bmatrix} -0.2472 \\ 0.3717 \\ 0.9206 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.2047 & 0.6295 \\ 0.2612 & 0.3877 \\ 0.2491 & 0.4202 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.7202 & 0.9475 \\ 1.1281 & 1.3045 \\ 0.8534 & 0.4250 \end{bmatrix}$$

For SS-W neuron network with Sinc wavelet function:

$$W^f = \begin{bmatrix} 1.1042 \\ -0.6174 \\ 0.4585 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.3352 & 0.4900 \\ 0.2803 & 0.3633 \\ 0.7303 & 0.8005 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.7244 & 0.4270 \\ 0.3401 & 0.7207 \\ 0.6166 & 0.6950 \end{bmatrix}$$

The number of hidden neuron for MS-W neuron network is selected by evaluating the performance index of the network in each step with an increase in scaling factor. Figures 3.24 and 3.38 show the performance index  $J$  for scaling factor ' $a=1$ ', ' $a=2$ ' and ' $a=3$ ' with one, three and six WAF, respectively. For this example scaling factor ' $a=2$ ' is selected and therefore in MS-W network there is six WAF in conjunction with six SAF.

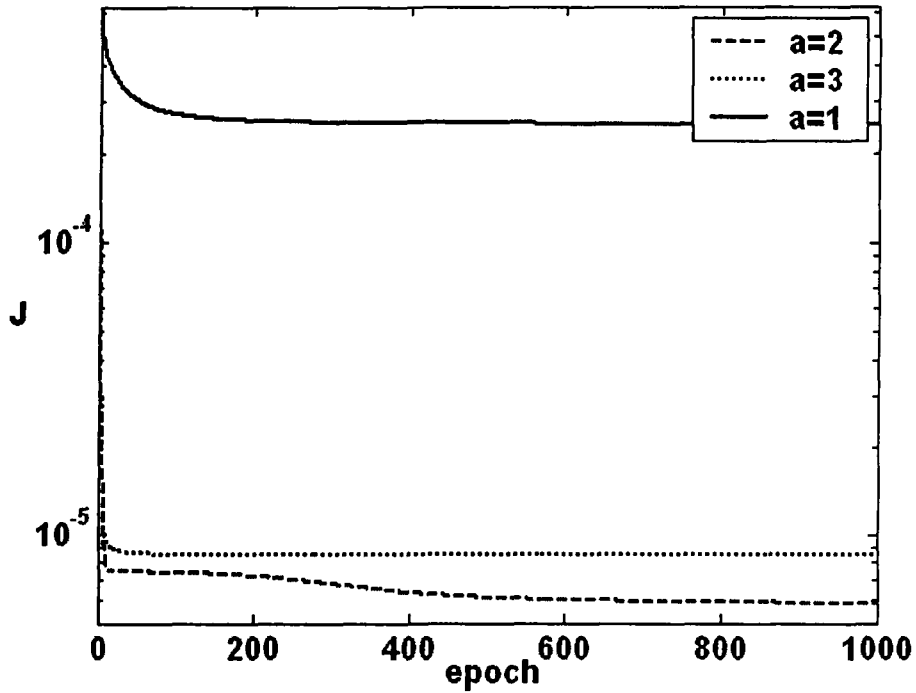


Fig. 3.22. Learning pattern of feed-forward network with SS-W neuron network using Morlet activation function with scaling factor  $a=1, 2$  &  $3$  for Example 4

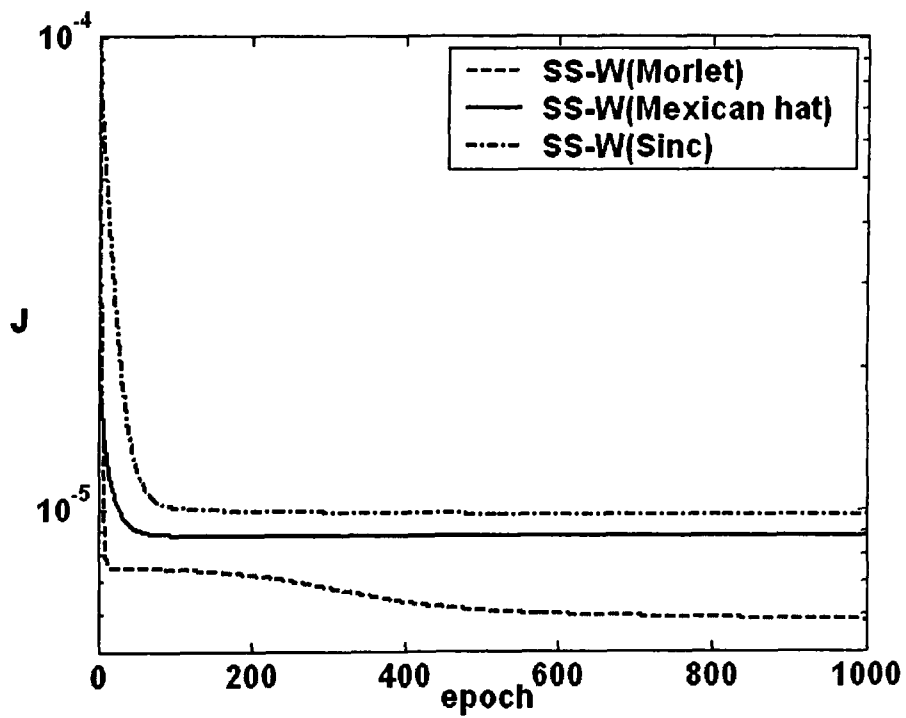


Fig. 3.23. Learning pattern of SS-W neuron network with all wavelet functions for Example 4

The learning pattern for different wavelet function is shown in Fig. 3.25. Morlet wavelet function with performance  $J=6.394 \times 10^{-6}$  is better. Initialization of the learning parameters for all wavelet functions is the same as SS-W neuron network. Learning parameters for Mexican hat, Morlet and Sinc wavelet function corresponding to the learned MS-W neuron network are as follows.

For MS-W neuron network with Mexican hat wavelet function:

$$W^f = \begin{bmatrix} 0.2633 \\ 0.9345 \\ 0.2150 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.4207 & 0.9500 \\ 0.5962 & 0.8587 \\ 0.7877 & 0.1882 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.4913 & 0.5861 \\ 1.1295 & 0.2060 \\ 0.7792 & 0.9338 \end{bmatrix}$$

For MS-W neuron network with Morlet wavelet function:

$$W^f = \begin{bmatrix} 0.4836 \\ 0.7579 \\ 1.8742 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.4960 & 0.4889 \\ 0.4507 & 0.8517 \\ 0.3975 & 0.9263 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.1344 & 0.6227 \\ 0.7711 & 0.7040 \\ 0.5809 & 0.6553 \end{bmatrix}$$

For MS-W neuron network with Sinc wavelet function:

$$W^f = \begin{bmatrix} 0.4035 \\ 0.1384 \\ 0.5889 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.6582 & 0.6469 \\ 0.5010 & 0.2651 \\ 0.4738 & 0.2669 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.7281 & 0.5289 \\ 1.0474 & 0.3279 \\ 0.9093 & 1.1550 \end{bmatrix}$$

From Table 3.4, SS-W neuron network with Morlet wavelet function is the best. Figure 3.26 shows its actual output & identify output and the error between them.

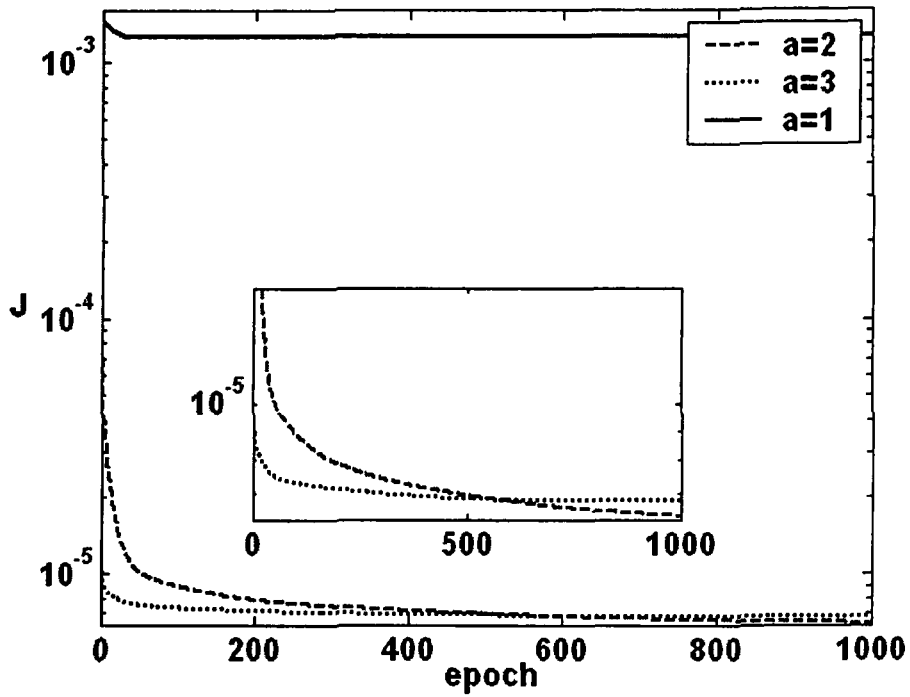


Fig. 3.24. Learning pattern of feed-forward network with MS-W neuron network with scaling factor  $a=1, 2$  &  $3$  for Example 4

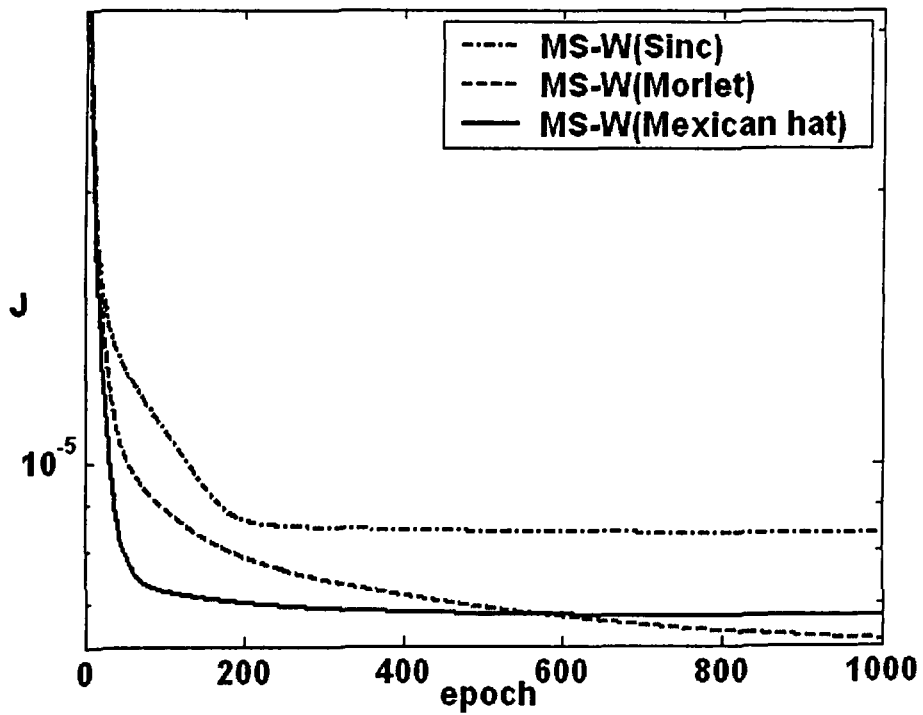


Fig. 3.25. Learning pattern of MS-W neuron network with all wavelet functions for Example 4

Table 3.4: Performance index ( $J$ ) with different networks and wavelet functions for Example 4

Model →	SS-W	MS-W	WNN	LWNN	NN
H.N. →	6	6	15	15	18
Mexican hat	$8.709 \times 10^{-6}$	$6.772 \times 10^{-6}$	$1.229 \times 10^{-5}$	$2.933 \times 10^{-5}$	$4.849 \times 10^{-5}$
Morlet	$5.814 \times 10^{-6}$	$6.394 \times 10^{-6}$	$9.737 \times 10^{-5}$	$1.973 \times 10^{-5}$	
Sinc	$9.673 \times 10^{-6}$	$8.349 \times 10^{-6}$	$2.280 \times 10^{-4}$	$9.372 \times 10^{-6}$	

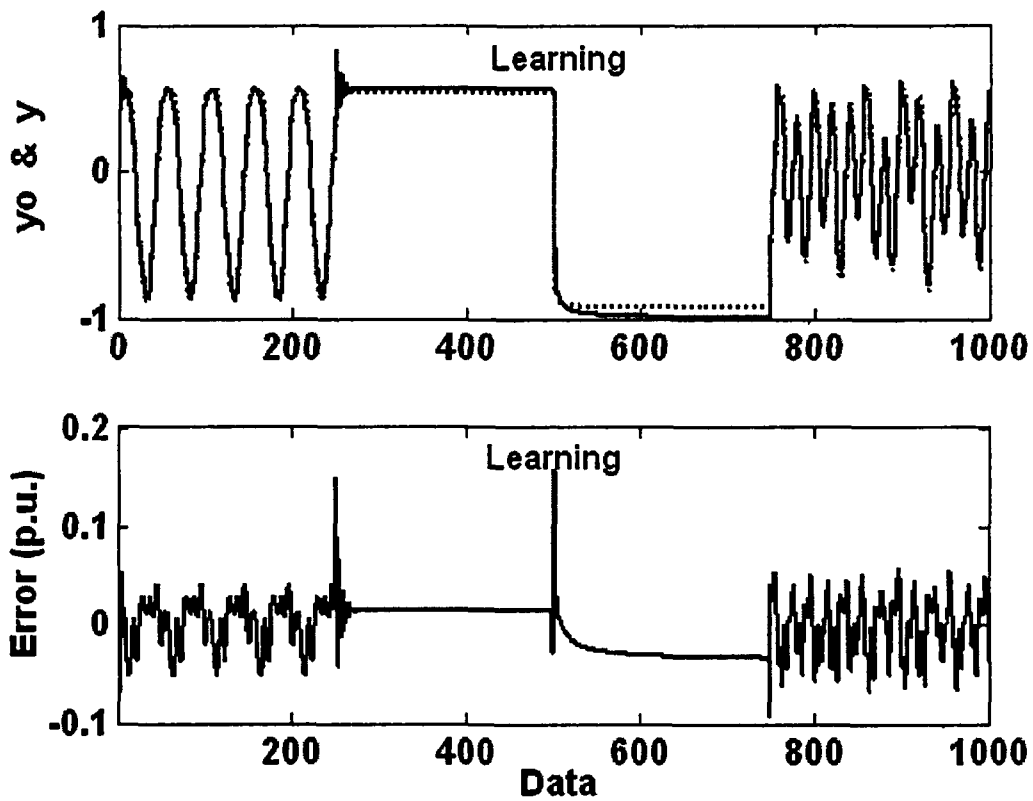


Fig. 3.26. Actual output & predicted output of SS-W (Morlet) neuron network and the error for Example 4

### Revisited Example 5: Gas Furnace Data

For structure determination of the SS-W neuron network, scaling factor increased one by one. Figures 3.27 show learning pattern for network with scaling factor equal ‘ $\alpha=1$ ’, ‘ $\alpha=2$ ’ and ‘ $\alpha=3$ ’. From Fig. 3.27 and 3.37 the model with scaling factor ‘ $\alpha=3$ ’ is selected. Figures 3.28, shows learning patterns for SS-W model with different wavelet functions. As shown in Table 3.5, Morlet wavelet function has better result with  $J=1.676 \times 10^{-7}$ . Initialization of the learning parameters  $W$ ,  $C_s$  and  $C_w$  for all wavelet functions and the learning parameters for Mexican hat, Morlet and Sinc wavelet function corresponding to the learned SS-W neuron network are as follows.

Initialization of the learning parameters:

$$W = \begin{bmatrix} 0.9501 \\ 0.2311 \\ 0.6068 \\ 0.4859 \\ 0.8913 \\ 0.7621 \end{bmatrix} \quad C_s = \begin{bmatrix} 0.1388 & 0.0152 & 0.8462 \\ 0.2027 & 0.7467 & 0.5251 \\ 0.1987 & 0.4451 & 0.2026 \\ 0.6037 & 0.9318 & 0.6721 \\ 0.2721 & 0.4659 & 0.8381 \\ 0.1988 & 0.4186 & 0.0196 \end{bmatrix} \quad C_w = \begin{bmatrix} 0.4564 & 0.9218 & 0.4102 \\ 0.0185 & 0.7382 & 0.8936 \\ 0.8214 & 0.1762 & 0.0578 \\ 0.4447 & 0.4057 & 0.3528 \\ 0.6154 & 0.9354 & 0.8131 \\ 0.7919 & 0.9169 & 0.0098 \end{bmatrix}$$

For SS-W neuron network with Mexican hat wavelet function:

$$W^f = \begin{bmatrix} -0.0497 \\ -0.6125 \\ 1.2767 \\ 0.3384 \\ 0.4341 \\ 0.8267 \end{bmatrix} \quad C_s^f = \begin{bmatrix} -0.0064 & 0.8239 & 0.7241 \\ 0.7804 & 0.5592 & 0.3090 \\ 0.3359 & 0.0917 & 0.9981 \\ 0.8861 & 0.6255 & 0.6050 \\ 0.4079 & 0.7789 & 0.8239 \\ 0.3213 & -0.0793 & 0.6170 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.7798 & 0.2648 & 0.1917 \\ 0.8278 & 0.9842 & -0.0306 \\ -0.0892 & -0.2117 & 0.9923 \\ 0.3719 & 0.3185 & 0.6285 \\ 0.8370 & 0.7128 & 0.4825 \\ 0.8309 & -0.0775 & 0.3643 \end{bmatrix}$$

For SS-W neuron network with Morlet wavelet function:

$$W^f = \begin{bmatrix} 0.7503 \\ 0.6885 \\ 0.2407 \\ 0.2820 \\ 0.1944 \\ 0.9577 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.5363 & 0.3312 & 0.7165 \\ 0.7994 & 0.5259 & 0.9740 \\ 0.6606 & 0.2490 & 0.7564 \\ 0.9880 & 0.5863 & 0.7473 \\ 0.9439 & 0.0322 & 0.4417 \\ 0.0198 & 0.5337 & 0.6517 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.0515 & 0.4726 & 0.9382 \\ 0.6862 & 0.7400 & 0.4478 \\ 0.5112 & 0.1053 & 0.0084 \\ 0.6421 & 0.7742 & 0.6421 \\ 0.6189 & 0.8886 & 0.4937 \\ 0.5659 & 0.8577 & -0.2022 \end{bmatrix}$$

For SS-W neuron network with Sinc wavelet function:

$$W^f = \begin{bmatrix} -0.5029 \\ 0.6198 \\ 0.3330 \\ 0.2710 \\ -0.2038 \\ 0.6084 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.8742 & 0.6873 & 0.0355 \\ 0.5390 & 0.5749 & 1.0030 \\ 0.2309 & 0.1239 & 0.2185 \\ 0.9098 & -0.0485 & 0.1770 \\ 0.6723 & 0.2953 & 0.6804 \\ 0.1571 & 0.0407 & 0.6745 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.2194 & 0.9243 & 0.0115 \\ 0.5954 & 0.8175 & 0.7418 \\ 0.2222 & 0.3052 & 0.8288 \\ 0.8815 & 0.3829 & 0.5185 \\ 0.1997 & 0.5362 & 0.2375 \\ 0.3975 & 0.0198 & 0.5854 \end{bmatrix}$$

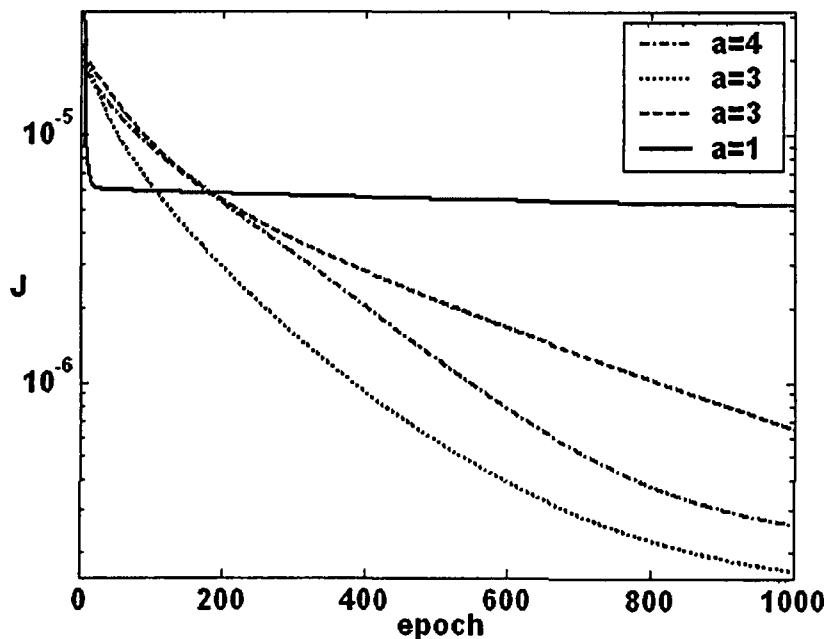


Fig. 3.27. Learning pattern of feed-forward network with SS-W neuron network using Morlet activation function with scaling factor a=1, 2, 3 & 4 for Example 5

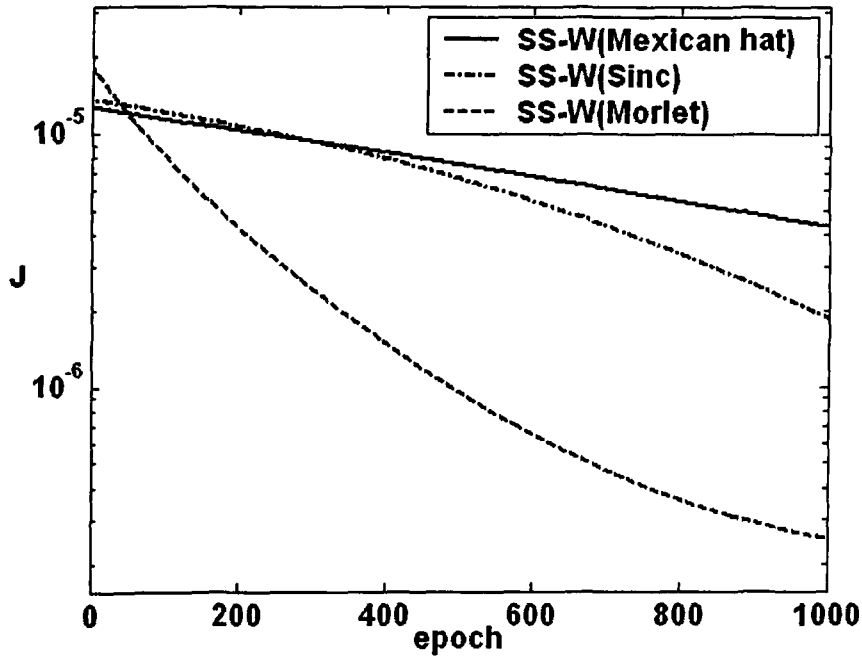


Fig. 3.28. Learning pattern of SS-W neuron network with all wavelet functions for Example 5

Figure 3.29 shows learning pattern for MS-W neuron network with scaling factor ' $\alpha=1$ ', ' $\alpha=2$ ' and ' $\alpha=3$ '. From this figure and Fig. 3.38, MS-W neuron network with scaling factor ' $\alpha=2$ ' is selected. Learning pattern of MS-W neuron network with different types of wavelet function is shown in Fig. 3.30. Morlet wavelet function yields better result with performance  $J=1.5258 \times 10^{-7}$ . Initialization of the learning parameters for all wavelet functions and the learning parameters for Mexican hat, Morlet and Sinc wavelet function corresponding to the learned MS-W neuron network are as follows.

Initialization of the learning parameters

$$W = \begin{bmatrix} 0.5364 \\ 0.1632 \\ 0.2109 \end{bmatrix} \quad C_s = \begin{bmatrix} 0.1311 & 0.1661 & 0.6170 \\ 0.0682 & 0.9114 & 0.2689 \\ 0.1252 & 0.1362 & 0.2206 \end{bmatrix} \quad C_w = \begin{bmatrix} 0.2168 & 0.2292 & 0.3066 \\ 0.6517 & 0.6674 & 0.7206 \\ 0.0527 & 0.3109 & 0.9544 \end{bmatrix}$$



For MS-W neuron network with Mexican hat wavelet function:

$$W^f = \begin{bmatrix} 0.5263 \\ 1.0675 \\ 0.9779 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.3676 & 0.2345 & 0.2319 \\ 0.8084 & 0.3898 & 1.0246 \\ 0.5864 & 0.5886 & 1.0055 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.4345 & 0.4230 & 0.9894 \\ -0.0158 & 0.3467 & 0.6899 \\ -0.0097 & -0.1890 & 0.9405 \end{bmatrix}$$

For MS-W neuron network with Morlet wavelet function:

$$W^f = \begin{bmatrix} 0.5364 \\ 0.1632 \\ 0.2109 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.1311 & 0.1661 & 0.6170 \\ 0.0682 & 0.9114 & 0.2689 \\ 0.1252 & 0.1362 & 0.2206 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.2168 & 0.2292 & 0.3066 \\ 0.6517 & 0.6674 & 0.7206 \\ 0.0527 & 0.3109 & 0.9544 \end{bmatrix}$$

For MS-W neuron network with Sinc wavelet function:

$$W^f = \begin{bmatrix} 0.6745 \\ 0.4021 \\ 0.7322 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.2838 & 0.3909 & 1.1315 \\ -0.0343 & 0.2040 & 0.6045 \\ 0.7817 & 0.3056 & 0.2442 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.5535 & 0.5946 & 0.8460 \\ 0.1973 & 0.0426 & 0.7206 \\ -0.0816 & 0.6083 & 0.7539 \end{bmatrix}$$

Table 3.5 shows comparative study of the different neuron networks. In this example, MS-W neuron network with Morlet wavelet function is better. Actual output and output of MS-W (with Morlet) model and error between them are shown in Fig. 3.31.

Table 3.5: Performance index ( $J$ ) with different networks and wavelet functions for Example 5

Model →	SS-W	MS-W	WNN	LWNN	NN
H.N. →	12	6	15	15	16
Mexican hat	$1.208 \times 10^{-6}$	$2.384 \times 10^{-6}$	<b><math>2.632 \times 10^{-7}</math></b>	$8.065 \times 10^{-6}$	$1.04 \times 10^{-5}$
Morlet	<b><math>1.676 \times 10^{-7}</math></b>	<b><math>1.665 \times 10^{-7}</math></b>	$7.530 \times 10^{-6}$	<b><math>3.719 \times 10^{-7}</math></b>	
Sinc	$1.343 \times 10^{-6}$	$6.641 \times 10^{-6}$	$6.837 \times 10^{-6}$	$6.837 \times 10^{-6}$	

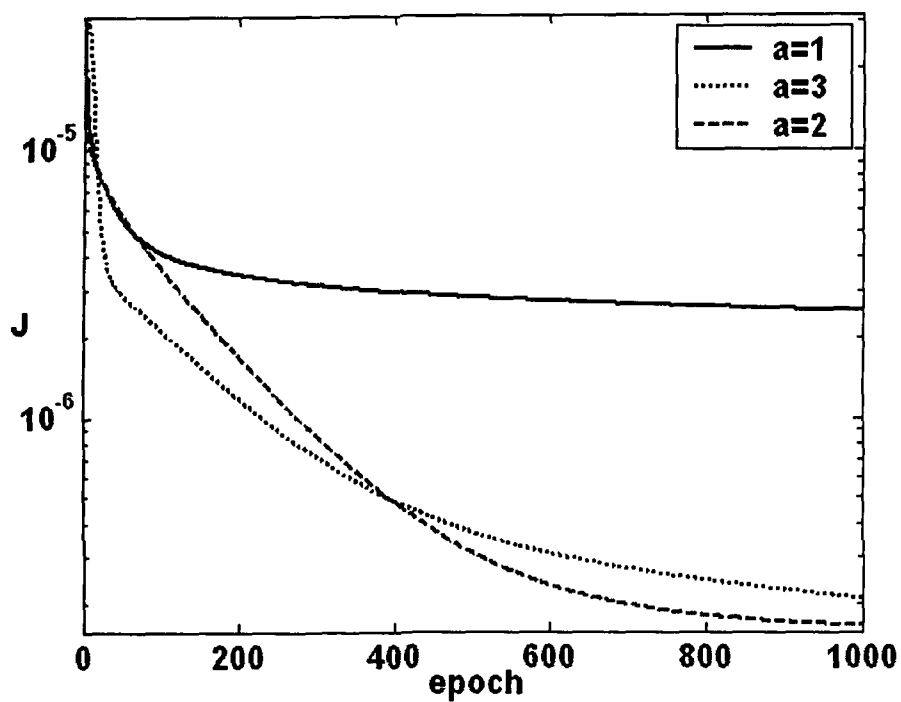


Fig. 3.29. Learning pattern of feed-forward network with MS-W neuron network with scaling factor  $a=1, 2$  &  $3$  for Example 5

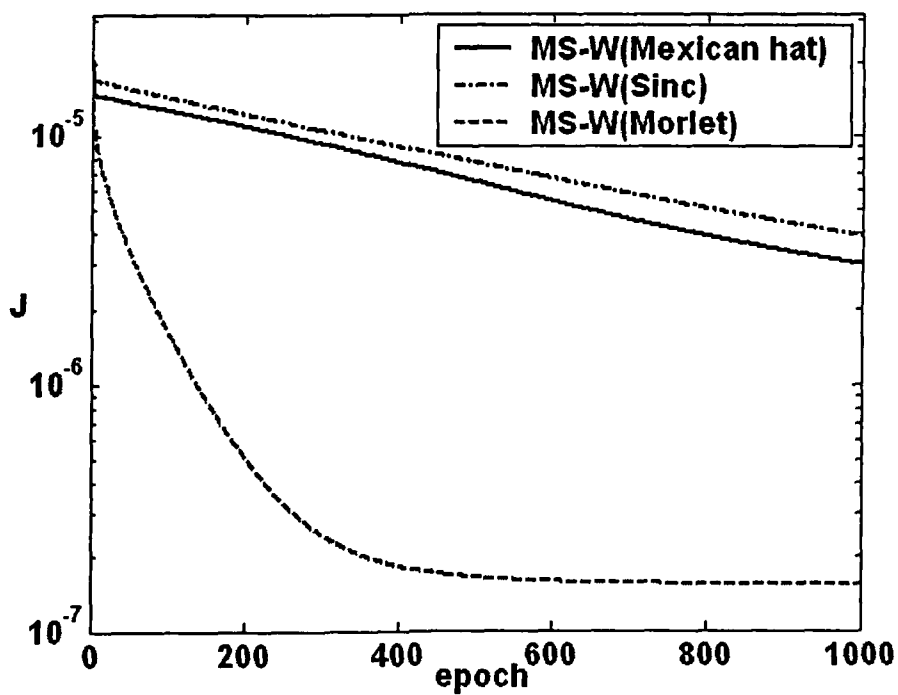


Fig. 3.30. Learning pattern of MS-W neuron network with all wavelet functions for Example 5

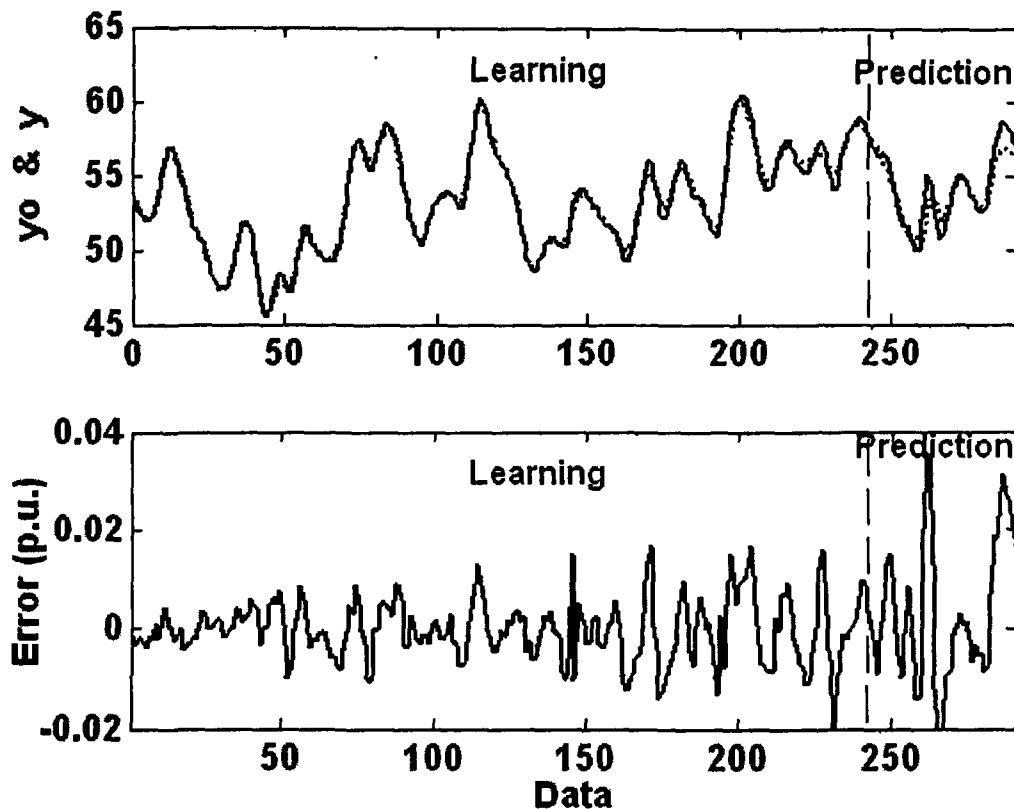


Fig. 3.31. Actual output & output of MS-W (Morlet) neuron network and the error for Example 5

### Revisited Example 6: *Human Operation at a Chemical Plant*

Figure 3.32 shows the structure determination of the SS-W neuron network for scaling factor ' $\alpha=1, 2$  and  $3$ '. From Fig. 3.32 and 3.37, scaling factor ' $\alpha=3$ ' is selected. The SS-W model is analyzed for three types of wavelet and the result is presented in Fig. 3.33 and Table 3.6. It illustrates that Sinc wavelet function is better among different types of the WAF with performance index  $J=8.1898 \times 10^{-6}$ . Initialization of the learning parameters  $W$ ,  $C_S$  and  $C_W$  for all wavelet functions and the learning parameters for Mexican hat, Morlet and Sinc wavelet function corresponding to the learned MS-W neuron network are as follows.

Initialization of the learning parameters:

$$W = \begin{bmatrix} 0.6273 \\ 0.6990 \\ 0.3971 \\ 0.4136 \\ 0.6552 \\ 0.8375 \end{bmatrix} \quad C_s = \begin{bmatrix} 0.7035 & 0.5667 \\ 0.4849 & 0.8230 \\ 0.1146 & 0.6739 \\ 0.6648 & 0.9994 \\ 0.3653 & 0.9616 \\ 0.1400 & 0.0588 \end{bmatrix} \quad C_w = \begin{bmatrix} 0.3716 & 0.7764 \\ 0.4252 & 0.4893 \\ 0.5946 & 0.1859 \\ 0.5657 & 0.7006 \\ 0.7165 & 0.9827 \\ 0.5113 & 0.8066 \end{bmatrix}$$

For SS-W neuron network with Mexican hat wavelet function:

$$W^f = \begin{bmatrix} 0.6597 \\ 1.3802 \\ 0.2992 \\ 0.3615 \\ 0.3513 \\ 0.7515 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.6509 & 0.7900 \\ 0.4109 & 1.1331 \\ 0.7211 & 0.9197 \\ 0.7308 & 0.1477 \\ 0.2305 & 0.2504 \\ 0.1848 & 0.9317 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.8098 & 0.5226 \\ 0.0132 & 1.0989 \\ 0.3258 & 0.4514 \\ 0.1742 & 0.7838 \\ 0.3722 & 0.8198 \\ 0.0117 & 0.2220 \end{bmatrix}$$

For SS-W neuron network with Morlet wavelet function:

$$W^f = \begin{bmatrix} -0.0321 \\ -0.1086 \\ 0.8564 \\ 0.0344 \\ 0.8818 \\ 0.2789 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.4881 & 0.4232 \\ 0.9926 & 0.6592 \\ 0.3732 & 0.6998 \\ 0.5313 & 0.9587 \\ 0.1812 & 0.2151 \\ 0.5019 & 0.1214 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.4194 & 0.4511 \\ 0.2129 & 0.9437 \\ 0.0352 & 0.3523 \\ 0.0811 & 0.8552 \\ 0.8501 & 0.9246 \\ 0.3402 & 0.8864 \end{bmatrix}$$

For SS-W neuron network with Sinc wavelet function:

$$W^f = \begin{bmatrix} 0.6629 \\ -0.3057 \\ 0.9393 \\ -0.9553 \\ 0.9507 \\ -0.0336 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.5588 & 0.2374 \\ 0.4870 & 0.8123 \\ 0.3973 & 0.9421 \\ 0.4930 & 0.1563 \\ 0.9685 & 1.0416 \\ 0.8270 & 0.1172 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.8016 & 0.6799 \\ 0.4330 & 0.3736 \\ 0.2408 & 1.1871 \\ 0.6348 & 0.3802 \\ 0.3667 & 1.3521 \\ 0.4314 & 0.2705 \end{bmatrix}$$

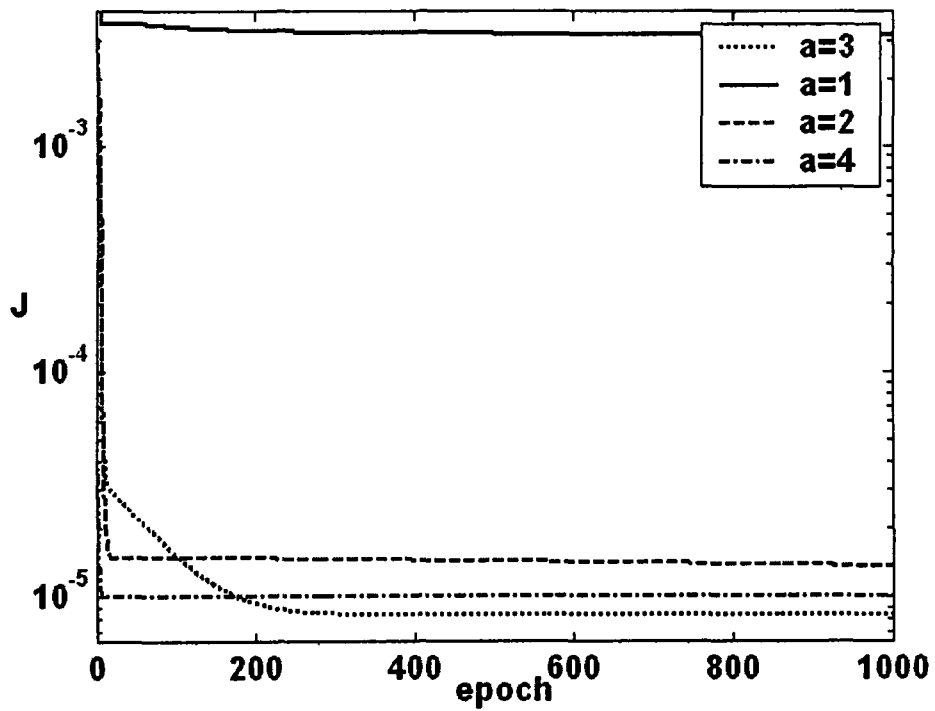


Fig. 3.32. Learning pattern of feed-forward network with SS-W neuron network using Morlet activation function with scaling factor  $a=1, 2$  &  $3$  for Example 6

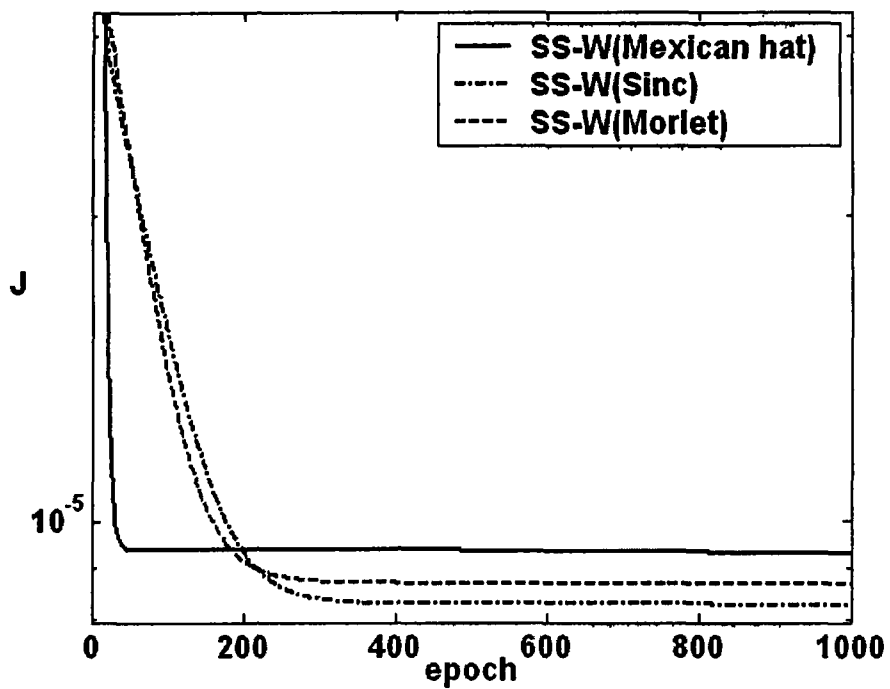


Fig. 3.33. Learning pattern of SS-W neuron network with all wavelet functions for Example 6

Figure 3.34 shows the structure determination of the MS-W neuron network for scaling factor ' $\alpha=1, 2, 3$  and  $4$ '. From this figure and Fig. 3.38, scaling factor ' $\alpha=3$ ' is selected. Figure 3.35 shows the learning pattern for different wavelet function. Mexican hat wavelet function yields better result with performance index  $J= 6.9712 \times 10^{-6}$ . Initialization of the learning parameters for all wavelet functions is the same as SS-W neuron network. Learning parameters for Mexican hat, Morlet and Sinc wavelet function corresponding to the learned MS-W neuron network are as follows.

For MS-W neuron network with Mexican hat wavelet function:

$$W^f = \begin{bmatrix} 0.8574 \\ 0.0283 \\ 0.7642 \\ -0.4525 \\ -0.0369 \\ 0.7375 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.4001 & 0.4904 \\ 0.1544 & 0.8003 \\ 0.0644 & 0.8871 \\ 0.1204 & 0.3581 \\ 0.0293 & 0.8250 \\ 0.8650 & 0.4208 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.2342 & 0.9645 \\ 0.1279 & 0.3487 \\ 0.4124 & 1.0744 \\ 0.6348 & 0.7235 \\ 0.1815 & 0.4005 \\ 0.4031 & 0.2550 \end{bmatrix}$$

For MS-W neuron network with Morlet wavelet function:

$$W^f = \begin{bmatrix} 0.1239 \\ 0.6012 \\ 0.4934 \\ 0.3015 \\ 0.9614 \\ 0.9189 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.7035 & 0.5486 \\ 0.4848 & 0.8584 \\ 0.1146 & 0.6605 \\ 0.6648 & 1.0156 \\ 0.3653 & 1.0146 \\ 0.1401 & 0.0199 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.3717 & 0.7282 \\ 0.4253 & 0.2667 \\ 0.5944 & 0.3501 \\ 0.5658 & 0.6061 \\ 0.7161 & 1.1179 \\ 0.5112 & 0.9401 \end{bmatrix}$$

For MS-W neuron network with Sinc wavelet function:

$$W^f = \begin{bmatrix} 0.8472 \\ 0.6839 \\ 0.4283 \\ -0.4333 \\ -0.0034 \\ 0.8511 \end{bmatrix} \quad C_s^f = \begin{bmatrix} 0.6508 & 0.8391 \\ 0.4108 & 1.0945 \\ 0.7211 & 0.9377 \\ 0.7309 & 0.0715 \\ 0.2305 & 0.2547 \\ 0.1847 & 0.9729 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.8095 & 0.9633 \\ 0.0133 & 0.5405 \\ 0.3257 & 0.5015 \\ 0.1743 & 0.6810 \\ 0.3722 & 0.7700 \\ 0.0116 & 0.2802 \end{bmatrix}$$

As shown in Table 3.6, MS-W neuron network has best performance index. Figure 3.36 shows system output and the output of the MS-W neuron network with the error between them.

Table 3.6: Performance index ( $J$ ) with different networks and wavelet functions for Example 6

Model →	SS-W	MS-W	WNN	LWNN	NN
H.N. →	12	12	15	15	20
Mexican hat	$9.277 \times 10^{-6}$	$6.971 \times 10^{-6}$	$8.668 \times 10^{-6}$	$8.360 \times 10^{-6}$	$1.096 \times 10^{-5}$
Morlet	$8.674 \times 10^{-6}$	$7.479 \times 10^{-6}$	$8.228 \times 10^{-6}$	$8.506 \times 10^{-6}$	
Sinc	$8.189 \times 10^{-6}$	$8.739 \times 10^{-6}$	$2.622 \times 10^{-5}$	$1.129 \times 10^{-5}$	

Figures 3.36 and 3.37 show the change of performance index with increasing scaling factor 'a' for all examples with SS-W & MS-W neuron model, respectively. The number of hidden neuron is increased if the performance index improved.

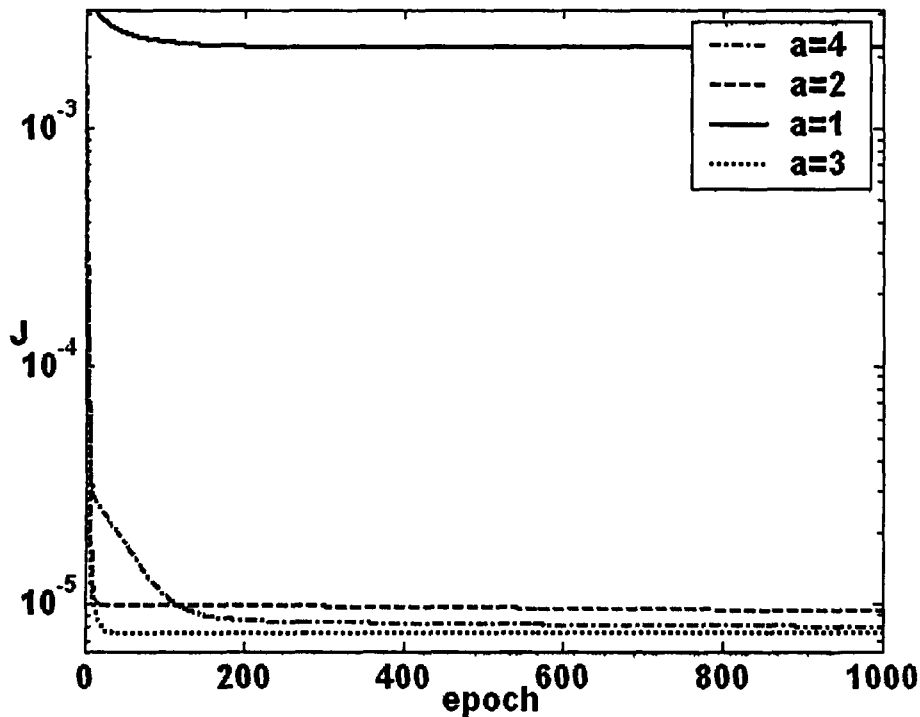


Fig. 3.34. Learning pattern of feed-forward network with MS-W neuron network with scaling factor  $a=1, 2, 3$  &  $4$  for Example 6

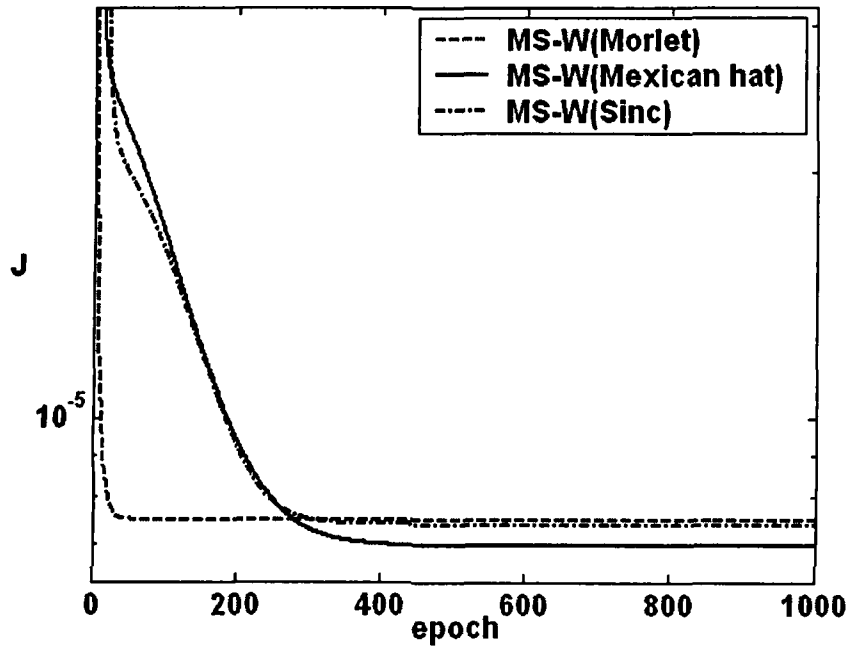


Fig. 3.35. Learning pattern of MS-W neuron network with all wavelet functions for Example 6

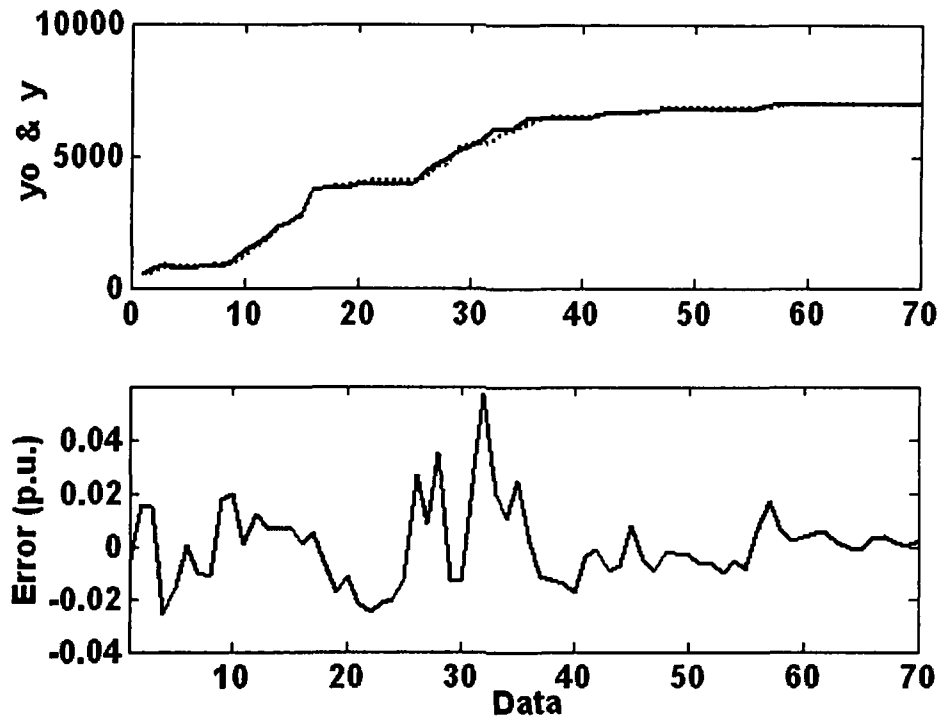


Fig. 3.36. Actual output & output of MS-W (Mexican hat) neuron network and the error for Example 6



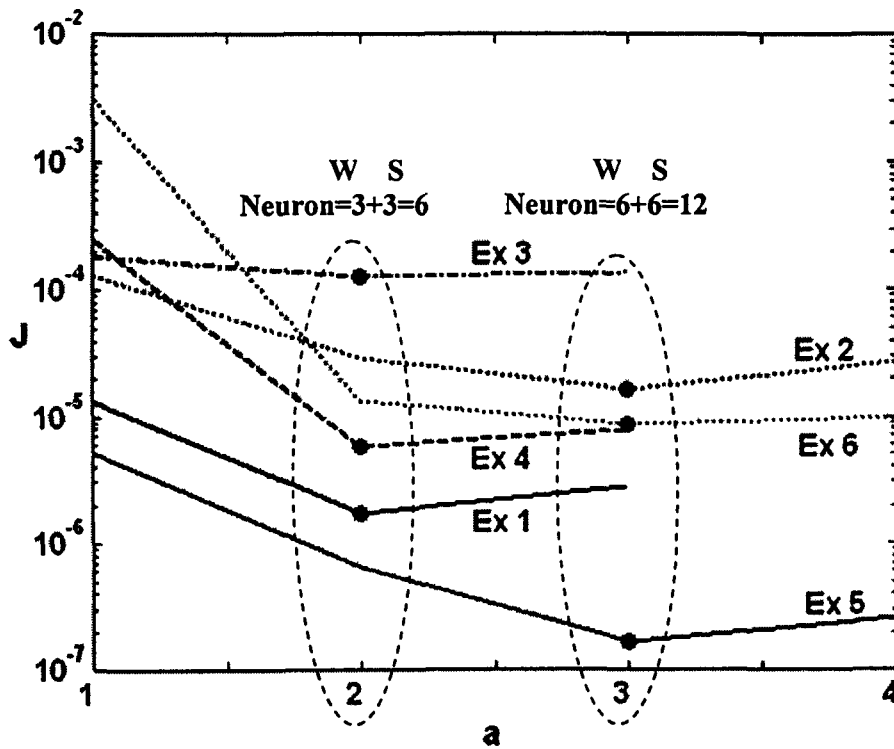


Fig. 3.37. Performance index of feed-forward SS-W neuron network with different scaling factor 'a' for all examples

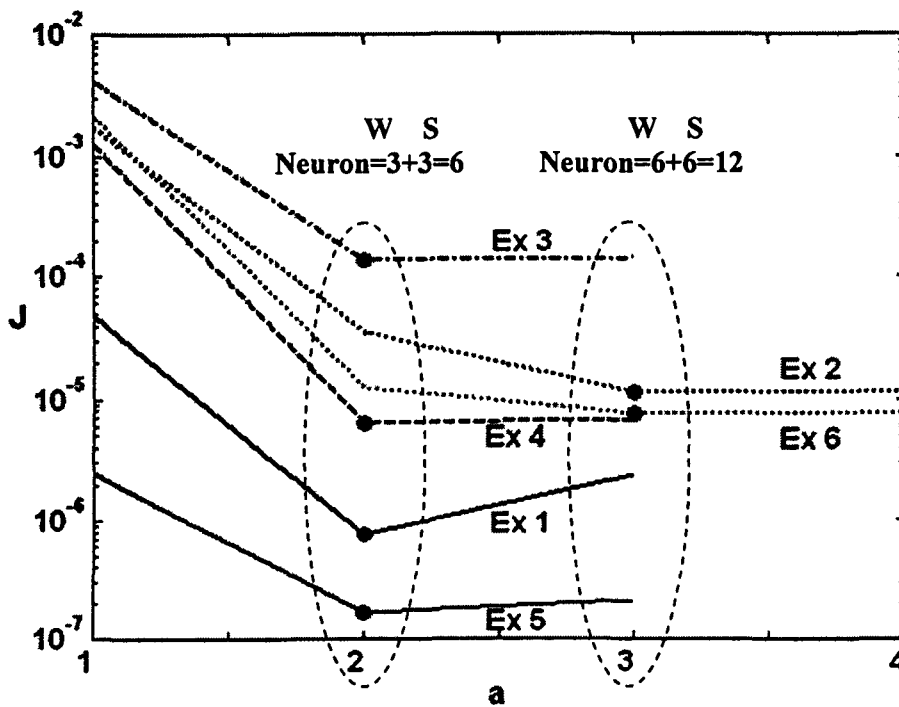


Fig. 3.38. Performance index of feed-forward MS-W neuron network with different scaling factor 'a' for all examples

### **3-8 Conclusions**

In this chapter, S-W neuron network has been proposed. Each S-W neuron network is a combination of SAF and WAF. The S-W neuron networks have advantage over either SAF or WAF separately applied to feed-forward networks. The proposed neuron networks are used in the hidden layer of a standard single hidden layered feed-forward network. Their performances are evaluated by modeling of dynamic system. They have been tested on six different examples.

Three types of wavelet activation functions, namely Mexican, Morlet and Sinc are tested in the S-W neuron network. The comparative results of different wavelets show that Morlet activation function yields better performance in either SS-W or MS-W neuron networks.

The proposed SS-W or MS-W neuron networks have better performance than WNN network with WAF only and NN with SAF only, even with fewer number of hidden layer neurons. The S-W neuron networks have better performance in comparing to LWNN network. MS-W neuron network yields better performance in comparison to SS-W neuron network.

# Chapter 4

## Neuro-Fuzzy Model

### 4-1 Introduction

Control theory deals with the analysis and synthesis of dynamical systems in which one or more variables are kept within prescribed limits. Many real world applications need to describe models for unknown systems [Narendra'90]. In the past few decades, system modeling and identification attracted the attention of a considerable number of researchers [Narendra'90, Qin'92, Mastorocostas'02, Xu'87, Sugeno'93, Takagi'85, Azeem'00b, Lee'00], the reason is its extensive application in practical life.

System identification plays a principal role In Input-Output data analysis, such that a better result can be obtained from better model. System identification includes two parts: Structure identification and parameter identification. In structure identification, input variables and input-output relations are found. In parameter identification, the parameters of the model are adjusted by optimizing a performance index [Narendra'90, Sugeno'93].

The Parallel (P) and the Series-Parallel (S-P) configurations are the two common methods to identify parameters for the unknown model of dynamic systems [Narendra'90, Qin'92- Bernieri'94]. In Series-Parallel configuration, the output of the system (plant) is fed into

the model. Since there is no feedback of the model output to itself, a static learning algorithm is applied. In this configuration, the parameter learning will converge if the outputs are bounded for bounded inputs [Narendra'90].

In the Parallel configuration, the output of the model is fed back as inputs to the model. Identification using parallel configuration, the model feedback introduces dynamics to the model; but it can learn the system dynamics without assuming much knowledge about the structure of the system under consideration [Qin'92]. This model is suitable for long-term and multi-step prediction in forecasting problem. When information about system is less, this configuration is better; however, the learning convergence is not guaranteed [Narendra'90]. Since the output of the model can be carried out on-line, the Parallel configuration can be used for on-line learning approach [Bernieri'94].

Recently fuzzy system identification has attracted the researchers involved with systems modeling [Jang'93, Sugeno'93, Yager'94, Gebhardt'94, Wu'00, Azeem'03a, Klir'03]. In describing the behavior of many complex and ill-defined systems, precise mathematical models may fail to give satisfactory results. In such cases, fuzzy models are used to reflect the uncertainty of the systems in a proper way. Takagi and Sugeno introduced Takagi-Sugeno-Kang (TSK) fuzzy model [Takagi'85, Sugeno'88]. The basic idea in this approach is to decompose the complicated input space into subspaces and then approximate the system in each subspace by a linear/non-linear regression model called local model. The resulting fuzzy model is the aggregation of these local models. Later Shing and Jang proposed Adaptive Neuro-Fuzzy Inference System (ANFIS) as a powerful method for mapping input-output system modeling based on fuzzy inference system [Jang'93, Nauck'97].

In these application models, it is possible to use both parallel and parallel-series

configuration for estimation of unknown parameters of the model. The present work proposes an implementation of combining parallel and series-parallel configuration on TSK fuzzy model. It has advantages over both parallel and series-parallel configuration. Premise and consequent part of the rules in TSK models are learned by parallel and series-parallel configuration respectively, or vice versa. If the output of plant is feedback to premise part and output of the model feedback to consequent part, it results a Premise Series-Parallel (PS-P) configuration. In the same way if model output feedback to premise part and the plant output feedback to consequent part, we have a Consequent Series-Parallel (CS-P) configuration. Therefore in this way the advantage of both configuration, i.e. tracking the real output of the plant by series-parallel configuration and long-term or multi step prediction with less knowledge about the plant by parallel configuration are exploited together. Consequently, we obtain the best model that follow real output for long time prediction with less knowledge about the plant.

A lot of learning algorithm has been developed for recurrent models. Two specific algorithm that are based on GD are Back-Propagation Through Time (BPTT) [Rumelhart'86-Werbos'88] and Real Time Recurrent Learning (RTRL) [Williams'89]. However, these algorithms have two main problems: stability and slow rate of the convergence during learning procedure. The problem is that for stability, learning rate should be small but when that is small the speed of the convergence become low. To eradicate these two discrepancies large number of studied has been done for improving the speed of convergence [Wu'00, Yu'95a, Barbounis'06] in addition to incorporating the stability [Yu'01a, Yu'01b, Chen'94, Wang'97, Yu'95b, Li'06, Jin'99, Yi'06] to the parameter learning procedure. In [Yu'01b], the passivity theory has been applied to analyze the stability of the dynamic neural network for identification problem. Yi, et al., [Yi'06] carried out a comparative study for output convergence [Yi'01, Liu'04, Li'04] and

the state convergence [Cao'03 a, Cao'03 b, Forti'94, Forti'95, Liang'01] of a recurrent neural network. Yu, et al. [Yu'95b] have shown that the neuro-fuzzy model under certain condition is stable by applying the Lyapunov stability theorem and passivity theory. However, they have not ascertained any boundary for learning parameter. Based on Lyapunov-Krasovskii functional method, Li and Liao [Li'06] have proposed a robust learning algorithm for recurrent neural network under noise disturbance while Chen and Jain [Chen'94] have proposed a robust BP algorithm and shown that by improving the learning rate the algorithm is stable under noise effect. Wang et al. [Wang'97] have introduced a robust and fast learning algorithm for B-spline membership function using robust objective function and gradient descent method. Using Lyapunov stability theorem a mathematical way to calculate the upper bound of the learning rate for recurrent wavelet neural network [Yoo'06] and a mamdani fuzzy model [Lee'00], based on the parameter of the network, have been introduced, respectively. Azeem, et al., [Azeem'03a] used an easy and understandable way for adaptive learning rate to increase speed of convergence rate. In this chapter, presented studies guarantee the stability and the speed of the learning procedure by applying the Lyapunov stability theorem and the adaptive learning rate to the learning procedure of neuro-fuzzy model.

The chapter spread over six sections: Brief discussion about neuro-fuzzy model is given in section 4-2. In section 4-3, parameter identification configurations are devised. Section 4-4 deals with the learning algorithm and convergence analysis of S-W neuron models. Section 4-5 consists of simulation results and discussions. Finally, the conclusions are relegated to section 4-6.

## 4-2 Neuro-Fuzzy model

Each rule of a fuzzy model based on TSK fuzzy model mapped the input space  $A^m \subset R^n$  to a linear function in the output space  $w^m \subset R$ , and has the form:

$$R^m : \text{if } x_1 \text{ is } A_1^m \wedge x_2 \text{ is } A_2^m \wedge \dots \wedge x_n \text{ is } A_n^m \text{ then } y \text{ is } w^m(x) \quad (4.1)$$

with  $m=1\dots M$ ,  $M$  being the number of rules. Each rule is premised on its own input vector  $X \in \mathcal{R}^n$ ,  $A_i^m$  is linguistic labels of fuzzy sets describing the qualitative nature of the input variable  $x_i$ ,  $\wedge$  and is a fuzzy conjunction operator (usually of T-norm).

The TSK model was introduced in [Takagi'85, Sugeno'88] as a hybrid model, which integrates the fuzzy conditions in the input space with the functional relationships in the output space. TSK-model has a linear or nonlinear relationship of inputs  $w^m(x)$  in the output space. Rules of TSK model are in the following form:

$$R^m : \text{if } \mathbf{x} \text{ is } A^m \text{ then } y \text{ is } w^m(x) \quad (4.2)$$

A linear form of  $w^m(x)$  in (4.1 & 4.2) is as follows:

$$w^m(X) = w_0^m + w_1^m x_1 + \dots + w_n^m x_n \quad (4.3)$$

where,  $w^m(x)$  defines a locally valid model on the support of the Cartesian product of fuzzy sets constituting the premise parts. The normalized firing strength for the normalized calculation or non-normalized firing strength for the non-normalized calculation is then multiplied with the output function  $w^m(x)$ . By taking Gaussian membership function and equal number of fuzzy sets to rules with respect to the inputs, firing strength of rules (4.2) is written as:

$$\mu_{A^m}(x) = \prod_{i=1}^n \exp\left(-\left(\frac{x_i - \bar{x}_{mi}}{\sigma_{mi}}\right)^2\right) \quad (4.4)$$

where  $\bar{x}_{mi}$  and  $\sigma_{mi}$  are the center and the standard deviation of the Gaussian membership functions. Applying T-norm (product operator) of the membership functions of the premise parts of the rule and the weighted average gravity method for defuzzification, the output of the TSK model is defined as:

$$\hat{Y} = \frac{\sum_{m=1}^M \mu_{A^m}(x) \cdot w^m(x)}{\sum_{m=1}^M \mu_{A^m}(x)} \quad (4.5)$$

The functionally equivalent neuro-fuzzy model of TSK model is shown in Fig. 4.1. In the following description,  $u'_j$  denotes the input to the  $j^{\text{th}}$  node in the  $l^{\text{th}}$  layer;  $O'_j$  denotes the  $j^{\text{th}}$  node output in layer  $l$ .

**Layer 1:** Nodes in layer 1 represent input variable. Every node accepts input values and transmits it to the next layer.

$$O_m^1 = u_m^1 = x_i \quad (4.6)$$

**Layer 2:** Nodes in this layer represent the terms of the respective linguistic variables. Every node operates on incoming signal with Gaussian membership function expressed by (4.7).

The parameters to be learned in this layer are  $\bar{x}_{mi}$  and  $\sigma_{mi}$ . Corresponding to each rule the learning parameter are expressed in vector form as  $\bar{x}_m = \{\bar{x}_{m1}, \bar{x}_{m2}, \dots, \bar{x}_{mn}\}$  and

$$\sigma_m = \{\sigma_{m1}, \sigma_{m2}, \dots, \sigma_{mn}\}.$$



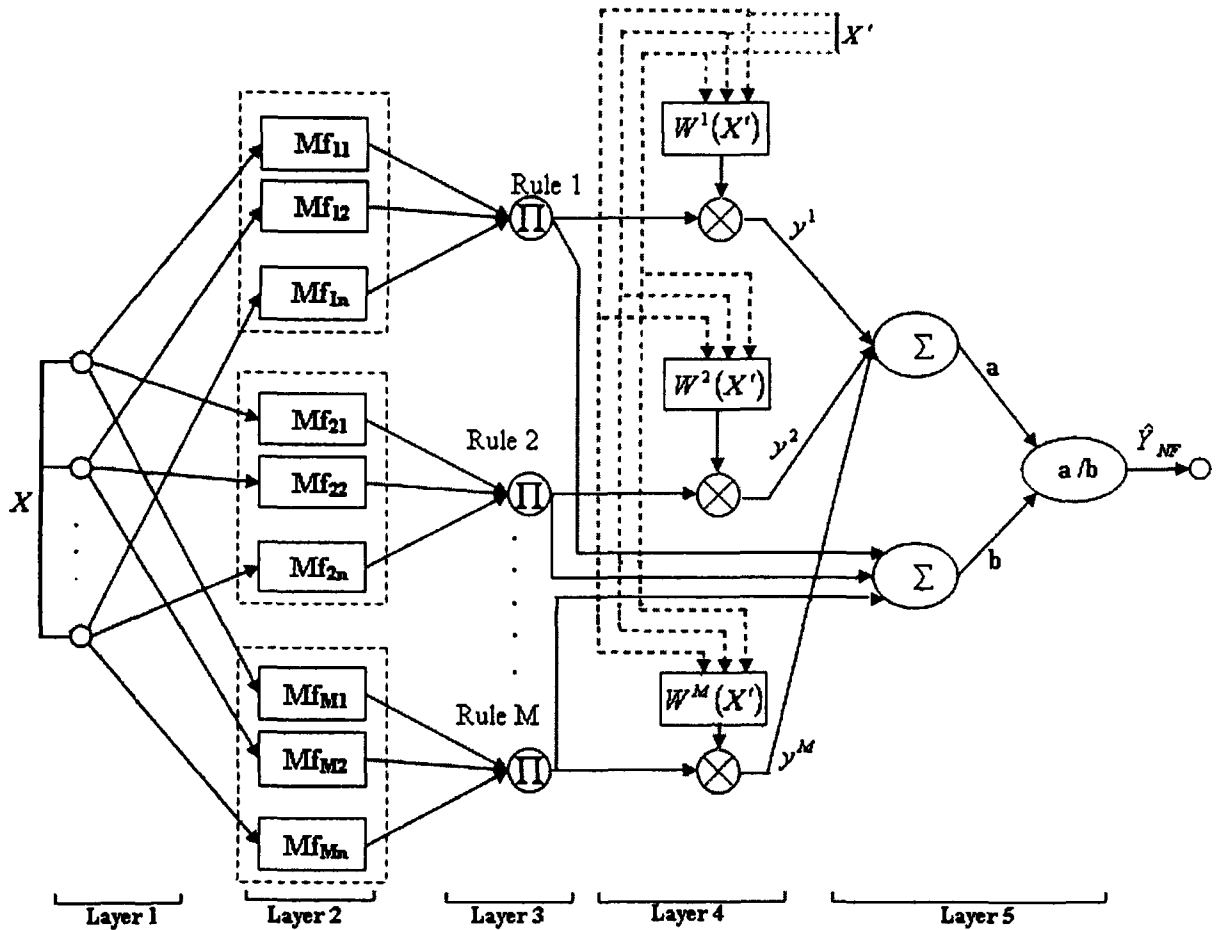


Fig. 4.1. Neuro-Fuzzy model

$$O_{mi}^2 = \mu_{A_i^m}(\mathbf{X}) = \exp\left\{-\frac{(x_i - \bar{x}_{mi})^2}{(\sigma_{mi})^2}\right\} \quad (4.7)$$

Layer 3: Each node in layer 3 represents a fuzzy rule. The output from the nodes in layer 2, specified for a fuzzy rule, is being input to the nodes, specified for that rule, in layer 3. The output of each node in layer 3 is the product of all inputs; it represents the firing strength of that rule. Thus, the firing strength of the  $m^{\text{th}}$  rule is specified as (4.4, 4.8).

$$O_m^3 = \prod_i O_{mi}^2 = \mu_{A^m}(\mathbf{X}) \quad (4.8)$$

Layer 4: Nodes in layer 4 are called consequent nodes. Two inputs are applied to each node in this layer, namely the output from the layer 3 node and the output from its corresponding local model approximated by (4.3). The output of each node is the product of both input and given by (4.9). Where  $X'$  is input for local model either from the system or from the model depending upon the configuration.

$$O_m^4 = y^m = w^m(\mathbf{X}') \cdot O_m^3 = w^m(\mathbf{X}') \cdot \mu_{A^m}(\mathbf{X}) \quad (4.9)$$

Layer 5: Three nodes in this layer constitute the aggregation and defuzzification of fuzzy rules.

The output of all nodes from layer 4 is the input to the first node and its output is the sum of all inputs and expressed as (4.10). The output of all nodes from layer 3 is the input to the second node and its output is the sum of all inputs and expressed as (4.11). Inputs to the third node in this layer are the output from first and second node. The output of the third node in the ratio of these two inputs is given in (4.12).

$$a = \sum_{m=1}^M O_m^4 = \sum_{m=1}^M (w^m(\mathbf{X}') \cdot \mu_{A^m}(\mathbf{X})) \quad (4.10)$$

$$b = \sum_{m=1}^M O_m^3 = \sum_{m=1}^M \mu_{A^m}(\mathbf{X}) \quad (4.11)$$

$$O_j^5 = \hat{Y}_{NF} = \frac{a}{b} \quad (4.12)$$

### 4-3 Configurations for Parameter Identification

The problem of identification consists of selecting a suitable model and algorithm for learning its parameter. In this section, two well-known parameter identification methods, series-

parallel and parallel configurations are discussed to optimize the learning parameter of the neuro-fuzzy model. Two new configurations, which are combination of series-parallel and parallel configurations, applicable to TSK model are proposed.

### 4-3.1 Parallel Configuration

Parallel configuration for system identification is shown in Fig. 4.2. A Linear/nonlinear dynamic system models may be represented by mapping from the input space to the output space, which we call as function approximation. To construct a neuro-fuzzy model for a Multi-Input and Single-Output (MISO) system using parallel configuration, consider a Non-linear Auto-Regressive Moving Average (NARMA) model representing a MISO system.

$$\hat{y}(t) = f \left( \begin{matrix} u_1(t), \dots, u_1(t - \tau_{i1}), u_2(t), \dots, u_r(t - \tau_{ir}), \\ \hat{y}(t-1), \hat{y}(t-2), \dots, \hat{y}(t - \tau_o) \end{matrix} \right) \quad (4.13)$$

where  $u_q; (q=1, \dots, r)$  and  $\hat{y}$  denote the inputs and model outputs respectively.  $\tau_{iq}$  and  $\tau_o$  are the corresponding delays. Function 'f' in (4.13) may be linear or non-linear. Here it is supposed to be a neuro-fuzzy model. The output of model in parallel configuration is a function of the past output of the model as well as input delays. The premises of the rules, which represent delays as well as the order of dynamic systems, for a NARMA model of a complex system, are denoted by:

$$\begin{aligned} X &= \{x_1, \dots, x_{(\tau_{i1} + \dots + \tau_{ir} + r)}, x_{(\tau_{i1} + \dots + \tau_{ir} + r + 1)}, \dots, x_n\} \\ &= \{u_1(t), \dots, u_1(t - \tau_{i1}), u_2(t), \dots, u_r(t - \tau_{ir}), \hat{y}(t-1), \dots, \hat{y}(t - \tau_o)\} \end{aligned} \quad (4.14)$$

where  $n = \tau_{i1} + \dots + \tau_{ir} + \tau_o + r$

In Parallel configuration, input to the consequent part is  $X' = X$ .

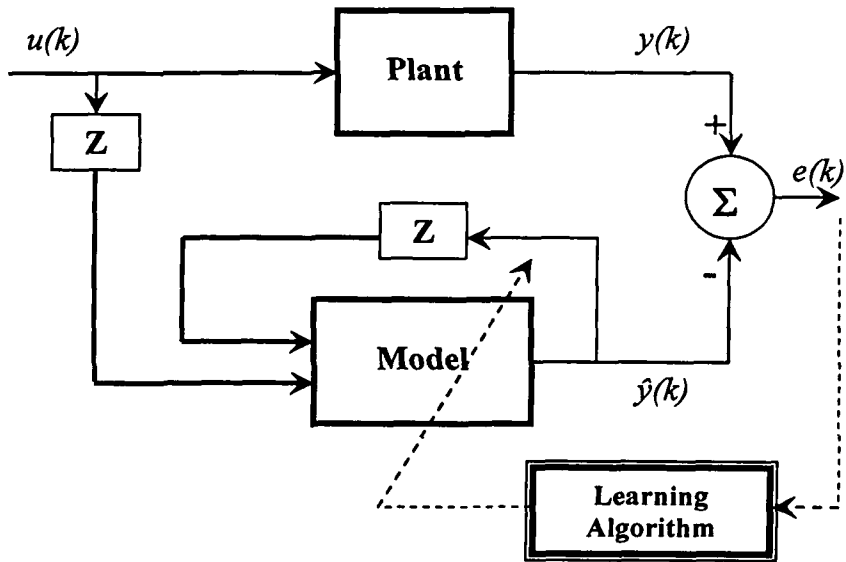


Fig. 4.2. Parallel configuration

### 4-3.2 Series-Parallel configuration

In the series-parallel configuration, output of the model is a function of the input delays and past values of the plant output as shown in Fig. 4.3. Plant is a system that should be identified with neuro-fuzzy model.

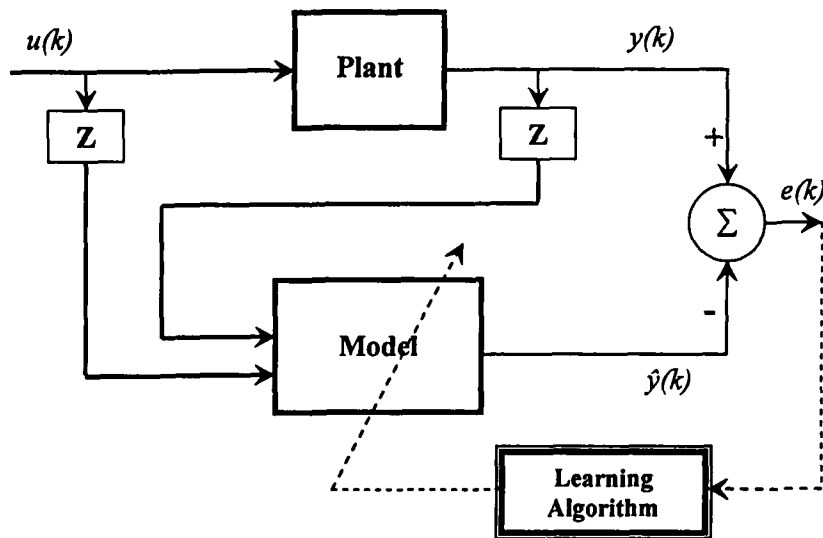


Fig. 4.3. Series-parallel configuration

We shall assume that output of unknown model in series-parallel configuration, which should be identified is as follows:

$$\hat{y}(t) = g \left( \begin{matrix} u_1(t), \dots, u_1(t - \tau_{i1}), u_2(t), \dots, u_r(t - \tau_{ir}), \\ y(t-1), y(t-2), \dots, y(t - \tau_o) \end{matrix} \right) \quad (4.15)$$

Function 'g' in (4.15), represented by neuro-fuzzy model. Where  $u_q; (q=1, \dots, r)$  and  $y$  denote the inputs and model outputs, respectively.  $\tau_{iq}$  and  $\tau_o$  are the corresponding delays. The premise inputs in this configuration are denoted by:

$$\begin{aligned} X &= \{x_1, \dots, x_{(\tau_{i1} + \dots + \tau_{ir} + r)}, x_{(\tau_{i1} + \dots + \tau_{ir} + r + 1)}, \dots, x_n\} \\ &= \{u_1(t), \dots, u_1(t - \tau_{i1}), u_2(t), \dots, u_r(t - \tau_{ir}), y(t-1), \dots, y(t - \tau_o)\} \end{aligned} \quad (4.16)$$

where  $n = \tau_{i1} + \dots + \tau_{ir} + \tau_o + r$

In S-P configuration, input to the consequent part is  $X' = X$ .

### 4-3.3 Proposed Configurations

In the proposed configuration, output of the model depends upon the system history as well as the present and past output of the model as shown in the Fig. 4.4. System history means present and past input and output of the system. Our objective is that the model should track the actual system output. It means that the error between plant and model output decrease and results in an improved performance of the model. In S-P configuration, the main problem is selecting a model from a class of models and its structure determination. After selecting the model and deciding about its structure, the problem is reduced to parameter learning of the model. One important problem in S-P configuration is that model output is of no use during learning procedure except calculating the error. By using output of the model to learn the learning

parameter the significance of the model in learning procedure can be acquired. Parallel configuration has advantage that without much information about the system, it can learn the parameter of the model. In this configuration, the model output tracks the plant output by minimizing the error between them. If the convergence of the learning procedure is guaranteed the parallel configuration, is most suitable for long-term prediction.

Since, a neuro-fuzzy model is divided into two parts; i.e. premise and consequent parts, either the system output feedback to the premise part and the model output to the consequent part or vice versa. If the output of the system is feedback to the premise part and the output of the model feedback to the consequent part of the neuro-fuzzy system, it results a Premise Series-Parallel (PS-P) configuration. In the same way, if the output of the model feedback to the premise part, and the output of the plant feedback to the consequent part, we have a Consequent Series-Parallel (CS-P) configuration.

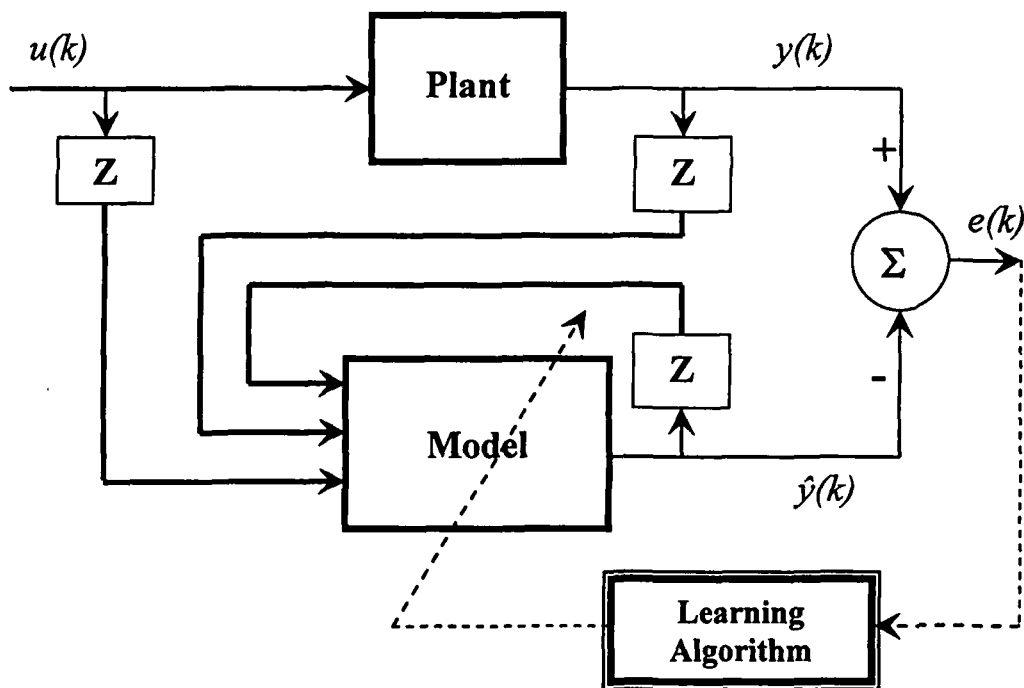


Fig. 4.4. Proposed parallel and Series-parallel configuration

This configurations can be used in special cases that model is combination of two part. Especially, as discussed before, fuzzy models are divided into two parts; premise and consequent parts. For the proposed configuration, the output of the model is written as:

$$\hat{y}(t) = h \left( \begin{array}{l} u_1(t), \dots, u_1(t - \tau_{i1}), u_2(t), \dots, u_r(t - \tau_{ir}), \\ y(t-1), y(t-2), \dots, y(t - \tau_o), \\ \hat{y}(t-1), \hat{y}(t-2), \dots, \hat{y}(t - \tau_o) \end{array} \right) \quad (4.17)$$

Function 'h' in (4.17) here is represented by neuro-fuzzy model. Where  $u_q; (q=1, \dots, r)$  denote the inputs.  $\hat{y}$  and  $y$  are model and system outputs, respectively.  $\tau_{iq}$  and  $\tau_o$  are the corresponding delays.

#### a) Consequent Series-Parallel configuration (CS-P)

In CS-P, the output of the model is feedback to the premise part as it is for parallel configuration and the output of the plant is feedback to the consequent part as it is for series-parallel configuration, which is shown in Fig. 4.5. It is a well-established fact that the premise part of each rule in fuzzy models exemplifies a local region in the input space in which consequent part act as a local model for the output space [Sugeno'93, Takagi'85- Sugeno'88, Zeng'94- Zeng'95]. These local models in the output space are approximated by linear or non-linear function of the premise variables. In CS-P configuration, the plant inputs and output with its delays are employed to approximate the local models in the consequent part of the TSK model, whereas the inputs and the delays of the model output are utilized to comprehend the input space region. The premise inputs in this configuration are denoted by:

$$\begin{aligned} X &= \{x_1, \dots, x_{(\tau_{i1} + \dots + \tau_{ir} + r)}, x_{(\tau_{i1} + \dots + \tau_{ir} + r + 1)}, \dots, x_n\} \\ &= \{u_1(t), \dots, u_1(t - \tau_{i1}), u_2(t), \dots, u_r(t - \tau_{ir}), \hat{y}(t-1), \dots, \hat{y}(t - \tau_o)\} \end{aligned} \quad (4.18)$$

where input to the consequent part in CS-P configuration is:

$$\begin{aligned} \mathbf{X}' &= \{x_1, \dots, x_{(\tau_{i1} + \dots + \tau_{ir} + r)}, x_{(\tau_{i1} + \dots + \tau_{ir} + r + 1)}, \dots, x_n\} \\ &= \{u_1(t), \dots, u_1(t - \tau_{i1}), u_2(t), \dots, u_r(t - \tau_{ir}), y(t-1), \dots, y(t - \tau_o)\} \end{aligned} \quad (4.19)$$

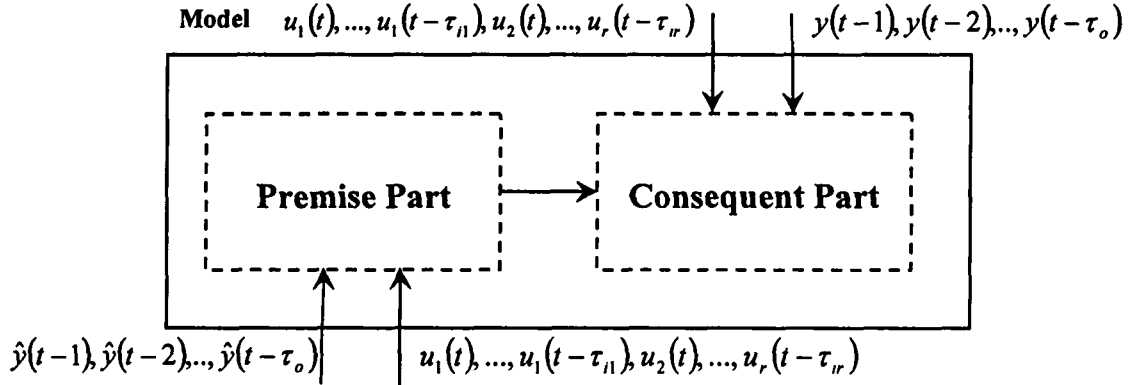


Fig. 4.5. Output of the plant feedback to the consequent part and the output of model feedback to the premise part

#### b) Premise Series-Parallel configuration (PS-P)

In PS-P, the output of the model feedback to the consequent part and the plant output is used for the premise part as shown in Fig. 4.6. In PS-P configuration, the model inputs and output with its delays are employed to approximate local models in the consequent part of the TSK model whereas the inputs and the plant output with their delays are utilized to comprehend the input space region. The premise inputs in this configuration are denoted by:

$$\begin{aligned} \mathbf{X} &= \{x_1, \dots, x_{(\tau_{i1} + \dots + \tau_{ir} + r)}, x_{(\tau_{i1} + \dots + \tau_{ir} + r + 1)}, \dots, x_n\} \\ &= \{u_1(t), \dots, u_1(t - \tau_{i1}), u_2(t), \dots, u_r(t - \tau_{ir}), y(t-1), \dots, y(t - \tau_o)\} \end{aligned} \quad (4.20)$$

and input to the consequent part in PS-P configuration is

$$\begin{aligned} \mathbf{X}' &= \{x_1, \dots, x_{(\tau_{i1} + \dots + \tau_{ir} + r)}, x_{(\tau_{i1} + \dots + \tau_{ir} + r + 1)}, \dots, x_n\} \\ &= \{u_1(t), \dots, u_1(t - \tau_{i1}), u_2(t), \dots, u_r(t - \tau_{ir}), \hat{y}(t-1), \dots, \hat{y}(t - \tau_o)\} \end{aligned} \quad (4.21)$$



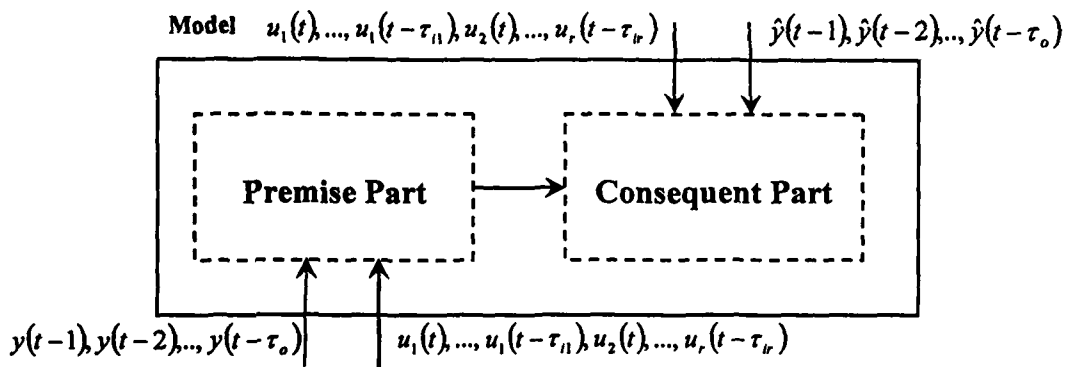


Fig. 4.6. Output of the plant feedback to the premise and the output of model feedback to the consequent part

## 4-4 Learning procedure

In this section, structure determination and initialization of the neuro-fuzzy model are presented. Discussion of different configurations in parameter learning of the neuro-fuzzy models also is including. In this section, an adaptive learning algorithm is introduced to learn the parameters of the model.

### 4-4.1 Structure determination and Initialization

Structure determination in neuro-fuzzy models means determination of the number of rules and input membership function. Initialization of the neuro-fuzzy models means that initializes center and standard deviation of membership function and initializes each linear function in consequent parts. In present work, Gaussian membership function is used. To determine number of necessary rules Modified Mountain Clustering (MMC) is applied [Azeem'03a, Yager'94, Chiu'96]. The purpose of clustering is to do natural grouping of large set of data, producing a concise representation of system's behavior. Azeem et.al., [Azeem'03a] have proposed a simple and easy way to implement, MMC for estimating the number and location of cluster centers. A brief discussion about MMC and its parameter is covered in Appendix B.

#### 4-4.2 Training

To adjust the learning parameter of the model, the performance index  $J$  as given in (2.18) is minimized by Gradient Descent (GD) algorithm. In this section, the GD based algorithm is applied. Since the parallel, CS-P and PS-P configurations include external recurrent to the model during learning procedure; criterion for learning stability and convergence has been evolved. To learn the parameters of the recurrent network, based on the gradient descent, different methods are presented in literature. All learning methods are the same as of back-propagation-through-time [Rumelhart'86- Werbos'88] or real-time recurrent learning algorithm [Williams'89] and it can be applied to adjust parameters of the recurrent network. In this work, by applying Lyapunov theorem, the learning stability and the convergence of learning procedure is guaranteed. To guarantee the speed of the convergence an adaptive learning rate with upper bound is applied.

##### a) Gradient Descent Technique of the parameters

For fine-tuning of initialized model/network parameters, a GD technique with momentum update and forgetting factor, as discussed in chapter 2, is applied to modify the parameters  $\bar{x}$ ,  $\sigma$  and  $W$  in (4.4 - 4.5). The parameter update formula for  $p^{\text{th}}$  data set is as follows:

$$\Delta_p W^m(q) = -\eta_w \cdot \frac{\partial J}{\partial W^m} = \eta_w \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial W^m} \right|_p \quad (4.23)$$

$$\Delta_p \sigma_{mi}(q) = -\eta_\sigma \cdot \frac{\partial J}{\partial \sigma_{mi}} = \eta_\sigma \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial \sigma_{mi}} \right|_p \quad (4.24)$$

$$\Delta_p \bar{x}_{mi}(q) = -\eta_{\bar{x}} \cdot \frac{\partial J}{\partial \bar{x}_{mi}} = \eta_{\bar{x}} \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial \bar{x}_{mi}} \right|_p \quad (4.25)$$

where  $e = y - \hat{y}$  is the error between the plant output and the model output. By applying the

chain rule to the above equation,  $\frac{\partial \hat{Y}_{NF}}{\partial w_m}$ ,  $\frac{\partial \hat{Y}_{NF}}{\partial \bar{x}_{m_i}}$  and  $\frac{\partial \hat{Y}_{NF}}{\partial \sigma_{m_i}}$  for different configurations are derived

as follows: Define  $\beta_m = \frac{\mu_{A^m}(X)}{\sum_{m=1}^M \mu_{A^m}(X)}$  then,

$$\frac{\partial \hat{Y}_{NF}}{\partial w_{m0}} = \beta_m \quad (4.26)$$

$$\frac{\partial \hat{Y}_{NF}}{\partial w_{m_i}} = x_i \cdot \beta_m \quad (4.27)$$

$$\frac{\partial \hat{Y}_{NF}}{\partial \bar{x}_{m_i}} = w_m(\mathbf{X}') \cdot \frac{\beta_m}{\mu_{A^m}} \cdot (1 - \beta_m) \cdot \frac{2 \cdot (x_i - \bar{x}_{m_i})}{\sigma_{m_i}^2} \quad (4.28)$$

$$\frac{\partial \hat{Y}_{NF}}{\partial \sigma_{m_i}} = w_m(\mathbf{X}') \cdot \frac{\beta_m}{\mu_{A^m}} \cdot (1 - \beta_m) \cdot \frac{2 \cdot (x_i - \bar{x}_{m_i})^2}{\sigma_{m_i}^3} \quad (4.29)$$

$X$  and  $X'$  in above equation are determined by (4.14) and (4.16) for P and S-P configurations, respectively. In CS-P configurations,  $X$  and  $X'$  are obtained by (4.18) and (4.19), respectively. With PS-P configuration, (4.20) and (4.21) are used to extract  $X$  and  $X'$ , respectively. Fig. 4.7 shows the learning algorithm for TSK Neuro-Fuzzy model with different configuration. Using performance indexes  $J$  as in (2.18) convergence theorem of the learning procedure is stated as follows:

### b) Learning Convergence theorems

Small value of learning rate  $\eta$  results in slower speed of convergence. Large value of  $\eta$  causes the learning procedure non-stable. Therefore learning rate should be chosen in such a way that the stability and convergence be guaranteed. To guarantee the stability during the learning procedure, Lyapunov stability theorem is applied. This formulates the appropriate range of

learning rate. Following Theorems guarantees the convergence stability of the neuro-fuzzy models:

Theorem 4.1: The asymptotic learning convergence of S-P and CS-P configurations (since local models have same variables i.e.  $\mathbf{X}$  ) are guaranteed if the learning rate for different learning parameters follows the upper bound as mentioned below:

$$0 < \eta_w < 2 \cdot P \cdot y_r^2 \quad (4.30)$$

$$0 < \eta_\sigma < \frac{2 \cdot P \cdot y_r^2}{\max_m |w(\mathbf{X})|^2 \cdot \left( \frac{2}{\sigma_{\min}^3} \right)^2} \quad (4.31)$$

$$0 < \eta_{\bar{x}} < \frac{2 \cdot P \cdot y_r^2}{\max_m |w(\mathbf{X})|^2 \cdot \left( \frac{2}{\sigma_{\min}^2} \right)^2} \quad (4.32)$$

Theorem 4.2: The asymptotic learning convergence of P and PS-P configurations (since local models have same variables i.e.  $\mathbf{X}'$  ) are guaranteed if the learning rate for different learning parameters follows the upper bound as mentioned below:

$$0 < \eta_w < 2 \cdot P \cdot y_r^2 \quad (4.33)$$

$$0 < \eta_\sigma < \frac{2 \cdot P \cdot y_r^2}{\max_m |w(\mathbf{X}')|^2 \cdot \left( \frac{2}{\sigma_{\min}^3} \right)^2} \quad (4.34)$$

$$0 < \eta_{\bar{x}} < \frac{2 \cdot P \cdot y_r^2}{\max_m |w(\mathbf{X}')|^2 \cdot \left( \frac{2}{\sigma_{\min}^2} \right)^2} \quad (4.35)$$

Stability analysis and convergence is carried out in Appendix C.

### c) Adaptive learning rate

The learning rate is adaptive with the lower and upper bounds as mentioned in the above stated Theorem. Whether the learning rate ( $\eta$ ) is increased or decreased, it depends on the change in the value of performance index  $J$ . A two-phase adaptive scheme, to make the learning rate adaptive, is used in the GD technique. The initial value of learning rate is kept at 0.1 for all applications. In the first phase either it increases or decreases by a factor of “10”. When it reaches within bounds, in a very few epochs (i.e.  $< 10$ ), then the second phase starts. This increase or decrease is dependent upon the acceptance or rejection, respectively, for updating the parameters. In the second phase, involving the operation  $\eta \leftarrow \gamma\eta$ ; we choose  $\gamma = 1.05$  for the acceptance of parameter updates and  $\gamma = 0.7$  for the rejection of the same. In the first phase, if the learning rate is continuously decreasing due to the rejection for update of the parameter, and the learning rate reaches within a bound, the update of the parameter is accepted. This acceptance forces the learning rate to increase according to the second phase. If the learning rate is continuously increasing, in the first phase due to the acceptance for update of the parameter, and this increase in learning rate goes beyond the upper bound, the update of the parameter is rejected. This rejection forces the learning rate to decrease according to the second phase. Once first phase finishes learning rate follows the rule of second phase until the learning last.

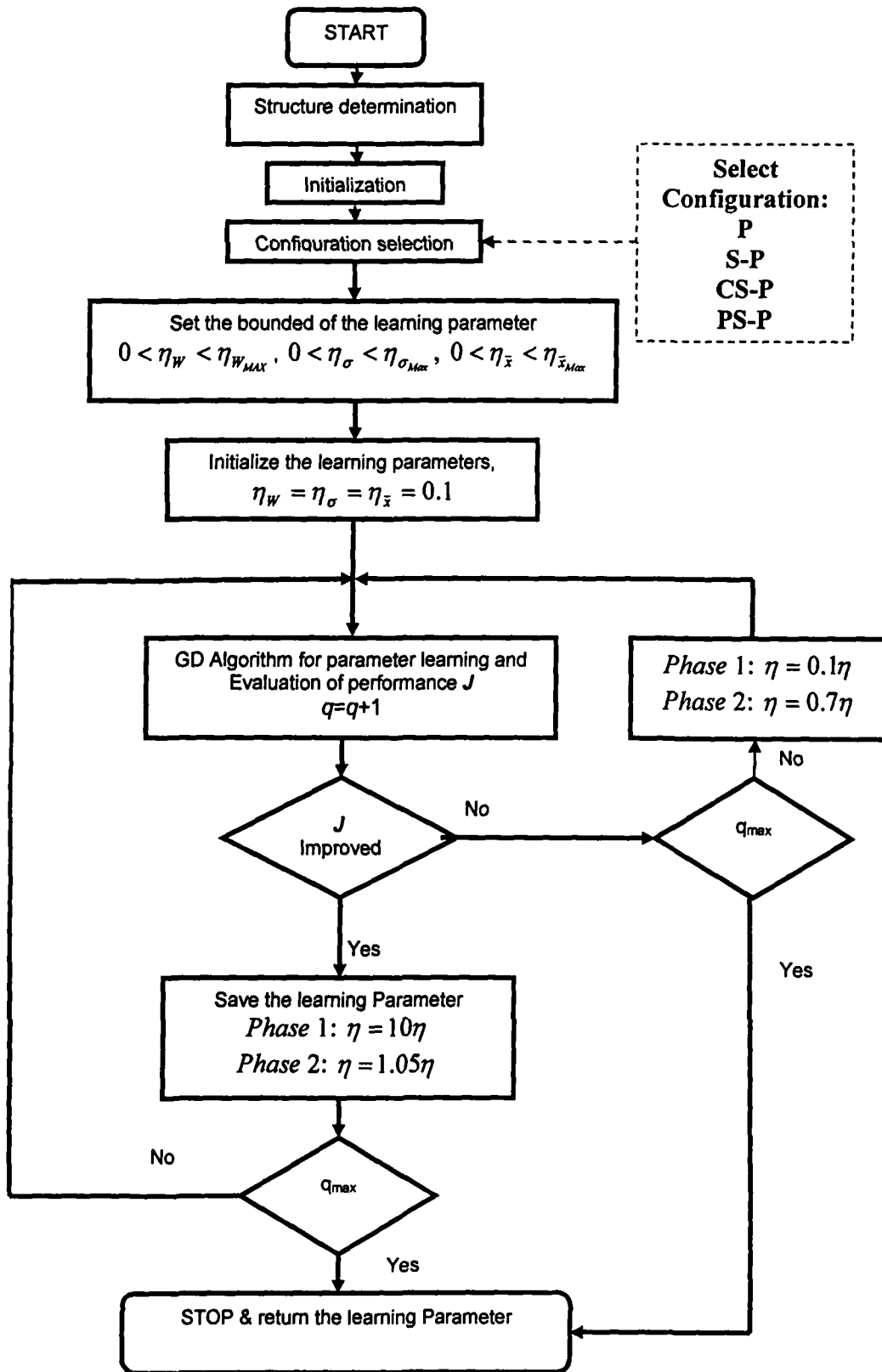


Fig. 4.7. Learning algorithm for Neuro-Fuzzy model

## 4-5 Simulation Results

In this section, different types of dynamic systems that have been discussed in chapter 1 is considered. The selected dynamic examples are different nonlinear equation with different dynamic order. First 4 example are dynamical equations and Example 5 is a general benchmark problem.

### Revisited Example 1: Linear regression with nonlinear input

By applying modified clustering and cluster validity function [Azeem'03a, Xie'87], five rules are obtained. Figure 4.8 illustrates the learning pattern and Table 4.1 shows the value of performance index of models obtained for all configurations. In this figure, the parallel-series category has shown by solid-blue, parallel with dot-black, PS-P with dot-slash & green and CS-P with slash-dot. The S-P and the CS-P configurations have better performance and CS-P is the best. It means that when the actual output of the system feedback to consequent part it yield better result. The initial fuzzy rules for the models are listed below:

$$R^1 : \text{if } u(k) \text{ is } A_1^1 \wedge \hat{y}(k-1) \text{ is } A_2^1 \wedge \hat{y}(k) \text{ is } A_3^1 \text{ then } y^1(k+1) \text{ is } w^1(X)$$

$$R^2 : \text{if } u(k) \text{ is } A_1^2 \wedge \hat{y}(k-1) \text{ is } A_2^2 \wedge \hat{y}(k) \text{ is } A_3^2 \text{ then } y^2(k+1) \text{ is } w^2(X)$$

$$R^3 : \text{if } u(k) \text{ is } A_1^3 \wedge \hat{y}(k-1) \text{ is } A_2^3 \wedge \hat{y}(k) \text{ is } A_3^3 \text{ then } y^3(k+1) \text{ is } w^3(X)$$

$$R^4 : \text{if } u(k) \text{ is } A_1^4 \wedge \hat{y}(k-1) \text{ is } A_2^4 \wedge \hat{y}(k) \text{ is } A_3^4 \text{ then } y^4(k+1) \text{ is } w^4(X)$$

$$R^5 : \text{if } u(k) \text{ is } A_1^5 \wedge \hat{y}(k-1) \text{ is } A_2^5 \wedge \hat{y}(k) \text{ is } A_3^5 \text{ then } y^5(k+1) \text{ is } w^5(X)$$

where,

$$w^1(X) = 0.0200 - 0.0356u(k) - 0.7385y(k-1) + 1.7168y(k)$$

$$w^2(X) = -0.2127 + 0.5018u(k) - 0.7315y(k-1) + 1.7663y(k)$$

$$w^3(X) = 0.0727 + 0.0387u(k) - 0.9743y(k-1) + 1.8446y(k)$$

$$w^4(X) = -1.4026 + 3.6808u(k) - 0.13761y(k-1) + 1.0485y(k)$$

$$w^5(X) = 0.4522 - 0.6490u(k) - 1.0283y(k-1) + 1.8407y(k)$$

The premise variable membership function  $A_1^1-A_1^5$ ,  $A_2^1-A_2^5$  and  $A_3^1-A_3^5$  for inputs  $u(k)$ ,  $\hat{y}(k-1)$  and  $\hat{y}(k)$  are shown in Fig. 4.9. The fuzzy rules corresponding to the learned network are listed below:

$$R^{1f} : \text{if } u(k) \text{ is } A_1^{1f} \wedge \hat{y}(k-1) \text{ is } A_2^{1f} \wedge \hat{y}(k) \text{ is } A_3^{1f} \text{ then } y^1(k+1) \text{ is } w^{1f}(X)$$

$$R^{2f} : \text{if } u(k) \text{ is } A_1^{2f} \wedge \hat{y}(k-1) \text{ is } A_2^{2f} \wedge \hat{y}(k) \text{ is } A_3^{2f} \text{ then } y^2(k+1) \text{ is } w^{2f}(X)$$

$$R^{3f} : \text{if } u(k) \text{ is } A_1^{3f} \wedge \hat{y}(k-1) \text{ is } A_2^{3f} \wedge \hat{y}(k) \text{ is } A_3^{3f} \text{ then } y^3(k+1) \text{ is } w^{3f}(X)$$

$$R^{4f} : \text{if } u(k) \text{ is } A_1^{4f} \wedge \hat{y}(k-1) \text{ is } A_2^{4f} \wedge \hat{y}(k) \text{ is } A_3^{4f} \text{ then } y^4(k+1) \text{ is } w^{4f}(X)$$

$$R^{5f} : \text{if } u(k) \text{ is } A_1^{5f} \wedge \hat{y}(k-1) \text{ is } A_2^{5f} \wedge \hat{y}(k) \text{ is } A_3^{5f} \text{ then } y^5(k+1) \text{ is } w^{5f}(X)$$

where,

$$w^{1f}(X) = 0.0246 - 0.0248u(k) - 0.7424y(k-1) + 1.7156y(k)$$

$$w^{2f}(X) = -0.2026 + 0.5045u(k) - 0.7499y(k-1) + 1.7484y(k)$$

$$w^{3f}(X) = 0.0757 + 0.0399u(k) - 0.9761y(k-1) + 1.8385y(k)$$

$$w^{4f}(X) = -1.4054 + 3.6797u(k) - 0.1360y(k-1) + 1.0498y(k)$$

$$w^{5f}(X) = 0.4420 - 0.6545u(k) - 1.0201y(k-1) + 1.8430y(k)$$

The learned premise variable membership functions  $A_1^{1f}-A_1^{5f}$ ,  $A_2^{1f}-A_2^{5f}$  and  $A_3^{1f}-A_3^{5f}$  for inputs  $u(k)$ ,  $\hat{y}(k-1)$  and  $\hat{y}(k)$  are shown in Fig. 4.10. Figure 4.11 shows the actual output, model output and model error with CS-P configuration. In this figure, actual output of the plant is solid-blue and the model output is dot-red. The error is solid-blue. The value of performance index  $J=1.8694 \times 10^{-7}$  for the model obtained for CS-P configuration.



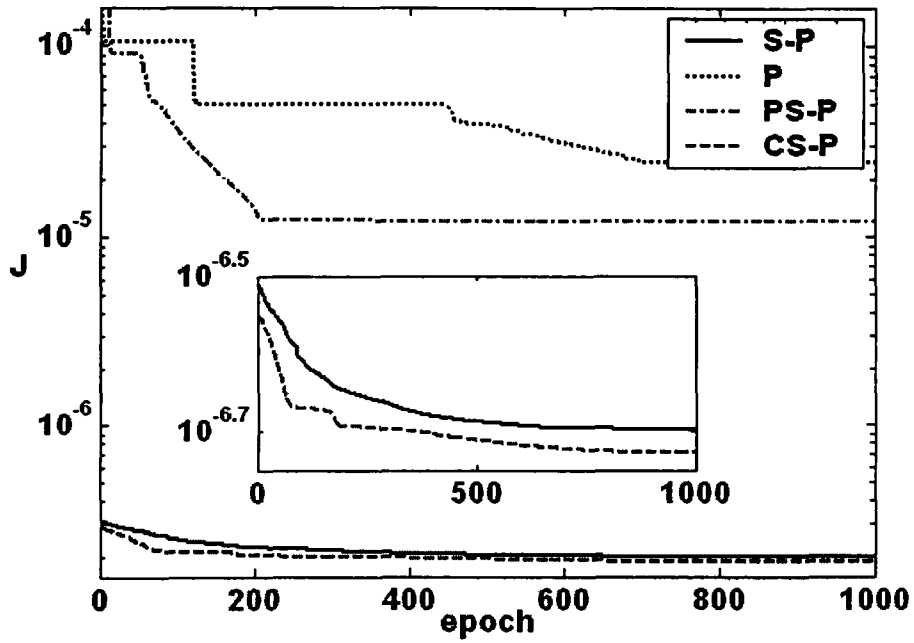


Fig. 4.8. Learning pattern of all configurations for Example 1

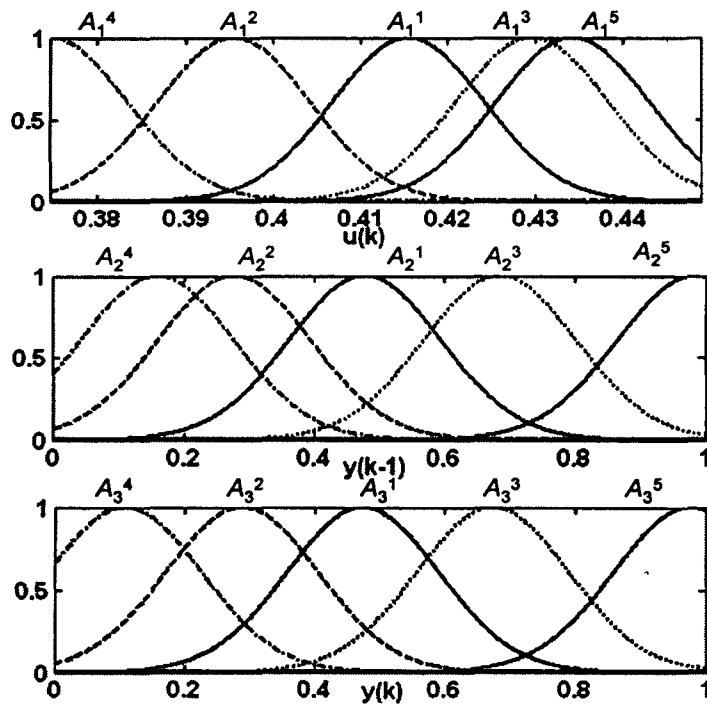


Fig. 4.9. Initial membership functions of the normalized inputs for Example 1

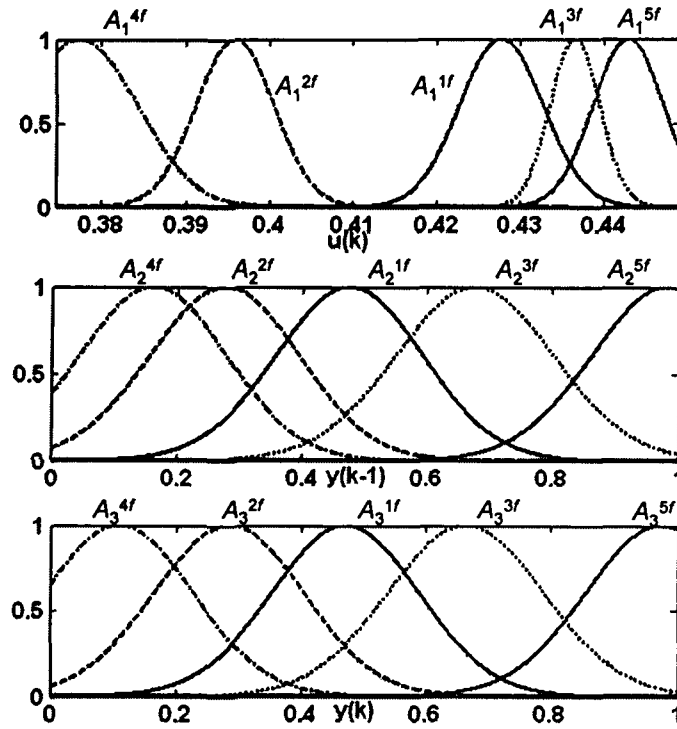


Fig. 4.10. Learned membership functions of the normalized inputs for Example 1

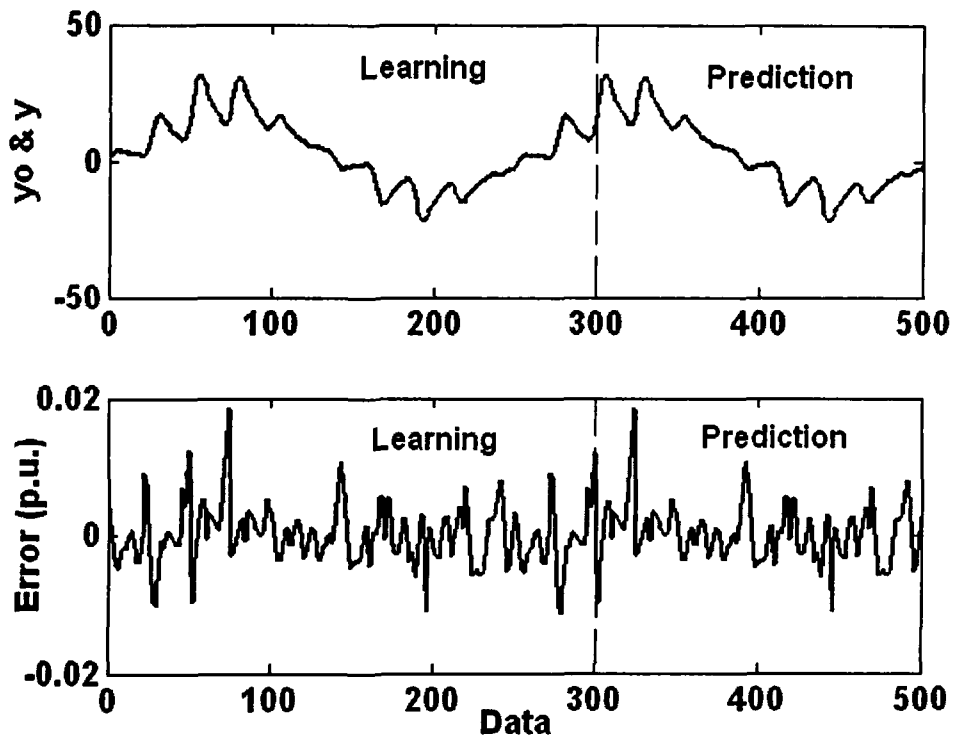


Fig. 4.11. Actual output & model output with CS-P configuration and the error for Example 1

## Revisited Example 2: Non-linear regression with random input

Figure 4.12 and Table 4.1, show the performance index for different identification configuration. The CS-P configuration model yields better result with five rules, obtained by MMC and cluster validity function. The initial fuzzy rule of the CS-P configuration is listed below:

$$R^1 : \text{if } u(k) \text{ is } A_1^1 \wedge \hat{y}(k-1) \text{ is } A_2^1 \wedge \hat{y}(k) \text{ is } A_3^1 \text{ then } y^1(k+1) \text{ is } w^1(X)$$

$$R^2 : \text{if } u(k) \text{ is } A_1^2 \wedge \hat{y}(k-1) \text{ is } A_2^2 \wedge \hat{y}(k) \text{ is } A_3^2 \text{ then } y^2(k+1) \text{ is } w^2(X)$$

$$R^3 : \text{if } u(k) \text{ is } A_1^3 \wedge \hat{y}(k-1) \text{ is } A_2^3 \wedge \hat{y}(k) \text{ is } A_3^3 \text{ then } y^3(k+1) \text{ is } w^3(X)$$

$$R^4 : \text{if } u(k) \text{ is } A_1^4 \wedge \hat{y}(k-1) \text{ is } A_2^4 \wedge \hat{y}(k) \text{ is } A_3^4 \text{ then } y^4(k+1) \text{ is } w^4(X)$$

where,

$$w^1(X) = 0.3527 + 0.6184u(k) - 0.5301y(k-1) + 0.5573y(k)$$

$$w^2(X) = 0.3308 + 0.7160u(k) - 0.3063y(k-1) - 0.1194y(k)$$

$$w^3(X) = 0.6575 + 0.6492u(k) - 0.3711y(k-1) - 0.2846y(k)$$

$$w^4(X) = 0.5092 + 0.7051u(k) - 0.6106y(k-1) - 0.4889y(k)$$

The premise variable membership function  $A_1^1$ - $A_1^4$ ,  $A_2^1$ - $A_2^4$  and  $A_3^1$ - $A_3^4$  for inputs  $u(k)$ ,  $\hat{y}(k-1)$  and  $\hat{y}(k)$  are shown in Fig. 4.13. The fuzzy rules corresponding to the learned non-normalized network are listed below:

$$R^{1f} : \text{if } u(k) \text{ is } A_1^{1f} \wedge \hat{y}(k-1) \text{ is } A_2^{1f} \wedge \hat{y}(k) \text{ is } A_3^{1f} \text{ then } y^1(k+1) \text{ is } w^{1f}(X)$$

$$R^{2f} : \text{if } u(k) \text{ is } A_1^{2f} \wedge \hat{y}(k-1) \text{ is } A_2^{2f} \wedge \hat{y}(k) \text{ is } A_3^{2f} \text{ then } y^2(k+1) \text{ is } w^{2f}(X)$$

$$R^{3f} : \text{if } u(k) \text{ is } A_1^{3f} \wedge \hat{y}(k-1) \text{ is } A_2^{3f} \wedge \hat{y}(k) \text{ is } A_3^{3f} \text{ then } y^3(k+1) \text{ is } w^{3f}(X)$$

$$R^{4f} : \text{if } u(k) \text{ is } A_1^{4f} \wedge \hat{y}(k-1) \text{ is } A_2^{4f} \wedge \hat{y}(k) \text{ is } A_3^{4f} \text{ then } y^4(k+1) \text{ is } w^{4f}(X)$$

where,

$$w^{1f}(X) = 0.4130 + 0.7532u(k) - 0.4786y(k-1) + 0.4839y(k)$$

$$w^{2f}(X) = 0.3017 + 0.7805u(k) - 0.3881y(k-1) - 0.1744y(k)$$

$$w^{3f}(X) = 0.6266 + 0.5967u(k) - 0.1121y(k-1) - 0.2307y(k)$$

$$w^{4f}(X) = 0.5018 + 0.8116u(k) - 0.4932y(k-1) - 0.6007y(k)$$

The learned premise variable membership function functions  $A_1^{1f}$ - $A_1^{4f}$ ,  $A_2^{1f}$ - $A_2^{4f}$  and  $A_3^{1f}$ - $A_3^{4f}$  for inputs  $u(k)$ ,  $\hat{y}(k-1)$  and  $\hat{y}(k)$  are shown in Fig. 4.14. Figure 4.15 shows actual output and model output of the CS-P model. The error for learning and prediction section is shown in Fig. 4.15. The value of performance Index  $J=1.1040 \times 10^{-6}$  is obtained for CS-P configuration.

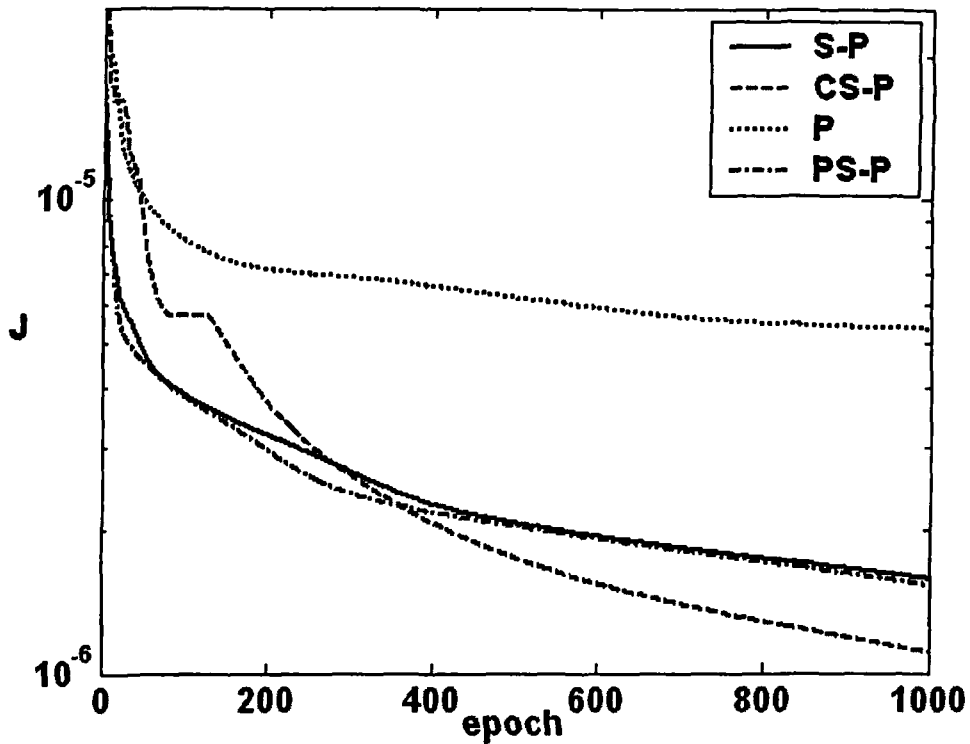


Fig. 4.12. Learning pattern of all configurations for Example 2

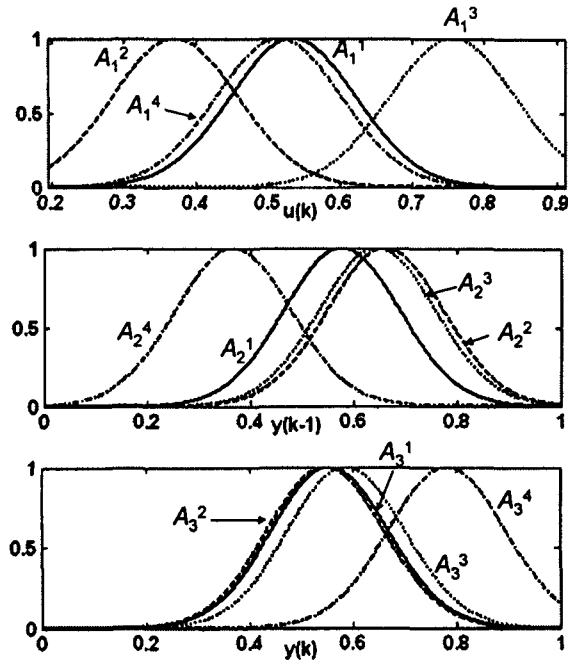


Fig. 4.13. Initial membership functions of the normalized inputs for Example 2

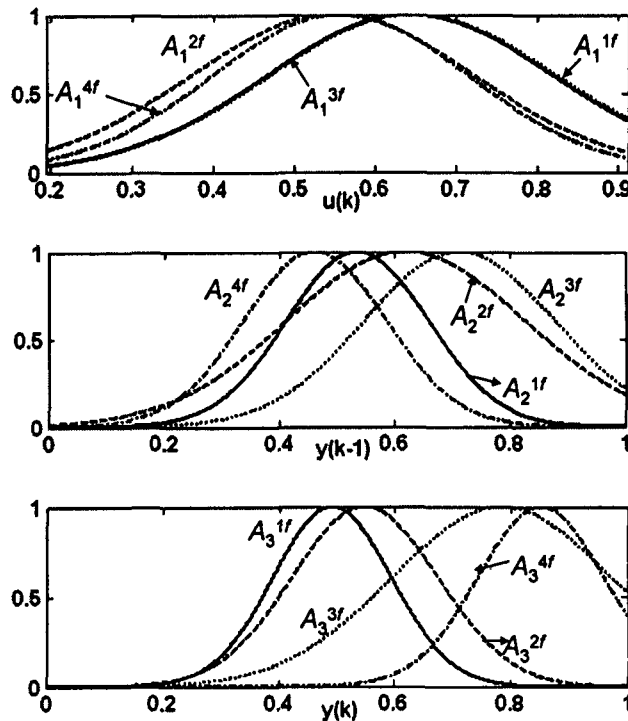


Fig. 4.14. Learned membership functions of the normalized inputs for Example 2

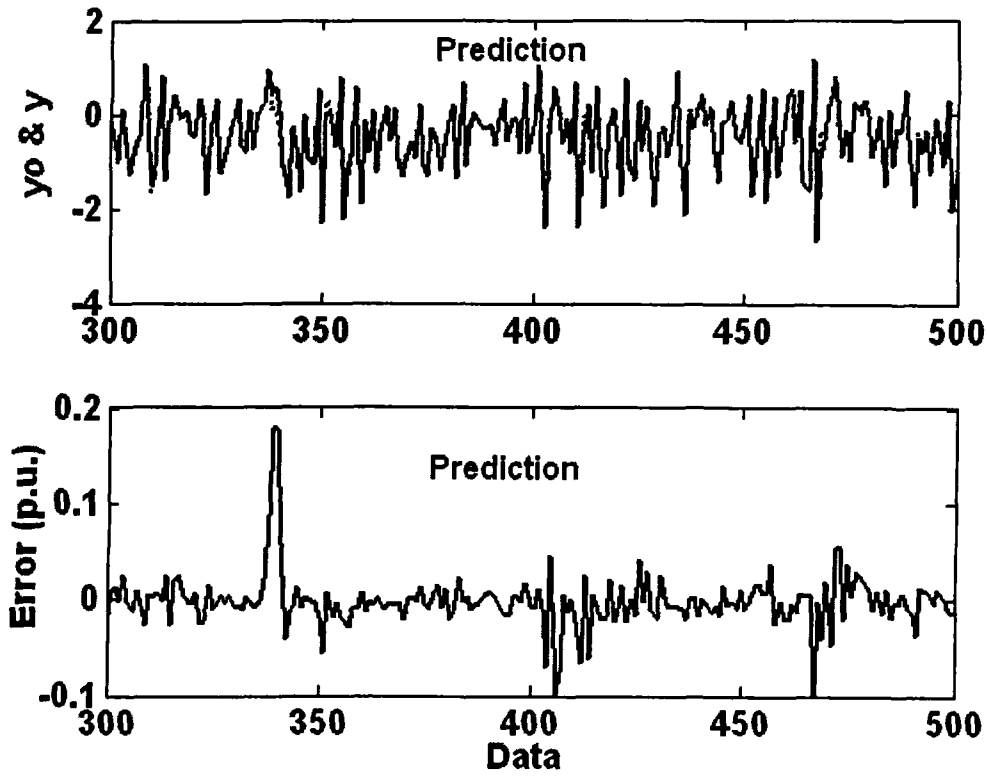


Fig. 4.15. Actual output & model output with CS-P configuration and the error for Example 2

### Revisited Example 3: *Non-Linear Regression with Non-Linear Input*

By applying MMC and cluster validity, three rules are generated. Figure 4.16 and Table 4.1, show performance index for different configuration of the identification models. CS-P configuration model yield better result with performance Index  $J=3.2015 \times 10^{-6}$ . Next on S-P model is better. The initial fuzzy rule of the CS-P configuration is listed below:

$$R^1 : \text{if } u(k) \text{ is } A_1^1 \wedge \hat{y}(k) \text{ is } A_2^1 \text{ then } y^1(k+1) \text{ is } w^1(X)$$

$$R^2 : \text{if } u(k) \text{ is } A_1^2 \wedge \hat{y}(k) \text{ is } A_2^2 \text{ then } y^2(k+1) \text{ is } w^2(X)$$

$$R^3 : \text{if } u(k) \text{ is } A_1^3 \wedge \hat{y}(k) \text{ is } A_2^3 \text{ then } y^3(k+1) \text{ is } w^3(X)$$

where,

$$w^1(X) = -1.3762 + 2.5888u(k) + 1.0779y(k)$$

$$w^2(X) = -3.8700 + 7.7564u(k) - 0.0247y(k)$$

$$w^3(X) = -2.9392 + 8.0787u(k) + 0.0428y(k)$$

algorithms have been proposed [Davis'89, Shaefer'87, Whitley'95, Lobo'97, Azeem'03b]. Azeem et.al. [Azeem'03b] introduced a hybrid method based on GD and GA. They have applied GA in each epoch of GD to increase the convergence and also to maximize the possibility of localizing in the region of global minimum. In this work we applied GA to initialization the proposed WNF models. Later by using GD we tune the learning parameters of the WNF model.

The organization of this chapter is as follows: Wavelet Neuro-Fuzzy model is introduced in Section 5-2. Section 5-3 envelops the initialization of the learning algorithm based on genetic algorithm and fine tuning of the WNF parameter by gradient descent. Examples and simulation are discussed in section 5-4 followed by the conclusion in section 5-5.

## 5-2 Wavelet Neuro-Fuzzy

In this section, based on MS-W neuron model, which has yield better performance as discussed in chapter 3, WNF model is proposed. The antecedent part of each fuzzy rule in the proposed neuro-fuzzy model represents input space in which a local model operates. These local models are estimated by MS-W neuron model, see chapter 3: Fig. 3.2 and Fig. 3.3.

### 5-2.1 Architecture of Proposed Wavelet Neuro-Fuzzy

Figure 5.1 shows a wavelet neuro-fuzzy model. This model can be described by a set of following fuzzy rules:

$$R^m : IF x_1 \text{ is } A_1^m \text{ and } \dots \text{ and } x_n \text{ is } A_n^m \text{ THEN } \hat{y}_m = \hat{Y}_{WNN_m} \quad (5.1)$$

Where  $R^m$  is the  $m^{\text{th}}$  rule;  $x_i$  is the  $i^{\text{th}}$  input variable;  $\hat{y}_m$  is the output of the  $m^{\text{th}}$  local model for rule  $R^m$ ;  $\hat{Y}_{WNN_m}$  is output of local, MS-W neuron network, model; and  $A_i^m$  is the linguistic term of the premise part with Gaussian membership function given by (4.7). From the Fig. 5.1, structure of WNF is described as follows:

Layer 1: Nodes in layer 1 represent input variable. Every node accepts input values and transmits it to the next layer.

Layer 2: Nodes in this layer represent the terms of the respective linguistic variables. Every node operates on incoming signal with Gaussian membership function expressed by (4.7).

The parameters to be learned in this layer are  $\bar{x}_m$  and  $\sigma_m$ . Corresponding to each rule the learning parameter are expressed in vector form as  $\bar{x}_m = \{\bar{x}_{m1}, \bar{x}_{m2}, \dots, \bar{x}_{mn}\}$  and  $\sigma_m = \{\sigma_{m1}, \sigma_{m2}, \dots, \sigma_{mn}\}$ .

Layer 3: Each node in layer 3 represents a fuzzy rule. The output from the nodes in layer 2, specified for a fuzzy rule, is being input to the nodes, specified for that rule, in layer 3. The output of each node in layer 3 is the product of all inputs; it represents the firing strength of that rule. Thus the firing strength of the  $m^{\text{th}}$  rule is specified as (4.4).

Layer 4: Nodes in layer 4 are called consequent nodes. Two inputs are applied to each node in this layer, namely the output from the layer 3 node and the output from its corresponding local model approximated by MS-W neuron network. The output of each node is the product of both input and given by (5.2).

$$\hat{Y}_{WNF_m}^* = \mu_{A^m}(\mathbf{X}) \cdot \hat{Y}_{WNN_m}(\mathbf{X}') \quad (5.2)$$

Layer 5: There are three nodes in this layer that constitutes the aggregation and defuzzification of fuzzy rules. The output of all nodes from layer 4 is the input to the first node and its output is the summation of all inputs and expressed as (5.3). The output of all nodes from layer 3 is the input to the second node and its output is the summation of all inputs and expressed as (5.4). Inputs to the third node in this layer are the output from first and second node. The output of the third node is the ratio of these two input and given in (5.5). The output of the third node in this layer is the output of WNF.



$$a = \sum_{m=1}^M (\mu_{A^m}(\mathbf{X}) \cdot \hat{Y}_{WNN_m}(\mathbf{X}')) \quad (5.3)$$

$$b = \sum_{m=1}^M \mu_{A^m}(\mathbf{X}) \quad (5.4)$$

$$\hat{Y}_{WNF} = \frac{a}{b} = \frac{\sum_{m=1}^M (\mu_{A^m}(\mathbf{X}) \cdot \hat{Y}_{WNN_m}(\mathbf{X}'))}{\sum_{m=1}^M \mu_{A^m}(\mathbf{X})} \quad (5.5)$$

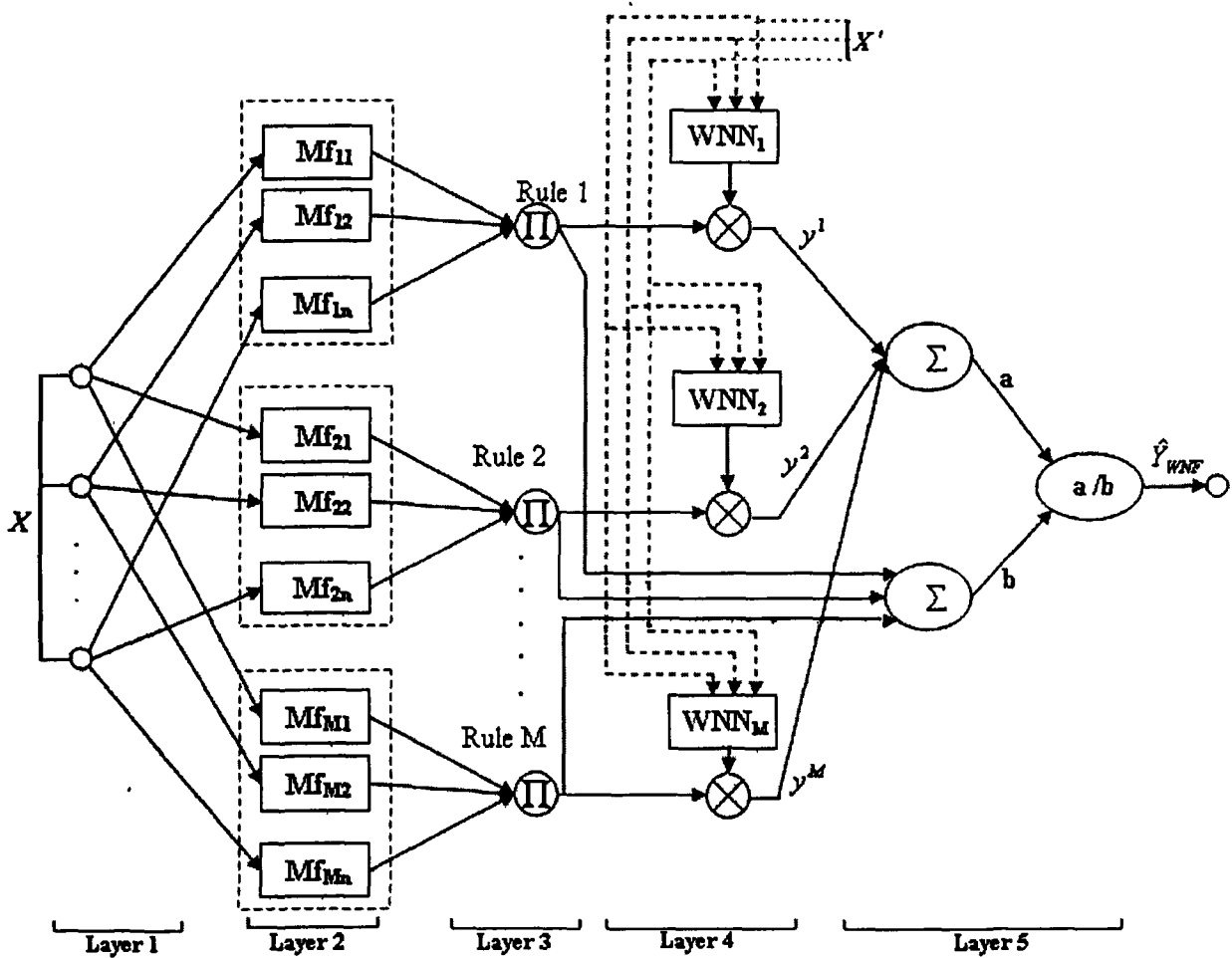


Fig. 5.1. Proposed Wavelet Neuro-Fuzzy Model

In these equations  $n$  is the number of inputs and  $M$  is number of rules and the number of fuzzy sets for each input is supposed to be equal to the number of rules. To determine number of necessary rules Modified Mountain Clustering (MMC) is applied. The purpose of clustering is to do natural grouping of large set of data, producing a concise representation of system's behavior. The description of MMC can be found in Appendix B.

## **5-3 Genetic Algorithm and Gradient Descent**

In this section a hybrid algorithm based on GA and GD introduced for learning of the parameter. We use the GA for initialization of the learning parameter. After that by applying GD technique the learning parameters are adjusted. In this chapter we have applied CS-P configuration that yield better performance as discussed in chapter 4. In CS-P configuration, the plant inputs and output with its delays are employed to approximate local models in consequent part of the fuzzy rules, whereas inputs and delays of the model output are utilized to comprehend the input space region. The premise and the consequent inputs in this configuration are denoted by (4.18) and (4.19), respectively.

### **5-3.1 Basic of the Genetic Algorithm**

The basic theory of GA can be found in [Goldberg'89], and in this Section we will discuss in brief what are the components of GA and how they function in the solution process.

Suppose we are seeking to find a solution to a problem. To apply a genetic algorithm to that problem, the first thing to do is to encode the problem into artificial chromosomes. These artificial chromosomes can be the strings of 1's and 0's, or the parameter lists, or even the complex computer codes, but the key thing to keep in mind is that the genetic machinery will manipulate a finite representation of the solutions, not the solutions themselves. The second

thing to do to solve a problem is to have some means of discriminating good solutions from bad ones. This can be as simple as having a human intuitively choose better solutions, or it can be an elaborate computer simulation or a model that helps to determine the quality of a solution. But the idea is to ascertain a solution's relative *fitness* to purpose by some means. The genetic algorithm will use these very means to guide the evolution of future generations.

Having encoded the problem in terms of chromosomes and having devised a means of discriminating good solutions from bad ones, we prepare to evolve solutions to our problem by creating an initial population of encoded solutions. The population can be created randomly or by using prior knowledge of possible good solutions, but either way GA will search from a population, not from a single point.

There are various types of operators that are used in GA, but quite often (i) selection, (ii) recombination and (iii) mutation. The selection and genetic operators can process the population iteratively to create a sequence of populations that will hopefully contain more and more good solutions to our problem over a period of time.

To cite briefly, selection operator allocates greater survival to better individuals. This is what is known as the survival of the fittest mechanism, which we impose on our solutions. This can be accomplished in a variety of ways. Weighted roulette wheels can be spun, local tournaments can be held, various ranking schemes can be invoked, but whatever we do, the main task is to seek better solutions over worse ones. Of course, if we were to only choose better solutions repeatedly from the original database of initial solutions, we would expect the population to contain the best solution of the first generation. However, simply selecting the best is not enough, and some means of creating new, possibly better individuals must be found. This is where the mechanisms like recombination and mutation emerge.

Recombination is a genetic operator that combines bits and pieces of parental solutions to form new, possibly better offspring. Again, there are many ways of accomplishing this. Achieving desirable performance does depend on getting the recombination mechanism designed properly; but the primary concern is to see that the offspring under recombination will not be identical to any particular parent, so we combine the parental traits in a novel manner. Recombination by itself is not very useful, because a population of individuals processed under repeated recombination alone will undergo what amounts to a random shuffling of existing traits.

As against recombination, which creates a new individual by combining the traits of two or more parents, mutation acts by simply modifying a single individual. There are many variations of mutation, but the main constraint is that the offspring must have traits identical to the individual parental traits except that the operator may make one or more changes to an individual's traits. Mutation by itself represents a "random walk" in the neighborhood of a particular solution. If applied repeatedly over a population of individuals, we might expect the resulting population to be indistinguishable from the one created at random.

### **5-3.2 Components of GA**

There are various possible combinations of different components to construct a GA. A detailed description of different combinations can be found in [Goldberg'89, Davis'91]. Here we shall give a brief description of the combinations used in this thesis.

#### **a) Solution Representation (Encoding & Decoding)**

Binary string representation scheme is used to perform GA evolution. Since we are required to encode the center matrix  $c \in R^{n \times m}$ , we normalize each element of  $c$  in the search space by the span, i.e.,  $(c_{k_i, \max} - c_{k_i, \min})$ , to yield  $c_{k_i, \text{nor}} = (c_{k_i} - c_{k_i, \min}) / (c_{k_i, \max} - c_{k_i, \min})$ . The

decimal value,  $decimal(c_{k_i,2})$ , of each element of  $c$ , for the binary string of length  $h$ , is obtained from the relation:  $decimal(c_{k_i,2}) = c_{k_i,nor} / (2^h - 1)$  and the resolution of the binary string is  $(c_{k_i,max} - c_{k_i,min}) / 2^h$ . Now the  $decimal(c_{k_i,2})$  is converted into the binary string by adding sufficient number of “0s” on the left side of the string in order to complete the specified string length, i.e.  $h$ . With this, the total binary length for each solution (chromosome) is  $h \times m \times n$ . A fixed binary length  $h$  is taken as 10 for each  $c_{k_i}$ . Further, it may be noted that gradient descent learning takes care of resolution interval, inherited by the genetic coding, if the solution lands in the region of basin.

The solution must be decoded before it is evaluated. Steps involved in decoding are (i) separate the string of length  $h$  corresponding to each  $c_{k_i}$ , (ii) convert this string into decimal value  $decimal(c_{k_i,2})$  and (iii) obtain the value of  $c_{k_i}$  from  $decimal(c_{k_i,2})$  by the following formula:

$$c_{k_i} = c_{k_i,min} + decimal(c_{k_i,2}) \frac{(c_{k_i,max} - c_{k_i,min})}{(2^h - 1)} \quad (5.6)$$

## b) Initialization

A specified number of solution strings of 0’s and 1’s is generated randomly as an initial population.

## c) Evaluation function

Each decoded solution represents a model. An evaluation (fitness) function is defined to evaluate the degree of fitness of all the models with respect to learning data set. Our goal is to minimize the objective function  $J$  defined by (2.18). Since, GA is used strictly for the maximization problems, without loss of generality; fitness function is defined as the reciprocal of

the objective function  $J$ . So, minimization of  $J$  and maximization of fitness function are equivalent.

#### d) Selection, Crossover, Mutation and Reproduction

Weighted roulette wheels approach is opted for parent selection. In the parent selection care has to be taken such as two identical parents should not crossover, to prevent the production of two identical children similar to their parent, whose fitness has already been tested.

The two-point crossover is used for the reproduction of offspring. One point is applied to the premise part of the rules and one point to the consequent part. The two points for the crossover in the chromosome string are selected randomly. The probability of crossover is set at  $p_c = 0.8$ .

The number of mutations in a solution is randomly selected with a very small value of probability, i.e.,  $p_m = 0.02$ .

The technique of generational replacement without duplication is used for reproduction, and to test for a new solution. In this technique, all the solutions of one generation are replaced by the solutions of the next generation.

### 5-3.3 Gradient Descent learning of parameters

After initialization of the learning parameters by GA, we apply gradient descent technique. Figure 5.2 shows algorithm for initialization by GA and fine tuning of the learning parameter of the WNF model. The parameter update formulas for  $p^{\text{th}}$  data set are as follows:

$$\Delta_p \sigma_{mi}(q) = -\eta_\sigma \cdot \frac{\partial J}{\partial \sigma_{mi}} = -\eta_\sigma \cdot \frac{-1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial \sigma_{mi}} \right|_p \quad (5.7)$$

$$\Delta_p \bar{x}_{mi}(q) = -\eta_{\bar{x}} \cdot \frac{\partial J}{\partial \bar{x}_{mi}} = -\eta_{\bar{x}} \cdot \frac{-1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial \bar{x}_{mi}} \right|_p \quad (5.8)$$

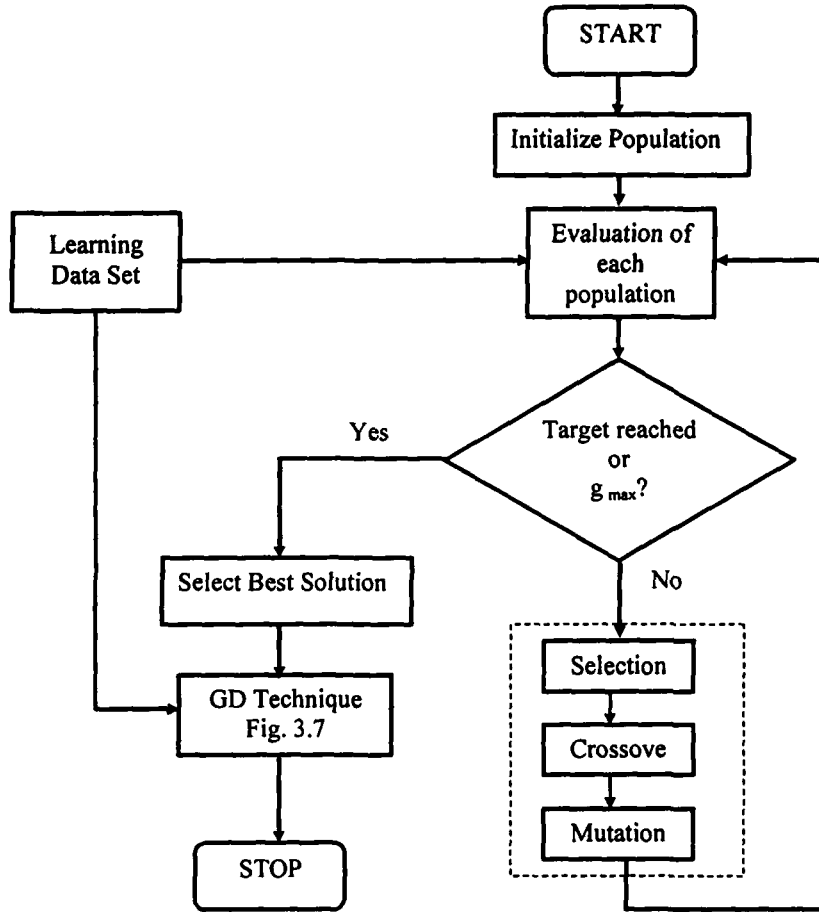


Fig. 5.2. Algorithm for initialized and learning of the learning parameters

$$\Delta_p W_i^m(q) = -\eta \frac{\partial J}{\partial W_i^m} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial W_i^m} \right|_p \quad (5.9)$$

$$\Delta_p C_{w_i^m}(q) = -\eta \frac{\partial J}{\partial C_{w_i^m}} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial C_{w_i^m}} \right|_p \quad (5.10)$$

$$\Delta_p C_{s_i^m}(q) = -\eta \frac{\partial J}{\partial C_{s_i^m}} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \left. \frac{\partial \hat{Y}}{\partial C_{s_i^m}} \right|_p \quad (5.11)$$

where  $e = Y - \hat{Y}$  is the error between the plant output,  $Y$ , and the model output,  $\hat{Y}$ . By applying

chain rule to the above equation  $\frac{\partial \hat{Y}}{\partial \bar{x}_{m_i}}$ ,  $\frac{\partial \hat{Y}}{\partial \sigma_{m_i}}$ ,  $\frac{\partial \hat{Y}}{\partial W_{l_i}^m}$ ,  $\frac{\partial \hat{Y}}{\partial C_{W_{l_i}^m}}$  and  $\frac{\partial \hat{Y}}{\partial C_{S_{l_i}^m}}$  are derived as follows:

Define  $\beta_m = \frac{\mu_{A^m}(\mathbf{X})}{\sum_{m=1}^M \mu_{A^m}(\mathbf{X})}$ , then

$$\frac{\partial \hat{Y}}{\partial \bar{x}_{m_i}} = \hat{Y}_{WNN_m}(\mathbf{X}') \cdot \frac{\beta_m}{\mu_{A^m}} \cdot (1 - \beta_m) \cdot \frac{2 \cdot (x_i - \bar{x}_{m_i})}{\sigma_{m_i}^2} \quad (5.12)$$

$$\frac{\partial \hat{Y}}{\partial \sigma_{m_i}} = \hat{Y}_{WNN_m}(\mathbf{X}') \cdot \frac{\beta_m}{\mu_{A^m}} \cdot (1 - \beta_m) \cdot \frac{2 \cdot (x_i - \bar{x}_{m_i})^2}{\sigma_{m_i}^3} \quad (5.13)$$

$$\frac{\partial \hat{Y}}{\partial W_{l_i}^m} = \beta_m \cdot \frac{\partial \hat{Y}_{WNN_m}(\mathbf{X}')}{\partial W_{l_i}^m} \quad (5.14)$$

$$\frac{\partial \hat{Y}}{\partial C_{W_{l_i}^m}} = \beta_m \cdot \frac{\partial \hat{Y}_{WNN_m}(\mathbf{X}')}{\partial C_{W_{l_i}^m}} \quad (5.15)$$

$$\frac{\partial \hat{Y}}{\partial C_{S_{l_i}^m}} = \beta_m \cdot \frac{\partial \hat{Y}_{WNN_m}(\mathbf{X}')}{\partial C_{S_{l_i}^m}} \quad (5.16)$$

$\frac{\partial \hat{Y}_{WNN_m}}{\partial C_{W_{l_i}^m}}$ ,  $\frac{\partial \hat{Y}_{WNN_m}}{\partial C_{N_{l_i}^m}}$  and  $\frac{\partial \hat{Y}_{WNN_m}}{\partial C_{W_{l_i}^m}}$  in (5.14) to (5.16) for WNF model are expressed by (3.24) to

(3.26), respectively.

### 5-3.4 Learning Convergence theorem

To consider stability of learning procedure Lyapunov stability theorem is applied. Theorem 5.1 shows convergence condition for proposed neuro-fuzzy models. The proof of this theorem is derived in Appendix C.

Theorem 5.1: The asymptotic learning convergence is guaranteed if the learning rate for different learning parameters follows the upper bound as mentioned below:



$$0 < \eta_{\sigma} < \frac{2 \cdot P \cdot y_r^2}{\left| \hat{Y}_{WNN}(\mathbf{X}') \right|_{\max}^2 \cdot \left( \frac{2}{\sigma_{\min}^3} \right)^2} \quad (5.17)$$

$$0 < \eta_{\bar{x}} < \frac{2 \cdot P \cdot y_r^2}{\left| \hat{Y}_{WNN}(\mathbf{X}') \right|_{\max}^2 \cdot \left( \frac{2}{\sigma_{\min}^2} \right)^2} \quad (5.18)$$

$$0 < \eta_w < \frac{2 \cdot P \cdot y_r^2}{\left| \frac{\partial \hat{Y}_{WNN}(\mathbf{X}')}{\partial w} \right|_{\max}^2} \quad (5.19)$$

$$0 < \eta_{C_s} < \frac{2 \cdot P \cdot y_r^2}{\left| \frac{\partial \hat{Y}_{WNN}(\mathbf{X}')}{\partial C_s} \right|_{\max}^2} \quad (5.20)$$

$$0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{\left| \frac{\partial \hat{Y}_{WNN}(\mathbf{X}')}{\partial C_w} \right|_{\max}^2} \quad (5.21)$$

## 5-4 Simulation Results

In this chapter, Multiplication Sigmoid-Wavelet (MS-W) network with the Morlet wavelet function is used in consequent part of the neuro-fuzzy model. In this chapter, the weights between input layer and hidden layer in MS-W neuron model,  $C_s$  and  $C_w$  in Fig. 3.2 & 3.3, for all rules are same and only the output weights,  $W$  in Fig. 3.2, are changed. Consequent Series-Parallel (CS-P) configuration that result better performance as mentioned in chapter 4 is selected to learn the parameters of the WNF model. MMC and GA have been applied to determine the structure and to initialize the network.

### Revisited Example 1: *Linear regression with nonlinear input*

With five rules as mentioned in Revisited Example 1 in section 4-5, we have applied genetic algorithm with 100 populations. We have fed the initial parameters for GA randomly. Figure 5.3 shows the maximum fitness of to each generation. The initial solution for GD is obtained over 250 generation. The value of performance index  $J$ , obtained by GA for initialization of the parameter, is  $2.1062 \times 10^{-6}$ .

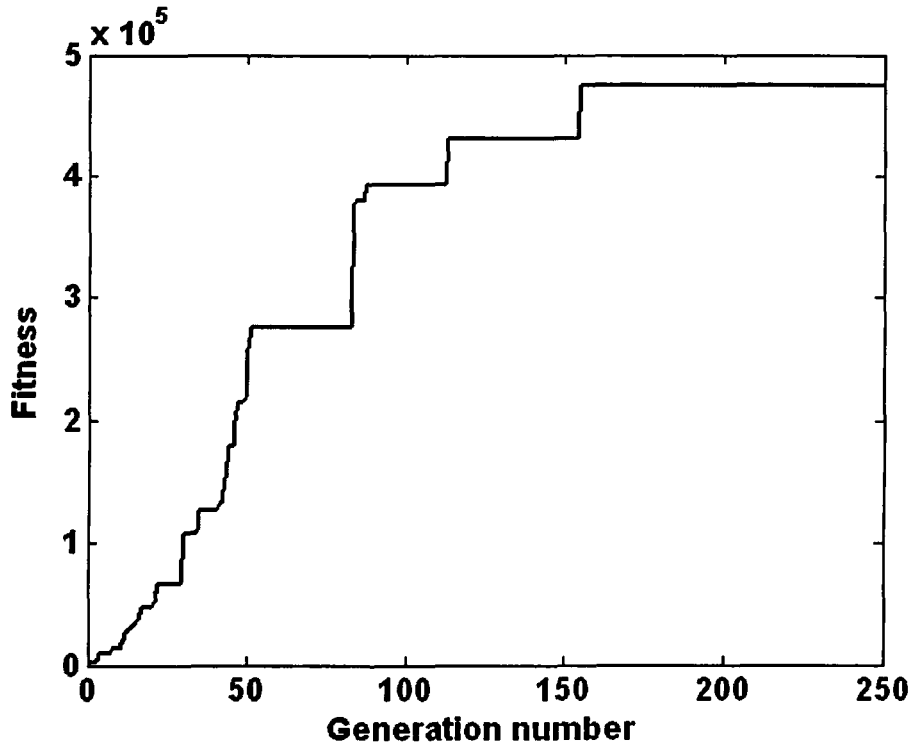


Fig. 5.3. Maximum fitness of GA up to each generation for Example 1

The initial fuzzy rules generalized by GA are listed below:

$$R^1 : \text{if } u(k) \text{ is } A_1^1 \wedge \hat{y}(k-1) \text{ is } A_2^1 \wedge \hat{y}(k) \text{ is } A_3^1 \text{ then } y^1(k+1) \text{ is } \hat{Y}_{WNN}^1(X)$$

$$R^2 : \text{if } u(k) \text{ is } A_1^2 \wedge \hat{y}(k-1) \text{ is } A_2^2 \wedge \hat{y}(k) \text{ is } A_3^2 \text{ then } y^2(k+1) \text{ is } \hat{Y}_{WNN}^2(X)$$

$$R^3 : \text{if } u(k) \text{ is } A_1^3 \wedge \hat{y}(k-1) \text{ is } A_2^3 \wedge \hat{y}(k) \text{ is } A_3^3 \text{ then } y^3(k+1) \text{ is } \hat{Y}_{WNN}^3(X)$$

$$R^4 : \text{if } u(k) \text{ is } A_1^4 \wedge \hat{y}(k-1) \text{ is } A_2^4 \wedge \hat{y}(k) \text{ is } A_3^4 \text{ then } y^4(k+1) \text{ is } \hat{Y}_{WNN}^4(X)$$

$$R^5 : \text{if } u(k) \text{ is } A_1^5 \wedge \hat{y}(k-1) \text{ is } A_2^5 \wedge \hat{y}(k) \text{ is } A_3^5 \text{ then } y^5(k+1) \text{ is } \hat{Y}_{WNN}^5(X)$$

where  $\hat{Y}_{WNN}^1(X)$ ,  $\hat{Y}_{WNN}^2(X)$ , ...,  $\hat{Y}_{WNN}^5(X)$  are the outputs of initialized MS-W neuron models by GA in consequent parts of  $R^1$  to  $R^5$ , respectively. Initialization of the learning parameters  $C_S$ ,  $C_W$  and  $W^1, \dots, W^5$ , for  $R^1$  to  $R^5$ , are as follow:

$$C_S = \begin{bmatrix} 0.53891 & 0.69682 & 0.62516 \\ 0.12482 & 0.37344 & 0.48868 \\ 0.21126 & 0.33559 & 0.25978 \end{bmatrix} \quad C_W = \begin{bmatrix} 0.052127 & 0.82934 & 0.9143 \\ 0.89941 & 0.98358 & 0.28005 \\ 0.30861 & 0.34731 & 0.77825 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \\ W^4 \\ W^5 \end{bmatrix} = \begin{bmatrix} 0.3613 & 0.0796 & 0.6563 \\ 0.9821 & 0.1616 & 0.9548 \\ 0.9242 & 0.0984 & 0.9061 \\ 0.6155 & 0.2415 & 0.1252 \\ 0.0079 & 0.6479 & 0.9080 \end{bmatrix}$$

Each hidden neuron in MS-W neuron model is conjunction of Sigmoid and Wavelet activation functions. Rows and column in  $C_S$  and  $C_W$  are corresponding to the number of hidden neurons in MS-W neuron model and the number of inputs. The number of rows in  $W$  is equal to the number of rules whereas number of column is equal to the number of hidden neuron in MS-W neuron model.

The premise variable membership function  $A_1^1-A_1^5$ ,  $A_2^1-A_2^5$  and  $A_3^1-A_3^5$  for inputs  $u(k)$ ,  $\hat{y}(k-1)$  and  $\hat{y}(k)$  are shown in Fig. 5.4. The fuzzy rules corresponding to the learned WNF are listed below:

$$R^{1f} : \text{if } u(k) \text{ is } A_1^{1f} \wedge \hat{y}(k-1) \text{ is } A_2^{1f} \wedge \hat{y}(k) \text{ is } A_3^{1f} \text{ then } y^1(k+1) \text{ is } \hat{Y}_{WNN}^{1f}(X)$$

$$R^{2f} : \text{if } u(k) \text{ is } A_1^{2f} \wedge \hat{y}(k-1) \text{ is } A_2^{2f} \wedge \hat{y}(k) \text{ is } A_3^{2f} \text{ then } y^2(k+1) \text{ is } \hat{Y}_{WNN}^{2f}(X)$$

$$R^{3f} : \text{if } u(k) \text{ is } A_1^{3f} \wedge \hat{y}(k-1) \text{ is } A_2^{3f} \wedge \hat{y}(k) \text{ is } A_3^{3f} \text{ then } y^3(k+1) \text{ is } \hat{Y}_{WNN}^{3f}(X)$$

$$R^{4f} : \text{if } u(k) \text{ is } A_1^{4f} \wedge \hat{y}(k-1) \text{ is } A_2^{4f} \wedge \hat{y}(k) \text{ is } A_3^{4f} \text{ then } y^4(k+1) \text{ is } \hat{Y}_{WNN}^{4f}(X)$$

$$R^{5f} : \text{if } u(k) \text{ is } A_1^{5f} \wedge \hat{y}(k-1) \text{ is } A_2^{5f} \wedge \hat{y}(k) \text{ is } A_3^{5f} \text{ then } y^5(k+1) \text{ is } \hat{Y}_{WNN}^{5f}(X)$$

where  $\hat{Y}_{WNN}^{1f}(X)$ ,  $\hat{Y}_{WNN}^{2f}(X)$ , ...,  $\hat{Y}_{WNN}^{5f}(X)$  are the outputs of learned MS-W neuron models in consequent parts of  $R^1$  to  $R^5$ , respectively. The learned parameters  $C_S$ ,  $C_W$  and  $W^1, \dots, W^5$ , for  $R^1$  to  $R^5$ , are as follow.

$$C_S = \begin{bmatrix} 0.30781 & 0.72408 & 0.39893 \\ 0.56248 & 0.81377 & 0.40816 \\ 0.55692 & 0.21718 & 0.14845 \end{bmatrix} \quad C_W = \begin{bmatrix} 0.75462 & 0.58899 & 0.83937 \\ 0.52688 & 0.98164 & 0.22244 \\ 0.62958 & 0.17485 & 0.80545 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \\ W^4 \\ W^5 \end{bmatrix} = \begin{bmatrix} 0.1990 & 0.0922 & 0.0276 \\ 0.7190 & 0.8573 & 0.7294 \\ 0.3221 & -0.0464 & 0.4456 \\ 0.1704 & 0.5098 & 0.9124 \\ -0.3468 & 0.5898 & 1.2822 \end{bmatrix}$$

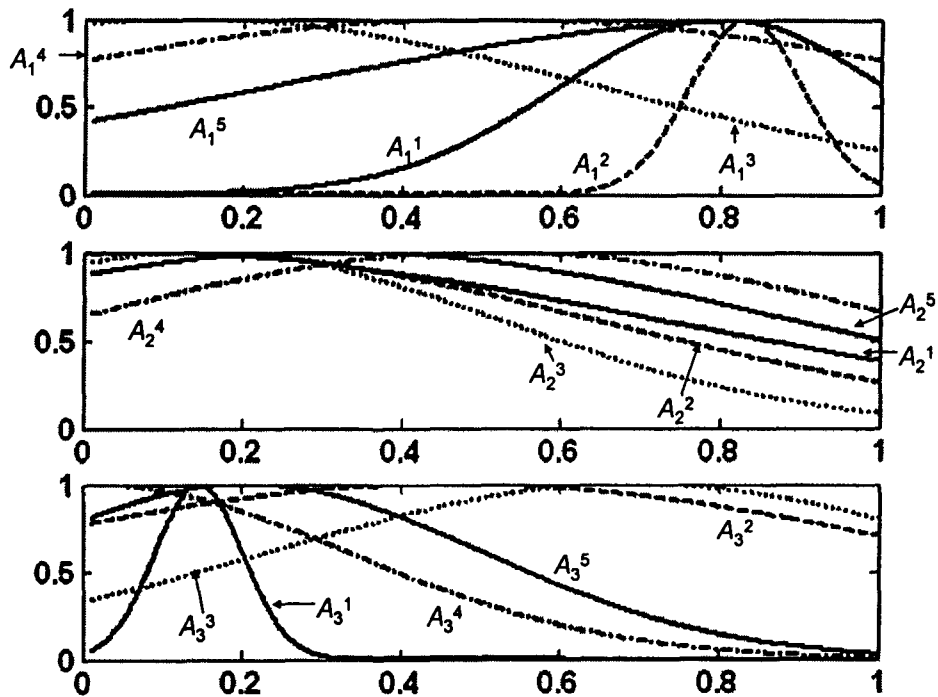


Fig. 5.4. Initialized membership functions, learned by GA, of the normalized inputs for Example 1

The learned premise variable membership functions  $A_1^{1f}$ - $A_1^{5f}$ ,  $A_2^{1f}$ - $A_2^{5f}$  and  $A_3^{1f}$ - $A_3^{5f}$  for inputs  $u(k)$ ,  $\hat{y}(k-1)$  and  $\hat{y}(k)$  are shown in Fig. 5.5. Figure 5.6 shows learning pattern of WNF model by Genetic Algorithm and Gradient Descent.

Figure 5.7 shows the actual output, WNF model output and model error with CS-P configuration. In this figure, actual output of the plant is solid line and the model output is dot line. The error also is solid line. The value of performance index  $J=1.6078 \times 10^{-7}$  for the WNF model obtained by CS-P configuration.

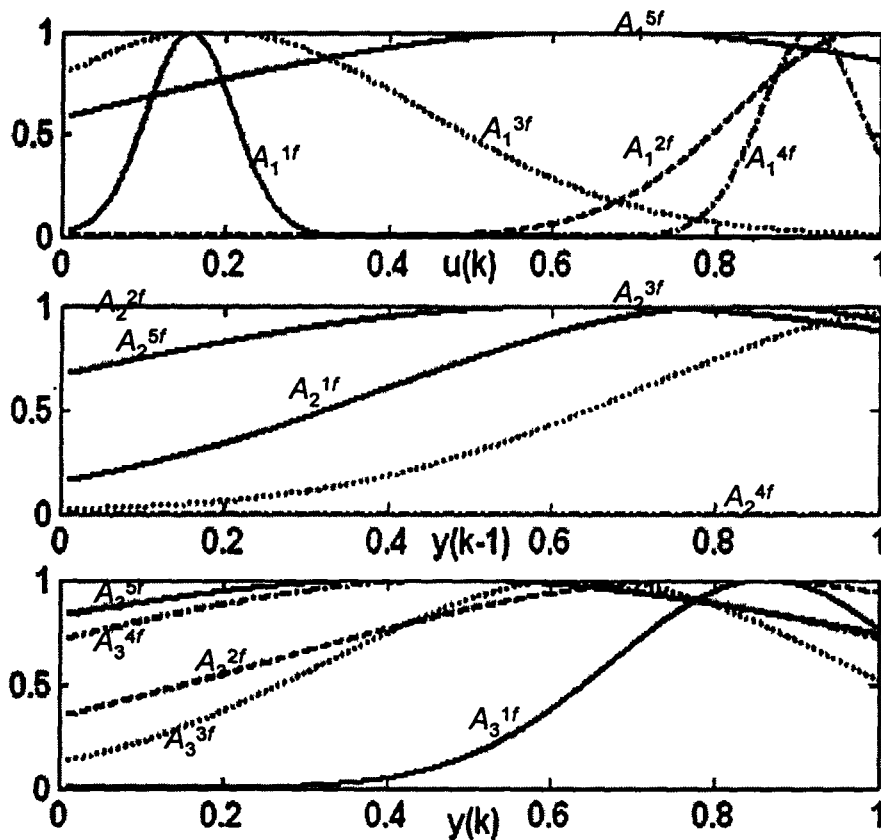


Fig. 5.5. Learned membership functions, obtained by GA & GD, of the normalized inputs for Example 1

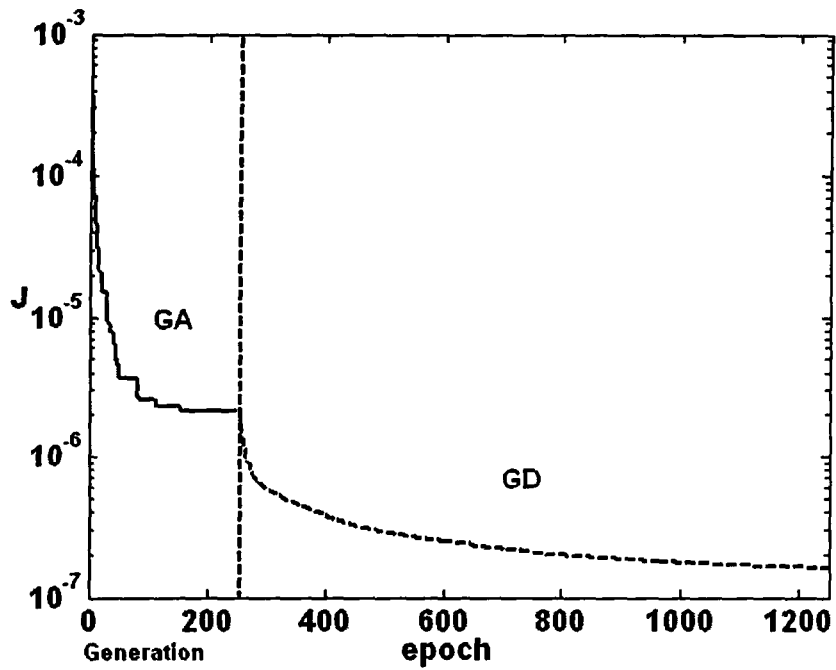


Fig. 5.6. Learning pattern of WNF model by Genetic Algorithm and Gradient Descent for Example 1

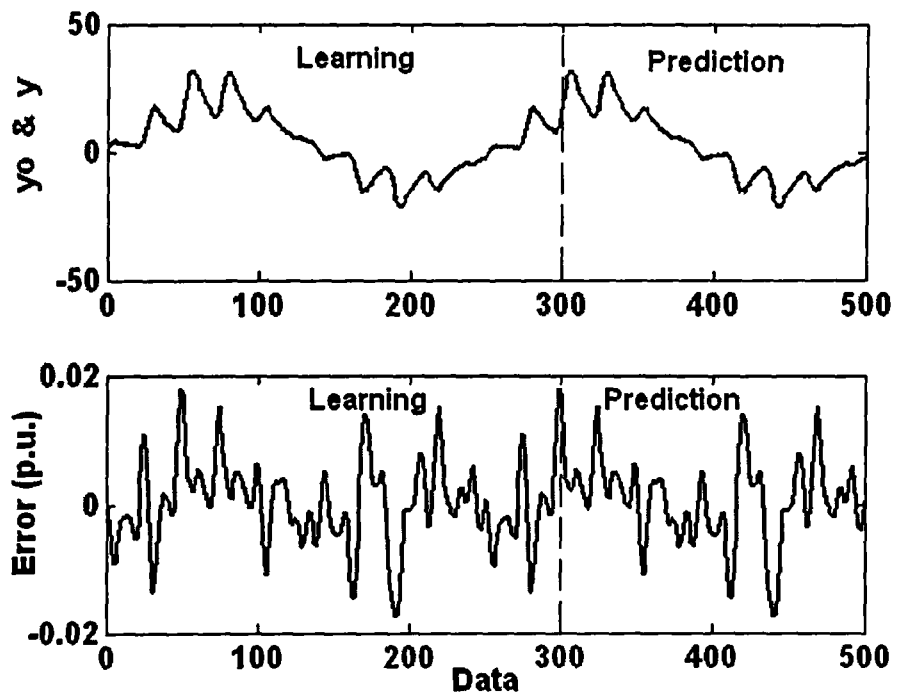


Fig. 5.7. Actual output & WNF model output and the error for Example 1

## Revisited Example 2: Non-linear regression with random input

Figure 5.8 shows the maximum fitness up to each generation. With five rules as mentioned in Revisited Example 2 in section 4-5, the initial solution for GD is obtained over 84 generations. The value of performance index  $J$ , initialize by GA, is  $3.7761 \times 10^{-5}$ . The initial fuzzy rules that generalize by GA are listed below:

$$R^1: \text{if } u(k) \text{ is } A_1^1 \wedge \hat{y}(k-1) \text{ is } A_2^1 \wedge \hat{y}(k) \text{ is } A_3^1 \text{ then } y^1(k+1) \text{ is } \hat{Y}_{WNN}^1(X)$$

$$R^2: \text{if } u(k) \text{ is } A_1^2 \wedge \hat{y}(k-1) \text{ is } A_2^2 \wedge \hat{y}(k) \text{ is } A_3^2 \text{ then } y^2(k+1) \text{ is } \hat{Y}_{WNN}^2(X)$$

$$R^3: \text{if } u(k) \text{ is } A_1^3 \wedge \hat{y}(k-1) \text{ is } A_2^3 \wedge \hat{y}(k) \text{ is } A_3^3 \text{ then } y^3(k+1) \text{ is } \hat{Y}_{WNN}^3(X)$$

$$R^4: \text{if } u(k) \text{ is } A_1^4 \wedge \hat{y}(k-1) \text{ is } A_2^4 \wedge \hat{y}(k) \text{ is } A_3^4 \text{ then } y^4(k+1) \text{ is } \hat{Y}_{WNN}^4(X)$$

where  $\hat{Y}_{WNN}^1(X)$ ,  $\hat{Y}_{WNN}^2(X)$ , ...,  $\hat{Y}_{WNN}^4(X)$  are the outputs of initialized MS-W neuron models by GA in consequent parts of  $R^1$  to  $R^4$ , respectively. Initialization of the learning parameters  $C_s$ ,  $C_w$  and  $W^1, \dots, W^4$ , for  $R^1$  to  $R^4$ , are as follows.

$$C_s = \begin{bmatrix} 0.93219 & 0.26576 & 0.40402 \\ 0.6945 & 0.72642 & 0.39803 \\ 0.62394 & 0.85308 & 0.75822 \\ 0.53891 & 0.53226 & 0.10993 \\ 0.096502 & 0.36611 & 0.51334 \\ 0.71281 & 0.18501 & 0.64408 \end{bmatrix} \quad C_w = \begin{bmatrix} 0.15223 & 0.059269 & 0.32979 \\ 0.091864 & 0.5911 & 0.50546 \\ 0.97296 & 0.38809 & 0.67765 \\ 0.22407 & 0.87127 & 0.70714 \\ 0.73784 & 0.78032 & 0.64164 \\ 0.23561 & 0.33492 & 0.2195 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \\ W^4 \end{bmatrix} = \begin{bmatrix} 0.0896 & 0.2914 & 0.1925 & 0.7222 & 0.1387 & 0.5118 \\ 0.1087 & 0.3501 & 0.1027 & 0.5249 & 0.9884 & 0.8257 \\ 0.8718 & 0.0204 & 0.3778 & 0.5875 & 0.4351 & 0.2690 \\ 0.2809 & 0.6385 & 0.1134 & 0.5698 & 0.8121 & 0.6706 \end{bmatrix}$$

The premise variable membership function  $A_1^1$ - $A_1^4$ ,  $A_2^1$ - $A_2^4$  and  $A_3^1$ - $A_3^4$  for inputs  $u(k)$ ,  $\hat{y}(k-1)$  and  $\hat{y}(k)$  are shown in Fig. 5.9.

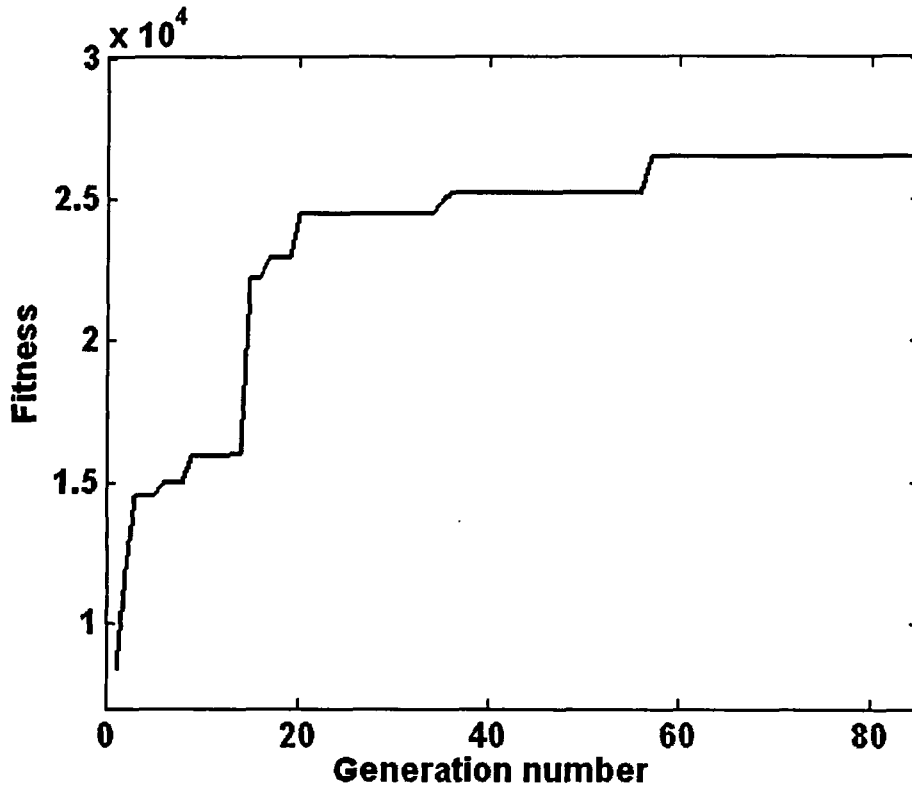


Fig. 5.8. Maximum fitness of GA up to each generation for Example 2

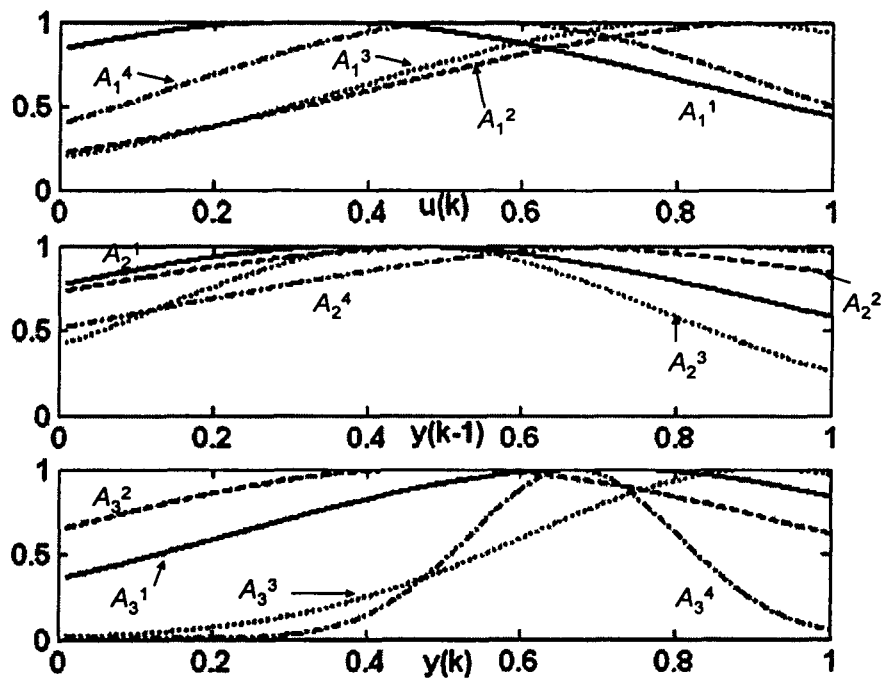


Fig. 5.9. Initialized membership functions, learned by GA, of the normalized inputs for Example 2



The fuzzy rules corresponding to the learned WNF are listed below:

$$R^{1f} : \text{if } u(k) \text{ is } A_1^{1f} \wedge \hat{y}(k-1) \text{ is } A_2^{1f} \wedge \hat{y}(k) \text{ is } A_3^{1f} \text{ then } y^1(k+1) \text{ is } \hat{Y}_{WNN}^{1f}(X)$$

$$R^{2f} : \text{if } u(k) \text{ is } A_1^{2f} \wedge \hat{y}(k-1) \text{ is } A_2^{2f} \wedge \hat{y}(k) \text{ is } A_3^{2f} \text{ then } y^2(k+1) \text{ is } \hat{Y}_{WNN}^{2f}(X)$$

$$R^{3f} : \text{if } u(k) \text{ is } A_1^{3f} \wedge \hat{y}(k-1) \text{ is } A_2^{3f} \wedge \hat{y}(k) \text{ is } A_3^{3f} \text{ then } y^3(k+1) \text{ is } \hat{Y}_{WNN}^{3f}(X)$$

$$R^{4f} : \text{if } u(k) \text{ is } A_1^{4f} \wedge \hat{y}(k-1) \text{ is } A_2^{4f} \wedge \hat{y}(k) \text{ is } A_3^{4f} \text{ then } y^4(k+1) \text{ is } \hat{Y}_{WNN}^{4f}(X)$$

where  $\hat{Y}_{WNN}^{1f}(X)$ ,  $\hat{Y}_{WNN}^{2f}(X)$ , ...,  $\hat{Y}_{WNN}^{4f}(X)$  are the outputs of learned MS-W neuron models in consequent parts of  $R^1$  to  $R^4$ , respectively. The learned parameters  $C_s$ ,  $C_w$  and  $W^1, \dots, W^4$ , for  $R^1$  to  $R^4$ , are as follow:

$$C_s = \begin{bmatrix} 0.41275 & 0.98765 & 0.65899 \\ 0.38067 & 0.67522 & 0.4018 \\ 0.73688 & 0.55452 & 0.83829 \\ 0.21594 & 0.71215 & 0.28372 \\ 0.095634 & 0.968 & 0.089004 \\ 0.31845 & 0.7872 & 0.38511 \end{bmatrix} \quad C_w = \begin{bmatrix} 0.28676 & 0.15502 & 1.0851 \\ 0.13362 & 1.2506 & 0.66065 \\ 0.21895 & 0.50995 & 0.90713 \\ -0.10607 & 0.042429 & 0.94806 \\ 0.23543 & 0.23625 & 0.82982 \\ 0.90667 & 0.6037 & 0.4772 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \\ W^4 \end{bmatrix} = \begin{bmatrix} 0.1203 & 0.9867 & 0.0945 & 0.8781 & 0.6765 & 0.8026 \\ 0.5563 & 1.0486 & 0.1301 & 0.7541 & 0.8585 & 0.9176 \\ -0.1189 & -0.0259 & 0.3733 & 0.4619 & 0.9993 & 0.0928 \\ 0.4445 & 0.9755 & 0.5684 & 0.3725 & 0.7328 & 0.1458 \end{bmatrix}$$

The learned premise variable membership function functions  $A_1^{1f}$ - $A_1^{4f}$ ,  $A_2^{1f}$ - $A_2^{4f}$  and  $A_3^{1f}$ - $A_3^{4f}$  for inputs  $u(k)$ ,  $\hat{y}(k-1)$  and  $\hat{y}(k)$  are shown in Fig. 5.10. Figure 5.11 shows learning pattern of WNF model by Genetic Algorithm and Gradient Descent. Figure 5.12 shows actual output and model output of the CS-P model. The error for learning and prediction section has been shown in Fig. 5.12. The value of performance Index  $J=7.9356 \times 10^{-7}$  is obtained for CS-P configuration.

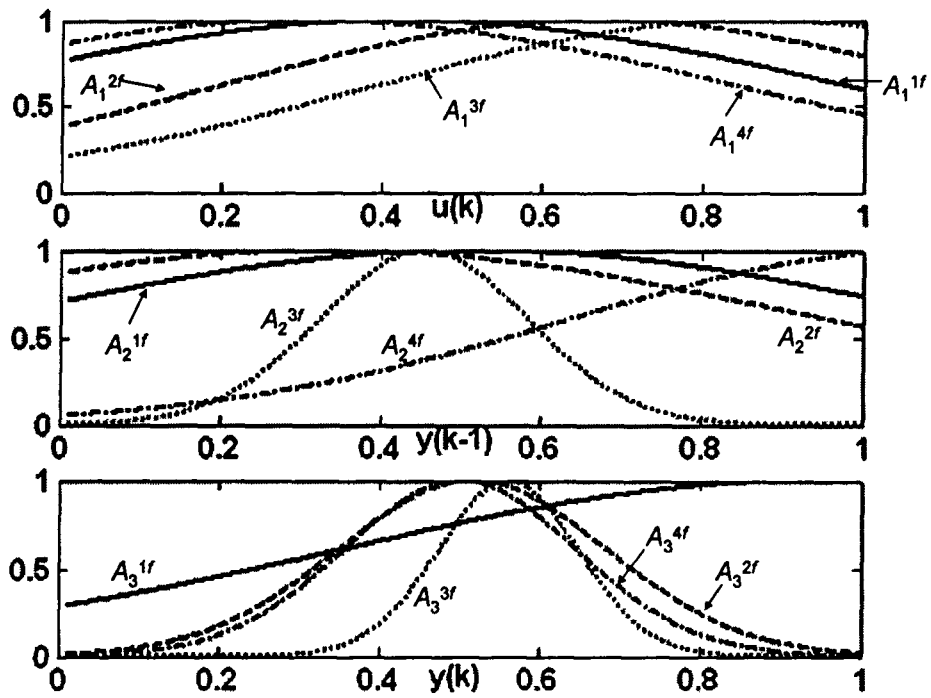


Fig. 5.10. Learned membership functions, obtained by GA & GD, of the normalized inputs for Example 2

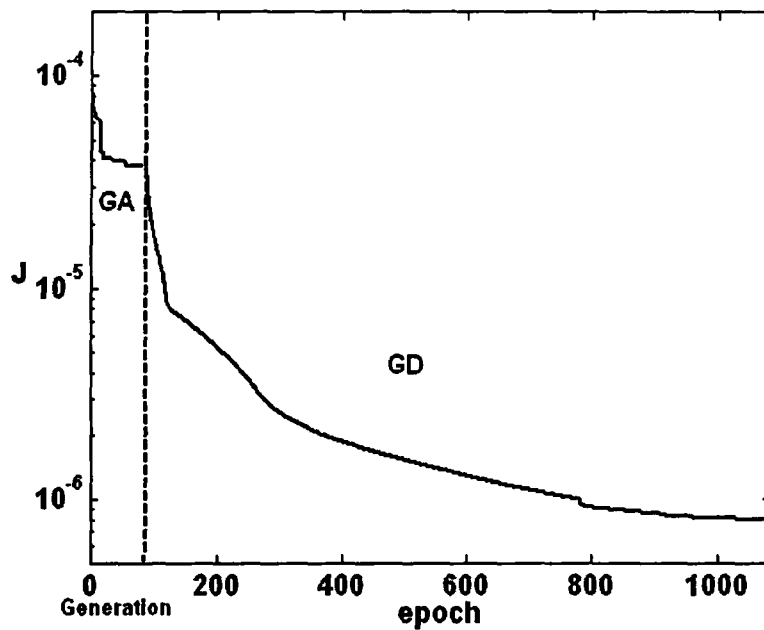


Fig. 5.11. Learning pattern of WNF model by Genetic Algorithm and Gradient Descent for Example 2

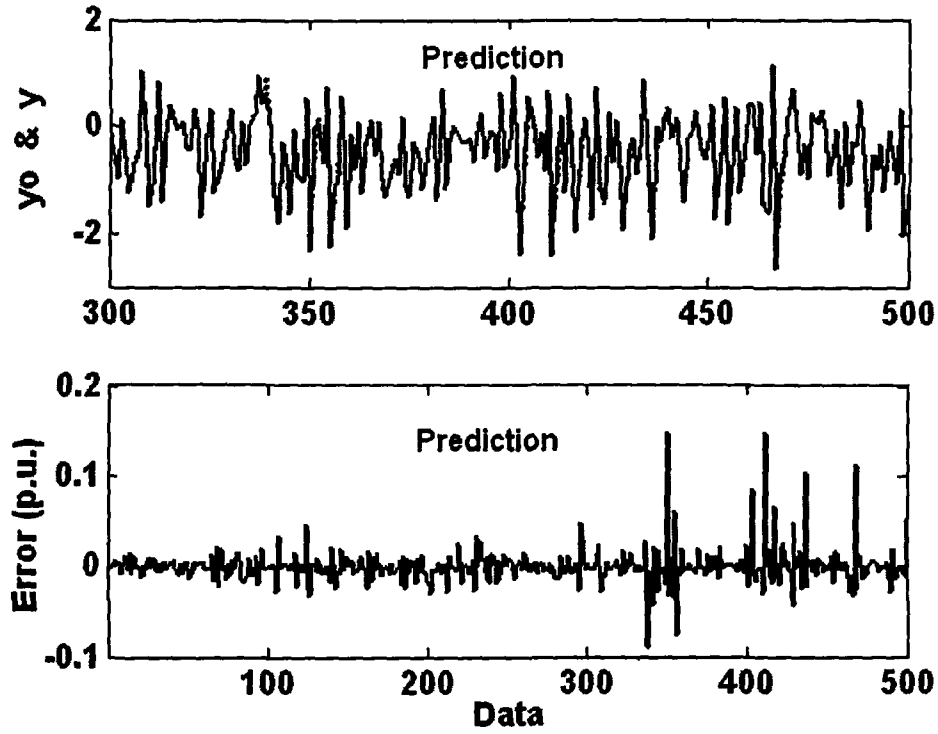


Fig. 5.12. Actual output & WNF model output and the error for Example 2

### Revisited Example 3: Non-Linear Regression with Non-Linear Input

Figure 5.13 shows the maximum fitness up to each generation. With three rules as mentioned in Revisited Example 3 in section 4-5, the initial solution for GD is obtained over 120 generations. The value of performance index  $J$  after initialization by GA is  $6.8541 \times 10^{-5}$ . The initial fuzzy rules generalized by GA are listed below:

$$R^1: \text{if } u(k) \text{ is } A_1^1 \wedge \hat{y}(k) \text{ is } A_2^1 \text{ then } y^1(k+1) \text{ is } \hat{Y}_{WNN}^1(X)$$

$$R^2: \text{if } u(k) \text{ is } A_1^2 \wedge \hat{y}(k) \text{ is } A_2^2 \text{ then } y^2(k+1) \text{ is } \hat{Y}_{WNN}^2(X)$$

$$R^3: \text{if } u(k) \text{ is } A_1^3 \wedge \hat{y}(k) \text{ is } A_2^3 \text{ then } y^3(k+1) \text{ is } \hat{Y}_{WNN}^3(X)$$

where  $\hat{Y}_{WNN}^1(X)$ ,  $\hat{Y}_{WNN}^2(X)$  and  $\hat{Y}_{WNN}^3(X)$  are the outputs of initialized MS-W neuron models by GA in consequent parts of  $R^1$  to  $R^3$ , respectively. Initialization of the learning parameters  $C_S$ ,  $C_W$  and  $W^1, \dots, W^3$ , for  $R^1$  to  $R^3$ , are as follow.

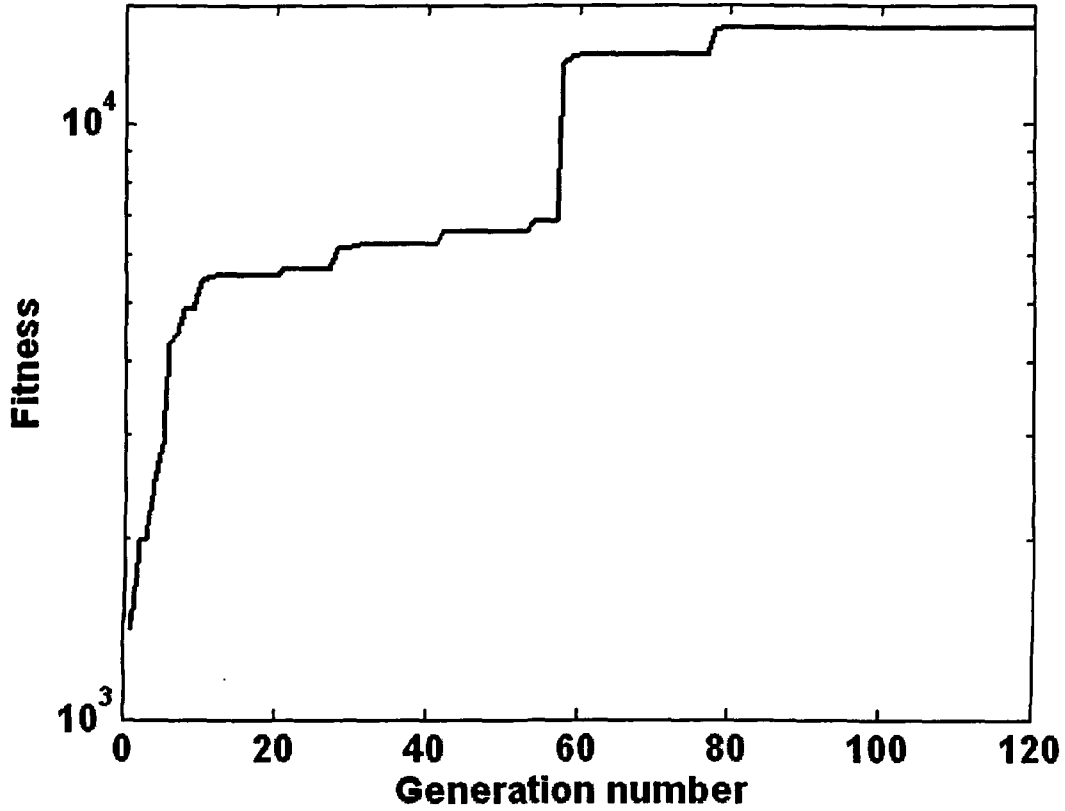


Fig. 5.13. Maximum fitness of GA up to each generation for Example 3

$$C_S = \begin{bmatrix} 0.66001 & 0.58115 \\ 0.95581 & 0.6237 \\ 0.11402 & 0.81176 \end{bmatrix}$$

$$C_W = \begin{bmatrix} 0.96533 & 0.39193 \\ 0.74907 & 0.82067 \\ 0.42654 & 0.6591 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \end{bmatrix} = \begin{bmatrix} 0.3183 & 0.1168 & 0.8820 \\ 0.3587 & 0.7858 & 0.6068 \\ 0.3324 & 0.9688 & 0.4925 \end{bmatrix}$$

The premise variable membership function  $A_1^1-A_1^3$  and  $A_2^1-A_2^3$  for inputs  $u(k)$  and  $\hat{y}(k)$  are shown in Fig. 5.14. The fuzzy rules corresponding to the learned WNF are listed below:

$$R^{1f} : \text{if } u(k) \text{ is } A_1^{1f} \wedge \hat{y}(k) \text{ is } A_2^{1f} \text{ then } y^1(k+1) \text{ is } \hat{Y}_{WNN}^{1f}(X)$$

$$R^{2f} : \text{if } u(k) \text{ is } A_1^{2f} \wedge \hat{y}(k) \text{ is } A_2^{2f} \text{ then } y^2(k+1) \text{ is } \hat{Y}_{WNN}^{2f}(X)$$

$$R^{3f} : \text{if } u(k) \text{ is } A_1^{3f} \wedge \hat{y}(k) \text{ is } A_2^{3f} \text{ then } y^3(k+1) \text{ is } \hat{Y}_{WNN}^{3f}(X)$$

where  $\hat{Y}_{WNN}^{1f}(X)$ ,  $\hat{Y}_{WNN}^{2f}(X)$  and  $\hat{Y}_{WNN}^{3f}(X)$  are the outputs of learned MS-W neuron models in consequent parts of  $R^1$  to  $R^3$ , respectively. Initialization of the learning parameters  $C_s$ ,  $C_w$  and  $W^1, \dots, W^3$ , for  $R^1$  to  $R^3$ , are as follow:

$$C_s = \begin{bmatrix} 0.65531 & 0.58598 \\ 0.95526 & 0.62217 \\ 0.11587 & 0.77571 \end{bmatrix} \quad C_w = \begin{bmatrix} 1.0845 & 0.27657 \\ 0.76702 & 0.83148 \\ 0.3453 & 0.37021 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \end{bmatrix} = \begin{bmatrix} 0.2809 & 0.0518 & 0.8880 \\ 0.8177 & 1.0447 & 1.1052 \\ -0.0105 & 0.6968 & 0.7751 \end{bmatrix}$$

The learned premise variable membership functions  $A_1^{1f}-A_1^{3f}$  and  $A_2^{1f}-A_2^{3f}$  for inputs  $u(k)$  and  $\hat{y}(k)$  are shown in Fig. 5.15. Figure 5.16 shows learning pattern of WNF model by Genetic Algorithm and Gradient Descent. Figure 5.17 shows actual output and model output of the CS-P model. The error for learning and prediction section is shown in Fig. 5.17. The value of performance Index  $J=2.7545 \times 10^{-6}$  is obtained for CS-P configuration.

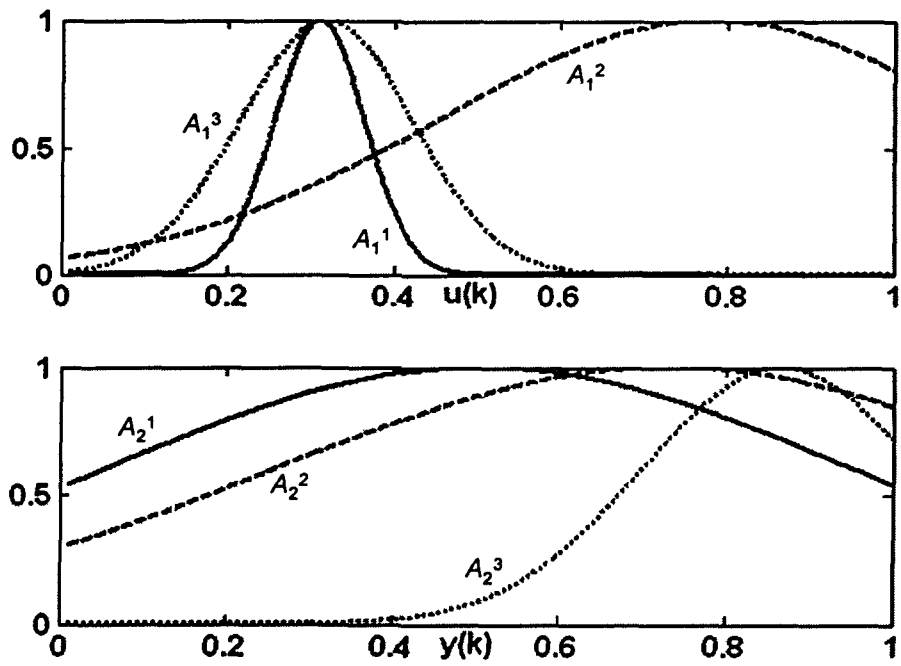


Fig. 5.14. Initialized membership functions, learned by GA, of the normalized inputs for Example 3

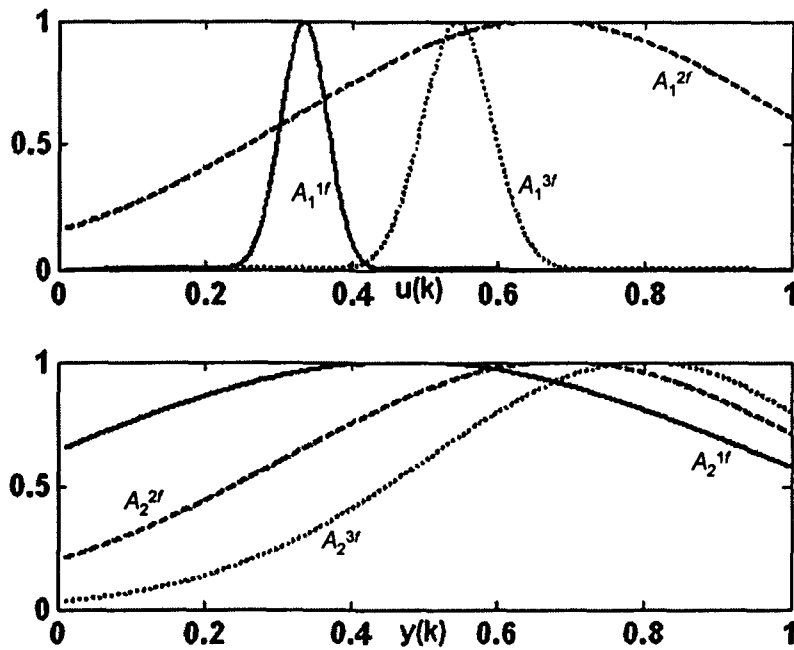


Fig. 5.15. Learned membership functions, obtained by GA & GD, of the normalized inputs for Example 3

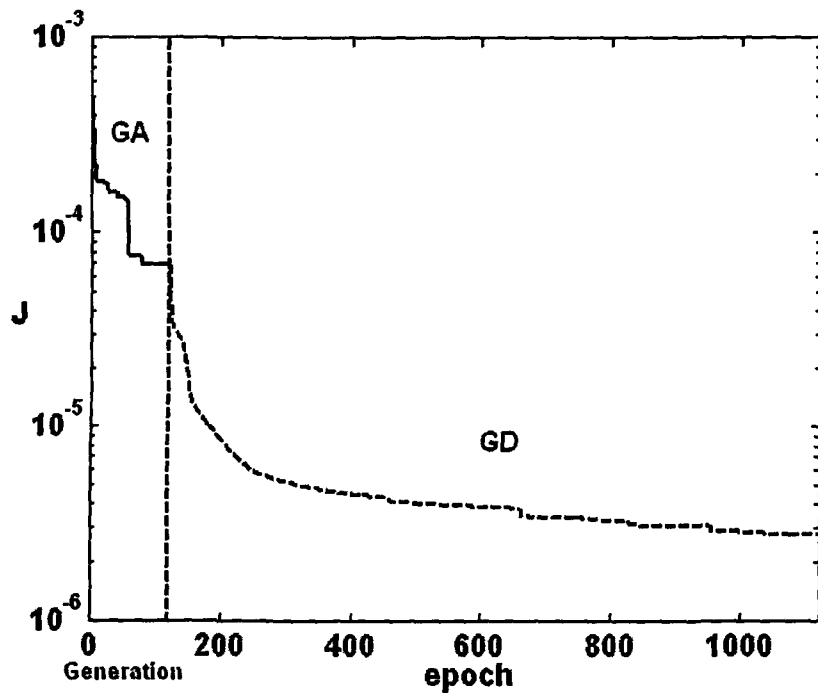


Fig. 5.16. Learning pattern of WNF model by Genetic Algorithm and Gradient Descent for Example 3

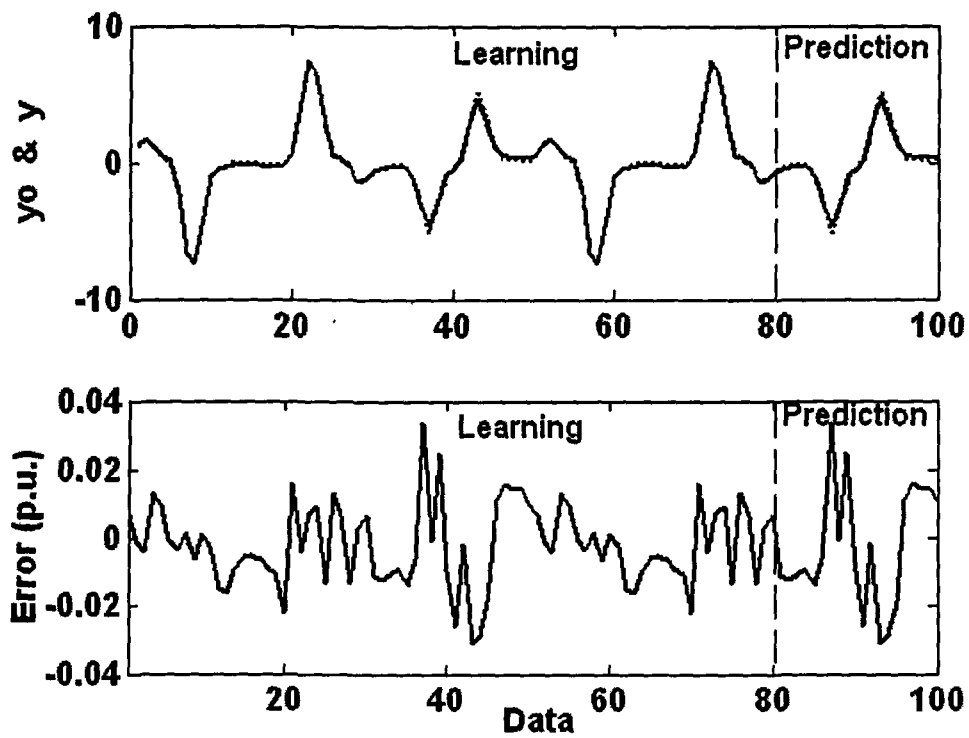


Fig. 5.17. Actual output & WNF model output and the error for Example 3

**Revisited Example 4: Non-linear Regression of Input and output**

Figure 5.18 shows the maximum fitness up to each generation. The initial solution for GD is obtained over 208 generation with three rules as mentioned in Revisited Example 4 in section 4-5. The value of performance index  $J$  after initialization by GA is  $4.3790 \times 10^{-6}$ . The initial fuzzy rules that generalize by GA are listed below:

$$R^1: \text{if } u(k) \text{ is } A_1^1 \wedge \hat{y}(k) \text{ is } A_2^1 \text{ then } y^1(k+1) \text{ is } \hat{Y}_{WNN}^1(X)$$

$$R^2: \text{if } u(k) \text{ is } A_1^2 \wedge \hat{y}(k) \text{ is } A_2^2 \text{ then } y^2(k+1) \text{ is } \hat{Y}_{WNN}^2(X)$$

$$R^3: \text{if } u(k) \text{ is } A_1^3 \wedge \hat{y}(k) \text{ is } A_2^3 \text{ then } y^3(k+1) \text{ is } \hat{Y}_{WNN}^3(X)$$

where  $\hat{Y}_{WNN}^1(X)$ ,  $\hat{Y}_{WNN}^2(X)$  and  $\hat{Y}_{WNN}^3(X)$  are the outputs of initialized MS-W neuron models by GA in consequent parts of  $R^1$  to  $R^3$ , respectively.

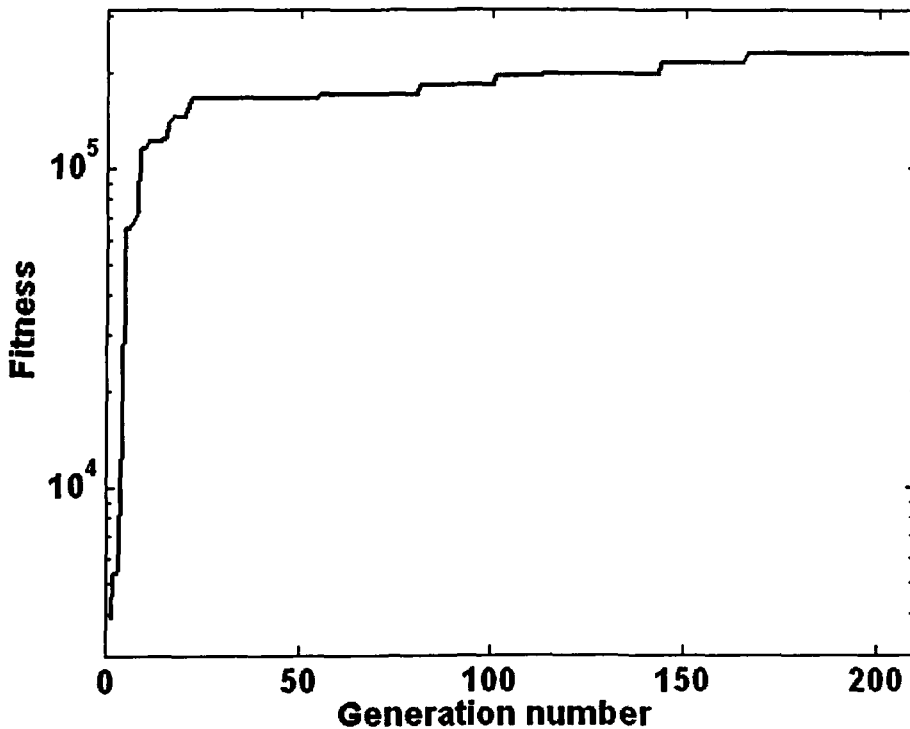


Fig. 5.18. Maximum fitness of GA up to each generation for Example 4



Initialization of the learning parameters  $C_S$ ,  $C_W$  and  $W^1, \dots, W^3$ , for  $R^1$  to  $R^3$ , are as follow:

$$C_S = \begin{bmatrix} 0.17347 & 0.25368 \\ 0.81066 & 0.34084 \\ 0.63248 & 0.74944 \end{bmatrix} \quad C_W = \begin{bmatrix} 0.84472 & 0.13111 \\ 0.33315 & 0.83605 \\ 0.95483 & 0.10688 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \end{bmatrix} = \begin{bmatrix} 0.6327 & 0.0317 & 0.7779 \\ 0.6670 & 0.5752 & 0.8146 \\ 0.0078 & 0.5692 & 0.9209 \end{bmatrix}$$

The premise variable membership function  $A_1^1$ - $A_1^3$  and  $A_2^1$ - $A_2^3$  for inputs  $u(k)$  and  $\hat{y}(k)$  are shown in Fig. 5.19. The fuzzy rules corresponding to the learned WNF are listed below:

$$R^{1f} : \text{if } u(k) \text{ is } A_1^{1f} \wedge \hat{y}(k) \text{ is } A_2^{1f} \text{ then } y^1(k+1) \text{ is } \hat{Y}_{WNN}^{1f}(X)$$

$$R^{2f} : \text{if } u(k) \text{ is } A_1^{2f} \wedge \hat{y}(k) \text{ is } A_2^{2f} \text{ then } y^2(k+1) \text{ is } \hat{Y}_{WNN}^{2f}(X)$$

$$R^{3f} : \text{if } u(k) \text{ is } A_1^{3f} \wedge \hat{y}(k) \text{ is } A_2^{3f} \text{ then } y^3(k+1) \text{ is } \hat{Y}_{WNN}^{3f}(X)$$

where  $\hat{Y}_{WNN}^{1f}(X)$ ,  $\hat{Y}_{WNN}^{2f}(X)$  and  $\hat{Y}_{WNN}^{3f}(X)$  are the outputs of learned MS-W neuron models in consequent parts of  $R^1$  to  $R^3$ , respectively.

Initialization of the learning parameters  $C_S$ ,  $C_W$  and  $W^1, \dots, W^3$ , for  $R^1$  to  $R^3$ , are as follow:

$$C_S = \begin{bmatrix} 0.026068 & 0.63253 \\ 0.18761 & 0.67029 \\ 0.076676 & 0.90617 \end{bmatrix} \quad C_W = \begin{bmatrix} 0.15438 & 0.91355 \\ 0.19916 & 0.71218 \\ 0.61996 & 0.28396 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \end{bmatrix} = \begin{bmatrix} 0.4932 & 0.0088 & 0.5814 \\ 0.4242 & 0.8924 & 0.9438 \\ 0.2665 & 0.2291 & 1.0572 \end{bmatrix}$$

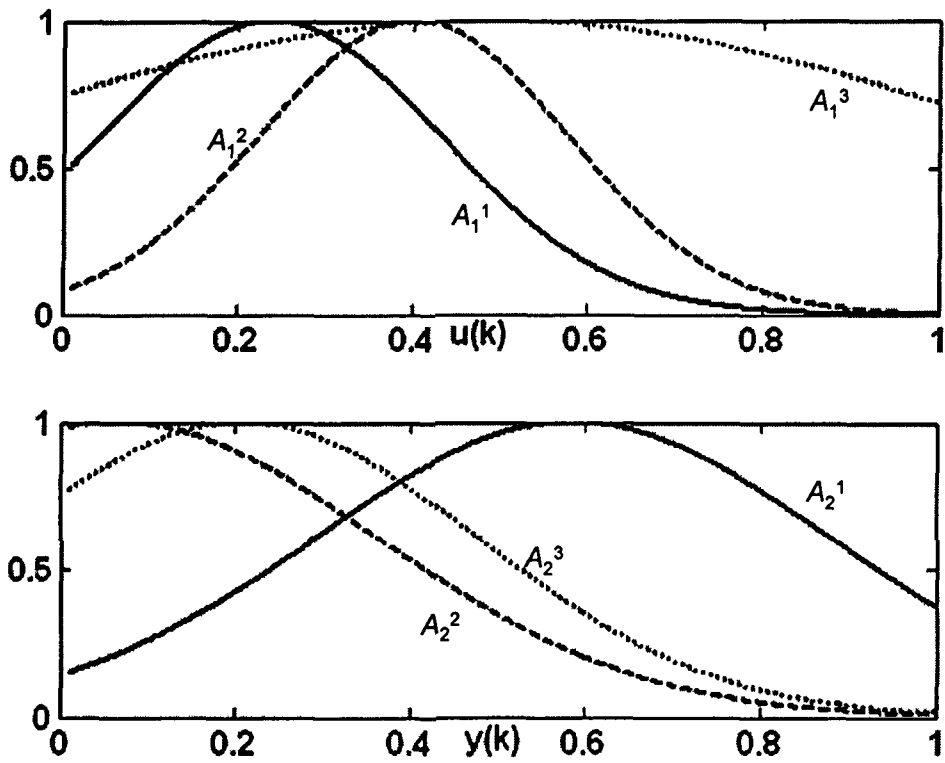


Fig. 5.19. Initialized membership functions, learned by GA, of the normalized inputs for Example 4

The learned premise variable membership functions  $A_1^{1f}$ - $A_1^{3f}$  and  $A_2^{1f}$ - $A_2^{3f}$  for inputs  $u(k)$  and  $\hat{y}(k)$  are shown in Fig. 5.20. Figure 5.21 shows learning pattern of WNF model by Genetic Algorithm and Gradient Descent. The models have been learned with all (one thousand) data to identify output  $y(k+1)$ . Figure 5.22 shows actual output and model output of the CS-P model. The error for learning and prediction section is shown in Fig. 5.22. The value of performance Index  $J=3.3841 \times 10^{-6}$  is obtained for CS-P configuration.

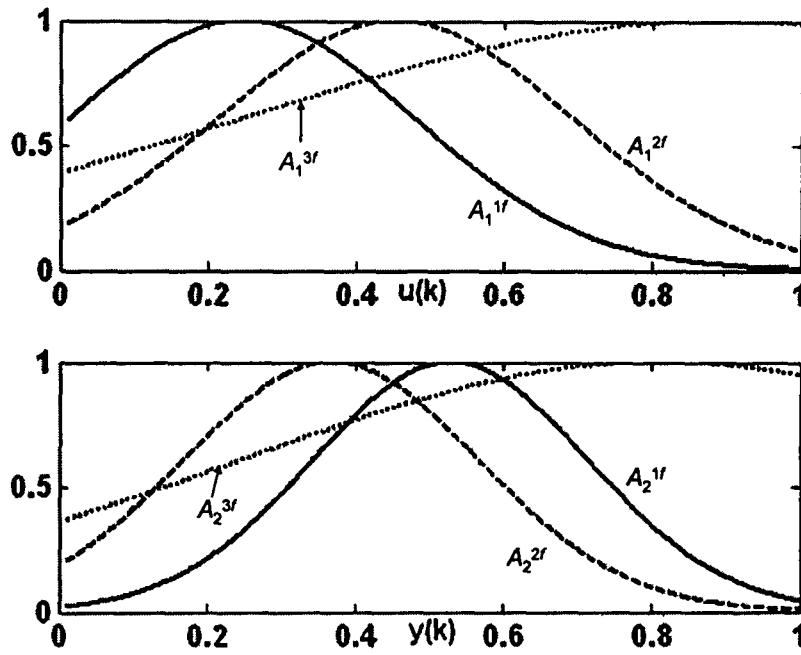


Fig. 5.20. Learned membership functions, obtained by GA & GD, of the normalized inputs for Example 4

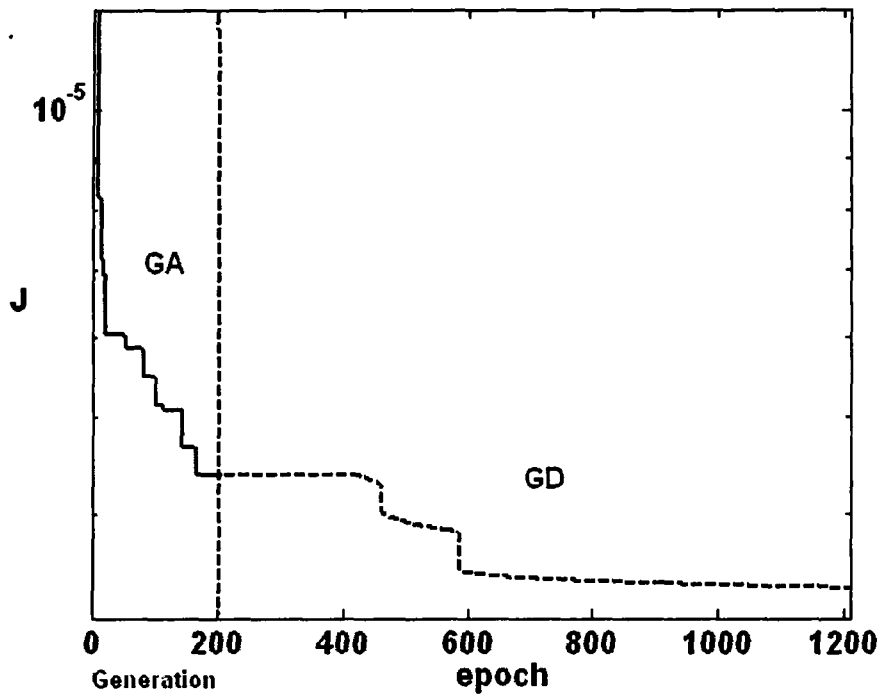


Fig. 5.21. Learning pattern of WNF model by Genetic Algorithm and Gradient Descent for Example 4

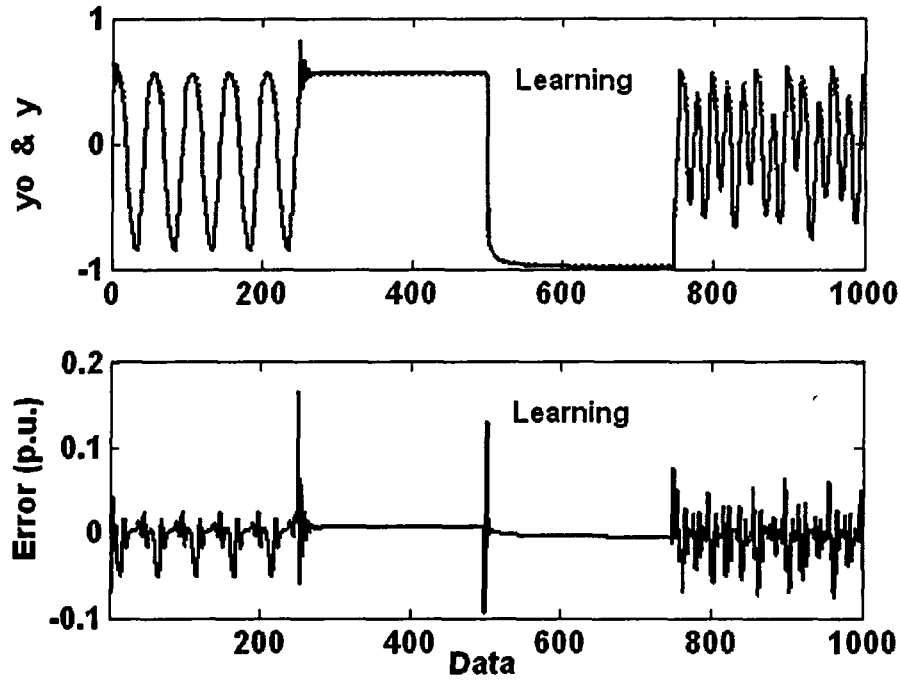


Fig. 5.22. Actual output & WNF model output and the error for Example 4

### Revisited Example 5: Gas Furnace Data

Figure 5.23 shows the maximum fitness up to each generation. With five rules as mentioned in Revisited Example 5 in section 4-5, the initial solution for GD is obtained over 118 generations. The value of performance index  $J$  after initialization by GA is  $1.2569 \times 10^{-7}$ . The initial fuzzy rules that generalize by GA are listed below:

$$R^1: \text{if } u(k-2) \text{ is } A_1^1 \wedge u(k-3) \text{ is } A_2^1 \wedge \hat{y}(k-1) \text{ is } A_3^1 \text{ then } y^1(k) \text{ is } \hat{Y}_{WNN}^1(X)$$

$$R^2: \text{if } u(k-2) \text{ is } A_1^2 \wedge u(k-3) \text{ is } A_2^2 \wedge \hat{y}(k-1) \text{ is } A_3^2 \text{ then } y^2(k) \text{ is } \hat{Y}_{WNN}^2(X)$$

$$R^3: \text{if } u(k-2) \text{ is } A_1^3 \wedge u(k-3) \text{ is } A_2^3 \wedge \hat{y}(k-1) \text{ is } A_3^3 \text{ then } y^3(k) \text{ is } \hat{Y}_{WNN}^3(X)$$

$$R^4: \text{if } u(k-2) \text{ is } A_1^4 \wedge u(k-3) \text{ is } A_2^4 \wedge \hat{y}(k-1) \text{ is } A_3^4 \text{ then } y^4(k) \text{ is } \hat{Y}_{WNN}^4(X)$$

$$R^5: \text{if } u(k-2) \text{ is } A_1^5 \wedge u(k-3) \text{ is } A_2^5 \wedge \hat{y}(k-1) \text{ is } A_3^5 \text{ then } y^5(k) \text{ is } \hat{Y}_{WNN}^5(X)$$

where  $\hat{Y}_{WNN}^1(X)$ ,  $\hat{Y}_{WNN}^2(X)$ , ...,  $\hat{Y}_{WNN}^5(X)$  are the outputs of initialized MS-W neuron models by

GA in consequent parts of  $R^1$  to  $R^5$ , respectively. Initialization of the learning parameters  $C_S$ ,  $C_W$

and  $W^1, \dots, W^5$ , for  $R^1$  to  $R^5$ , are as follow:

$$C_s = \begin{bmatrix} 0.84557 & 0.34688 & 0.21425 \\ 0.22682 & 0.52268 & 0.81884 \\ 0.89416 & 0.96521 & 0.022035 \end{bmatrix} \quad C_w = \begin{bmatrix} 0.42105 & 0.19557 & 0.09284 \\ 0.76817 & 0.018312 & 0.98541 \\ 0.577 & 0.82293 & 0.34646 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \\ W^4 \\ W^5 \end{bmatrix} = \begin{bmatrix} 0.5491 & 0.7375 & 0.1560 \\ 0.6525 & 0.6169 & 0.8344 \\ 0.1843 & 0.6985 & 0.2811 \\ 0.0581 & 0.1842 & 0.6572 \\ 0.1355 & 0.9379 & 0.8700 \end{bmatrix}$$

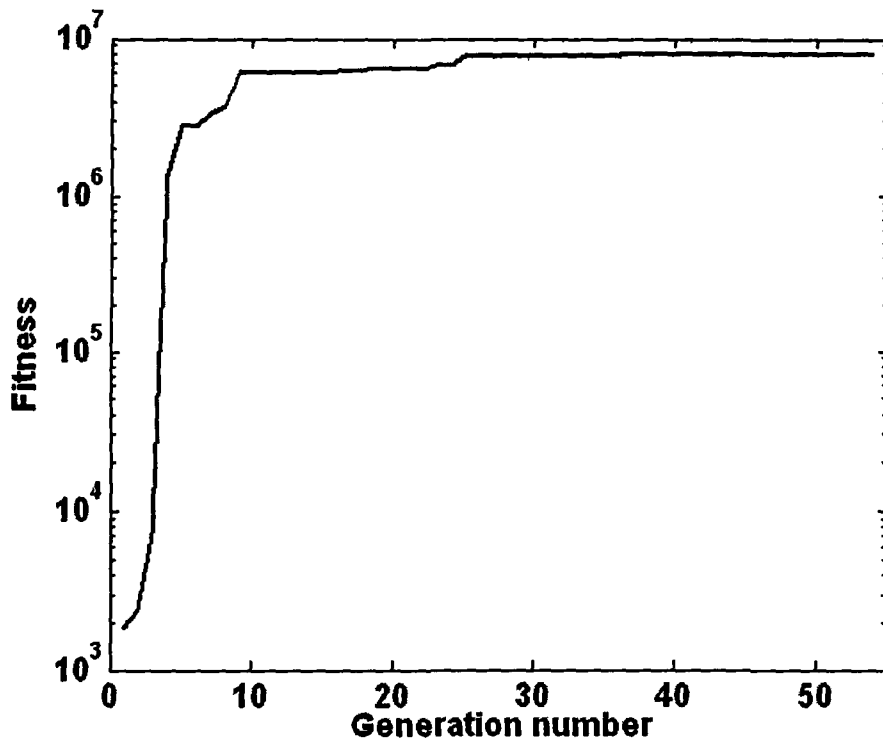


Fig. 5.23. Maximum fitness of GA up to each generation for Example 5

The premise variable membership function  $A_1^1-A_1^5$ ,  $A_2^1-A_2^5$  and  $A_3^1-A_3^5$  for inputs  $u(k-2)$ ,  $u(k-3)$  and  $\hat{y}(k-1)$  are shown in Fig. 5.24. The fuzzy rules corresponding to the learned WNF are listed below:

- $R^{1f}$  : if  $u(k-2)$  is  $A_1^{1f} \wedge u(k-3)$  is  $A_2^{1f} \wedge \hat{y}(k-1)$  is  $A_3^{1f}$  then  $y^1(k)$  is  $\hat{Y}_{WNN}^{1f}(X)$   
 $R^{2f}$  : if  $u(k-2)$  is  $A_1^{2f} \wedge u(k-3)$  is  $A_2^{2f} \wedge \hat{y}(k-1)$  is  $A_3^{2f}$  then  $y^2(k)$  is  $\hat{Y}_{WNN}^{2f}(X)$   
 $R^{3f}$  : if  $u(k-2)$  is  $A_1^{3f} \wedge u(k-3)$  is  $A_2^{3f} \wedge \hat{y}(k-1)$  is  $A_3^{3f}$  then  $y^3(k)$  is  $\hat{Y}_{WNN}^{3f}(X)$   
 $R^{4f}$  : if  $u(k-2)$  is  $A_1^{4f} \wedge u(k-3)$  is  $A_2^{4f} \wedge \hat{y}(k-1)$  is  $A_3^{4f}$  then  $y^4(k)$  is  $\hat{Y}_{WNN}^{4f}(X)$   
 $R^{5f}$  : if  $u(k-2)$  is  $A_1^{5f} \wedge u(k-3)$  is  $A_2^{5f} \wedge \hat{y}(k-1)$  is  $A_3^{5f}$  then  $y^5(k)$  is  $\hat{Y}_{WNN}^{5f}(X)$

where  $\hat{Y}_{WNN}^{1f}(X)$ ,  $\hat{Y}_{WNN}^{2f}(X)$ , ...,  $\hat{Y}_{WNN}^{5f}(X)$  are the outputs of learned MS-W neuron models in consequent parts of  $R^1$  to  $R^5$ , respectively.

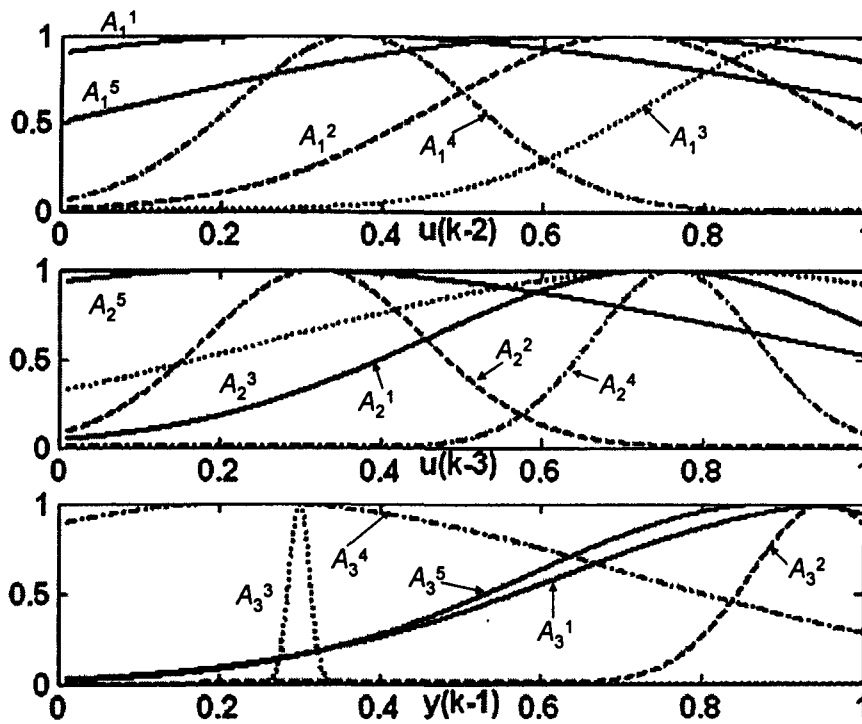


Fig. 5.24. Initialized membership functions, learned by GA, of the normalized inputs for Example 5

Initialization of the learning parameters  $C_S$ ,  $C_W$  and  $W^1, \dots, W^5$ , for  $R^1$  to  $R^5$ , are as follow:

$$C_S = \begin{bmatrix} 0.84612 & 0.34695 & 0.21303 \\ 0.22651 & 0.52271 & 0.81928 \\ 0.89424 & 0.96529 & 0.022302 \end{bmatrix} \quad C_W = \begin{bmatrix} 0.41147 & 0.18949 & 0.082668 \\ 0.76163 & 0.014421 & 1.0007 \\ 0.57856 & 0.82369 & 0.34636 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \\ W^4 \\ W^5 \end{bmatrix} = \begin{bmatrix} 0.5456 & 0.7397 & 0.1621 \\ 0.6521 & 0.6135 & 0.8289 \\ 0.1843 & 0.6985 & 0.2811 \\ 0.0581 & 0.1842 & 0.6572 \\ 0.0994 & 0.9778 & 0.9129 \end{bmatrix}$$

The learned premise variable membership function functions  $A_1^{1f}$ - $A_1^{5f}$ ,  $A_2^{1f}$ - $A_2^{5f}$  and  $A_3^{1f}$ - $A_3^{5f}$  for inputs  $u(k-2)$ ,  $u(k-3)$  and  $\hat{y}(k-1)$  are shown in Fig. 5.25. Figure 5.26 shows learning pattern of WNF model by Genetic Algorithm and Gradient Descent. The models have been learned with all (one thousand) data, to identify output  $y(k+1)$ . Figure 5.27 shows actual output and model output of the CS-P model. The error for learning and prediction section has been shown in Fig. 5.27. The value of performance Index  $J=9.5489 \times 10^{-8}$  is obtained for CS-P configuration.

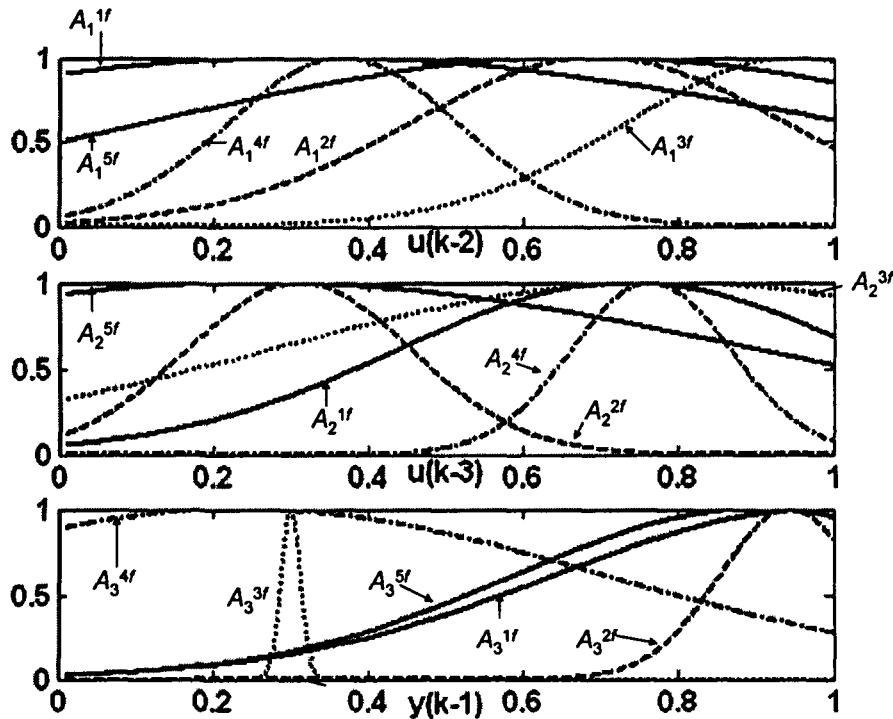


Fig. 5.25. Learned membership functions, obtained by GA & GD, of the normalized inputs for Example 5

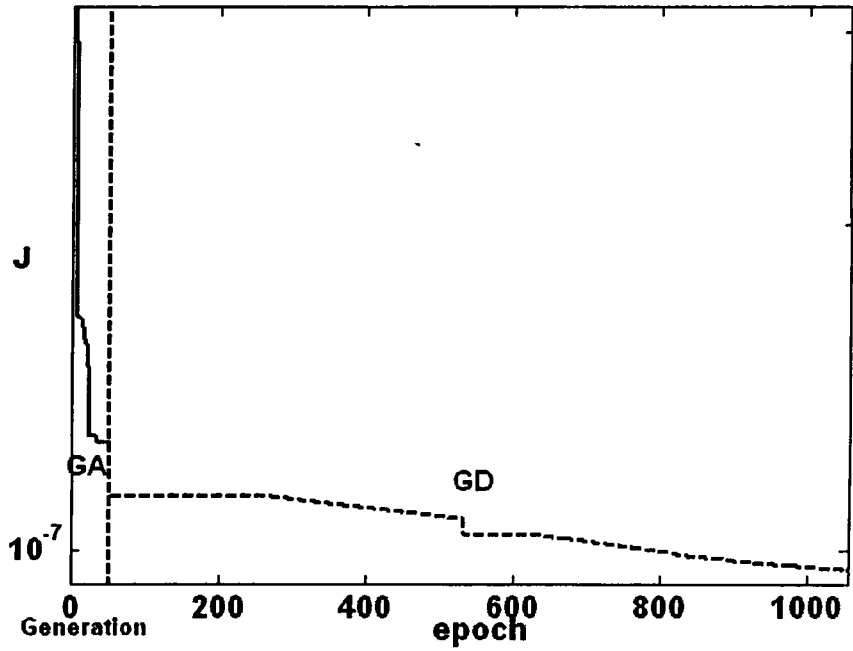


Fig. 5.26. Learning pattern of WNF model by Genetic Algorithm and Gradient Descent for Example 5

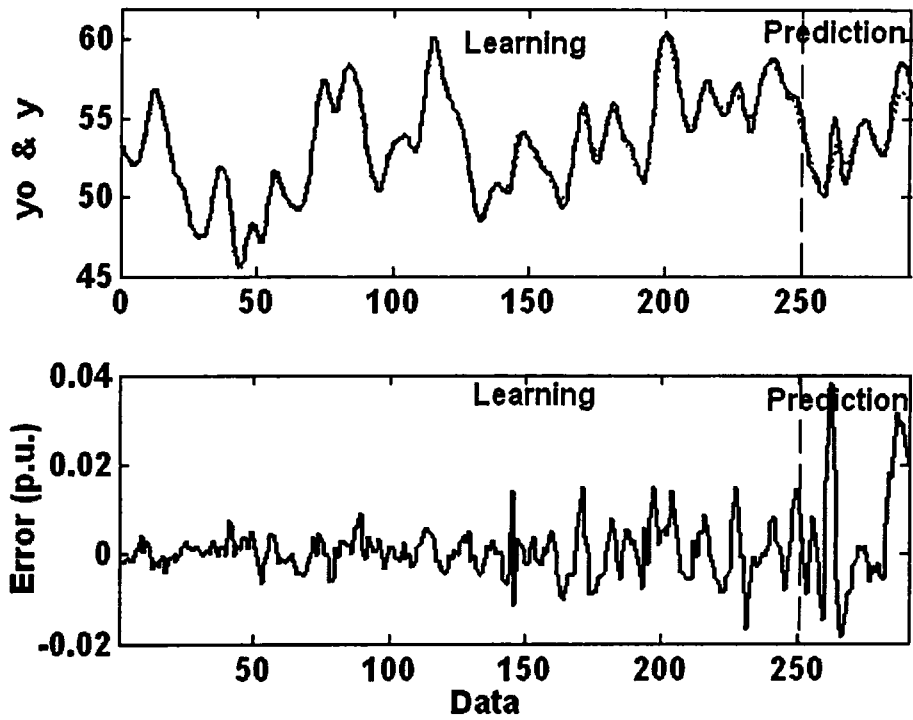


Fig. 5.27. Actual output & WNF model output and the error for Example 5



### Revisited Example 6: Human Operation at a Chemical Plant

S-P configuration is selected to learn the learning parameters of the WNF model. Figure 5.28 shows the maximum fitness up to each generation. By applying modified mountain clustering and cluster validity function, three rules are generated. The value of performance index  $J$  is  $6.7911 \times 10^{-6}$  after initialization by GA over 135 generations. The initial fuzzy rules generalized by GA and S-P configuration are listed below:

$$R^1: \text{if } u_1(k) \text{ is } A_1^1 \wedge u_3(k) \text{ is } A_2^1 \text{ then } y^1(k+1) \text{ is } \hat{Y}_{WNN}^1(X)$$

$$R^2: \text{if } u_1(k) \text{ is } A_1^2 \wedge u_3(k) \text{ is } A_2^2 \text{ then } y^2(k+1) \text{ is } \hat{Y}_{WNN}^2(X)$$

$$R^3: \text{if } u_1(k) \text{ is } A_1^3 \wedge u_3(k) \text{ is } A_2^3 \text{ then } y^3(k+1) \text{ is } \hat{Y}_{WNN}^3(X)$$

where  $\hat{Y}_{WNN}^1(X)$ ,  $\hat{Y}_{WNN}^2(X)$  and  $\hat{Y}_{WNN}^3(X)$  are the outputs of initialized MS-W neuron models by GA in consequent parts of  $R^1$  to  $R^3$ , respectively. Initialization of the learning parameters  $C_s$ ,  $C_w$  and  $W^1, \dots, W^3$ , for  $R^1$  to  $R^3$ , are as follow:

$$C_s = \begin{bmatrix} 0.57029 & 0.93597 \\ 0.08844 & 0.50082 \\ 0.56852 & 0.38607 \\ 0.60258 & 0.24153 \\ 0.86175 & 0.34633 \\ 0.32271 & 0.41873 \end{bmatrix} \quad C_w = \begin{bmatrix} 0.088689 & 0.31038 \\ 0.66819 & 0.47977 \\ 0.92346 & 0.41842 \\ 0.9140 & 0.44003 \\ 0.31936 & 0.34145 \\ 0.57572 & 0.39694 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \end{bmatrix} = \begin{bmatrix} 0.1673 & 0.4499 & 0.2877 & 0.6862 & 0.8148 & 0.4063 \\ 0.0559 & 0.3691 & 0.4679 & 0.1667 & 0.9771 & 0.2477 \\ 0.7635 & 0.3202 & 0.2123 & 0.5041 & 0.7642 & 0.7494 \end{bmatrix}$$

The premise variable membership function  $A_1^1$ - $A_1^3$  and  $A_2^1$ - $A_2^3$  for inputs  $u_1(k)$  and  $u_3(k)$  are shown in Fig. 5.29. The fuzzy rules corresponding to the learned WNF and S-P configuration are listed below:

$$\begin{aligned}
R^{1f} &: \text{if } u_1(k) \text{ is } A_1^{1f} \wedge u_3(k) \text{ is } A_2^{1f} \text{ then } y^1(k+1) \text{ is } \hat{Y}_{WNN}^{1f}(X) \\
R^{2f} &: \text{if } u_1(k) \text{ is } A_1^{2f} \wedge u_3(k) \text{ is } A_2^{2f} \text{ then } y^2(k+1) \text{ is } \hat{Y}_{WNN}^{2f}(X) \\
R^{3f} &: \text{if } u_1(k) \text{ is } A_1^{3f} \wedge u_3(k) \text{ is } A_2^{3f} \text{ then } y^3(k+1) \text{ is } \hat{Y}_{WNN}^{3f}(X)
\end{aligned}$$

where  $\hat{Y}_{WNN}^{1f}(X)$ ,  $\hat{Y}_{WNN}^{2f}(X)$  and  $\hat{Y}_{WNN}^{3f}(X)$  are the outputs of learned MS-W neuron models in consequent parts of  $R^1$  to  $R^3$ , respectively. Initialization of the learning parameters  $C_S$ ,  $C_W$  and  $W^1, \dots, W^3$ , for  $R^1$  to  $R^3$ , are as follow:

$$C_S = \begin{bmatrix} 0.57029 & 0.93597 \\ 0.08844 & 0.50083 \\ 0.56852 & 0.38607 \\ 0.60258 & 0.24154 \\ 0.86175 & 0.34634 \\ 0.32271 & 0.41872 \end{bmatrix} \qquad C_W = \begin{bmatrix} 0.088689 & 0.31036 \\ 0.66819 & 0.47974 \\ 0.92346 & 0.41844 \\ 0.9140 & 0.44002 \\ 0.31936 & 0.34149 \\ 0.57572 & 0.39693 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \end{bmatrix} = \begin{bmatrix} 0.1674 & 0.4500 & 0.2877 & 0.6862 & 0.8148 & 0.4062 \\ 0.0813 & 0.3595 & 0.4747 & 0.1559 & 0.9985 & 0.2587 \\ 0.7635 & 0.3202 & 0.2123 & 0.5041 & 0.7642 & 0.7494 \end{bmatrix}$$

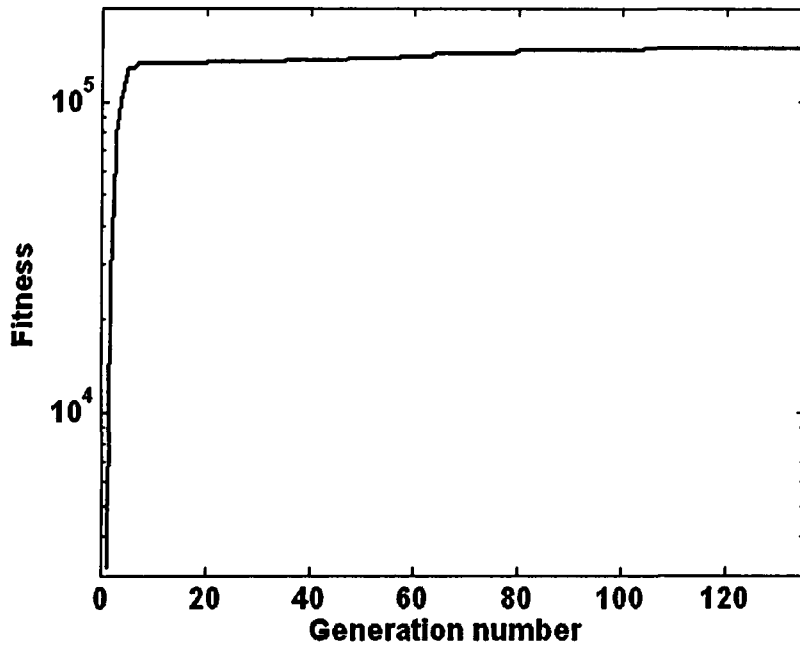


Fig. 5.28. Maximum fitness of GA up to each generation for Example 6

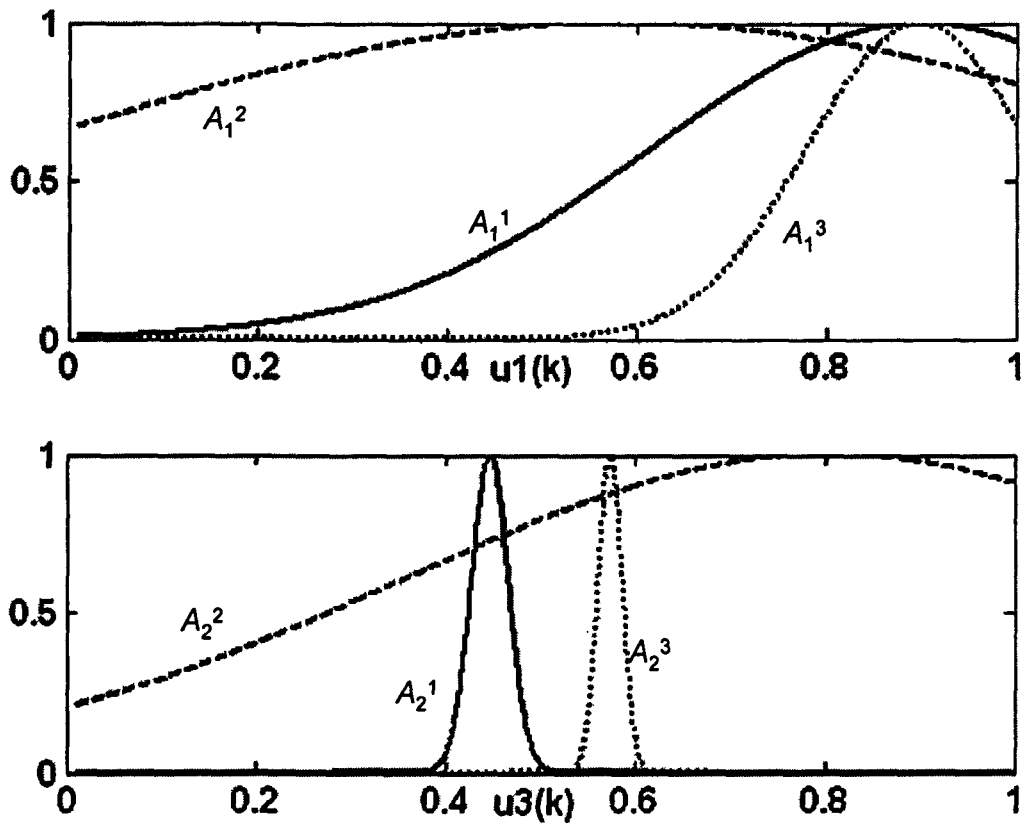


Fig. 5.29. Initialized membership functions, learned by GA, of the normalized inputs for Example 6

The learned premise variable membership function  $A_1^{1f} - A_1^{3f}$  and  $A_2^{1f} - A_2^{3f}$  for inputs  $u_1(k)$  and  $u_3(k)$  are shown in Fig. 5.30. Figure 5.31 shows learning pattern of WNF model by Genetic Algorithm and Gradient Descent. The models have been learned with all (one thousand) data, to identify output  $y(k+1)$ . Figure 5.32 shows actual output and model output of the CS-P model. The error for learning and prediction section is shown in Fig. 5.32. The value of performance Index  $J = 6.7658 \times 10^{-6}$  is obtained for CS-P configuration.

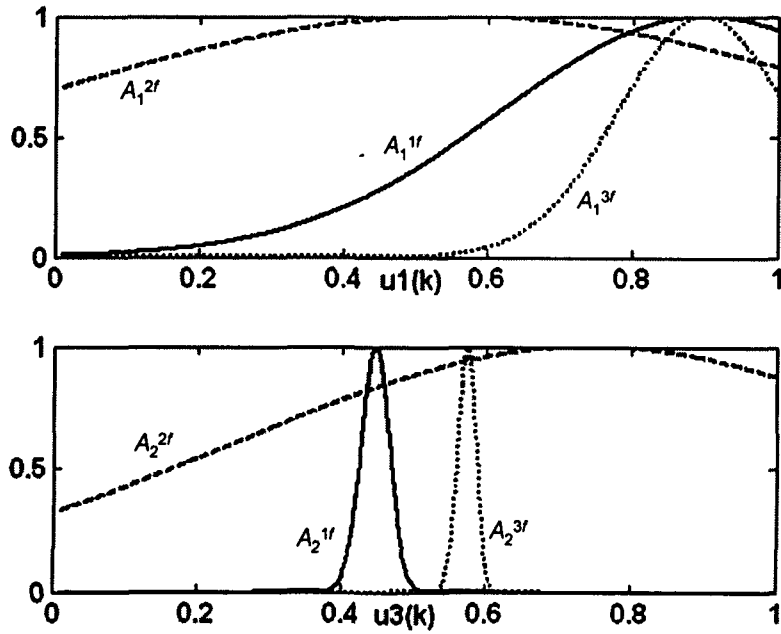


Fig. 5.30. Learned membership functions, obtained by GA & GD, of the normalized inputs for Example 6

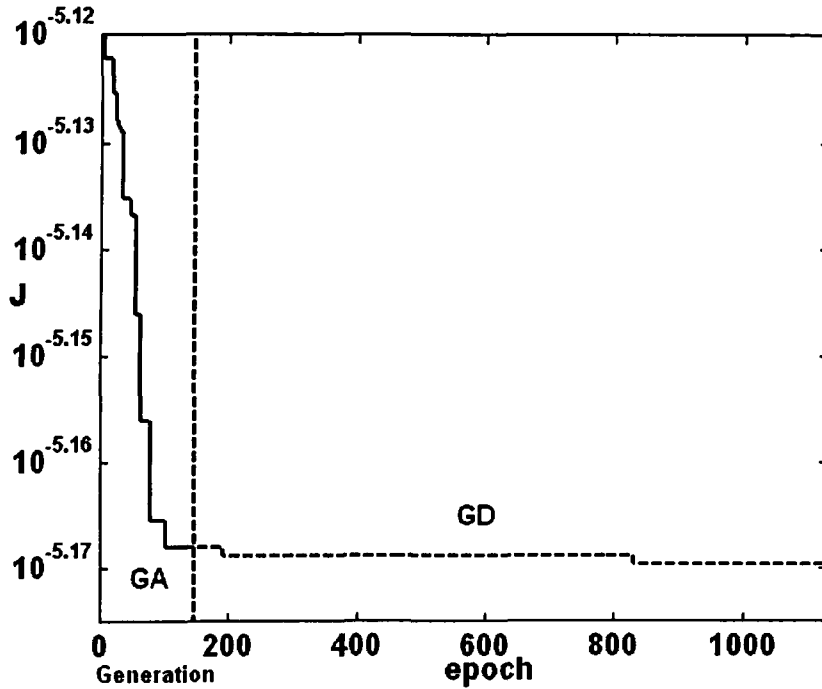


Fig. 5.31. Learning pattern of WNF model by Genetic Algorithm and Gradient Descent for Example 6

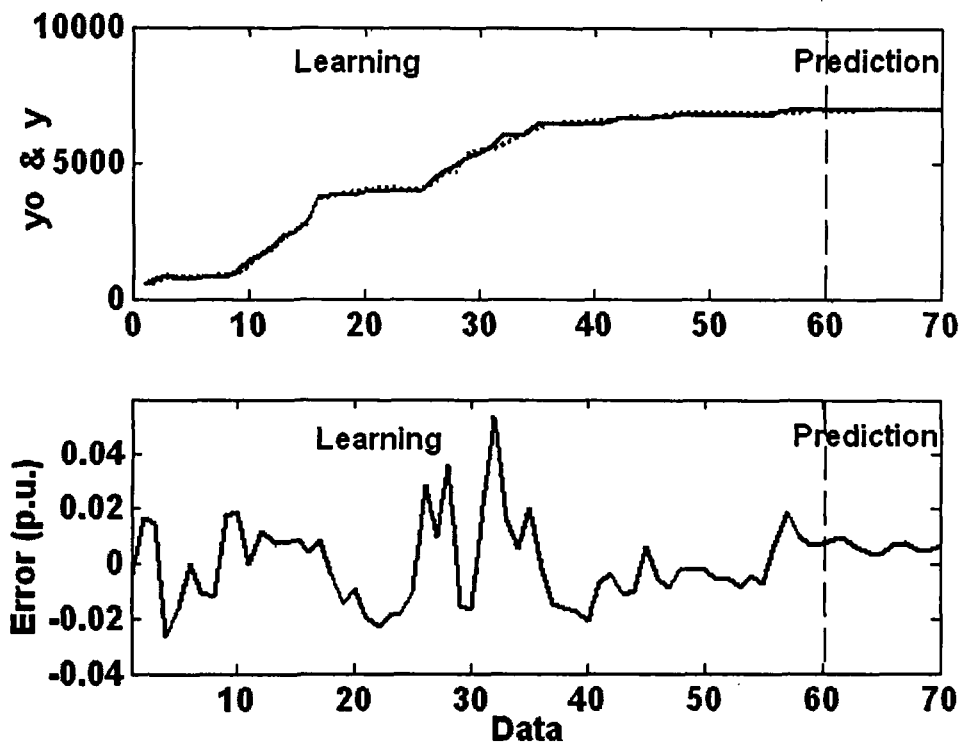


Fig. 5.32. Actual output & WNF model output and the error for Example 6

Table 5.1: Performance Index ( $J$ ) of MS-W neuron model, NF and WNF models

	MS-W	NF	WNF
	Performance Index ( $J$ )	Performance Index ( $J$ )	Performance Index ( $J$ )
<b>Example 1</b>	$7.585 \times 10^{-7}$	$1.8694 \times 10^{-7}$	<b><math>1.6078 \times 10^{-7}</math></b>
<b>Example 2</b>	$1.145 \times 10^{-5}$	$1.1040 \times 10^{-6}$	<b><math>7.9356 \times 10^{-7}</math></b>
<b>Example 3</b>	$1.361 \times 10^{-4}$	$3.2015 \times 10^{-6}$	<b><math>2.7545 \times 10^{-6}</math></b>
<b>Example 4</b>	$6.394 \times 10^{-6}$	$5.3610 \times 10^{-6}$	<b><math>3.3841 \times 10^{-6}</math></b>
<b>Example 5</b>	$1.665 \times 10^{-7}$	$6.8899 \times 10^{-8}$	$9.5489 \times 10^{-8}$
<b>Example 6</b>	$7.479 \times 10^{-6}$	$9.1412 \times 10^{-6}$	<b><math>6.7658 \times 10^{-6}</math></b>

Table 5.1 shows the performance index J for three models: MS-W neuron model, Neuro-Fuzzy (NF) and Wavelet Neuro-Fuzzy (WNF) model. In this table, the model with the better performance, for each example, is Bold.

## 5-5 Conclusions

In this chapter based on MS-W neuron model that yields better result than SS-W neuron model, as discussed in chapter 2, WNF model is proposed. The consequent parts of each rule in WNF model is localized by MS-W neuron model. The proposed neuro-fuzzy model is initialized by GA and to learn learning parameter of the proposed model, CS-P configuration which had better performance in chapter 3, is applied. The proposed model examine by six dynamic examples.

Table 5.1 shows that fuzzy models have better performance than MS-W neuron model. The propose WNF model also mostly yields better performance than TSK neuro-fuzzy model.

# Chapter 6

## Recurrent Wavelet Networks

### 6-1 Introduction

Science has evolved from trying to understand and predict the behavior of the universe and systems within it. Much of this owes to the development of suitable models, which is in conformity with the observations. These models are in symbolic form, which the humans use, and in mathematical form that are found from physical laws. Most systems are causal, which can be categorized as either static where the output depends on the current inputs, or dynamic where the output not only depends on the current inputs, but also on the past inputs and outputs. Many systems also possess unobservable inputs, which cannot be measured, but they affect the system's output, i.e., time series system. These inputs are known as disturbances and aggravate the modeling process.

To cope with the complexity of dynamic systems, there have been significant developments in the field of Artificial Neural Network (ANN), applied for identification and modeling, during last three decades [Narendra'90, Zhang'92]. One major application for proposing the different types of the network is to predict the dynamic behavior of many systems

in nature. ANN is a powerful method in approximation of nonlinear system and mapping between input-output data [Narendra'90].

Due to the dynamic behavior of recurrent network, they are suitable in dealing with the modeling of dynamic systems as compared to static behavior of feed-forward network [Qin'92, Li'05]. Owing to the above fact, the presented work proposes different types of recurrent neurons. Since the proposed neurons are used in feed-forward network making them as recurrent network. It has already been shown that the recurrent networks are less sensitive to noise with relatively smaller network size and simpler structure. Their long-term prediction property makes the recurrent network much powerful in dealing with dynamic systems. Recurrent networks are less sensitive to noise because, the recurrent network could recognize and generate periodic waves in spite of the existence of a large number of noises. This means that the network is able to regenerate the original periodic waves in the process of learning the teachers' signals with noises [Qin'92]. For unknown dynamic system, recurrent network results in a smaller size network as compared to feed-forward network [Li'05, Juang'02]. For time-series modeling, it generates a simpler structure [Lee'00, Mastorocostas'02, Yoo'06] and gives long-term prediction [Mastorocostas'02, Barbounis'06]. Because of dynamic behavior of recurrent networks, they are suitable in dealing with the modeling of dynamic systems as compared to static behavior of feed-forward network [Serinivasan'94, Lee'00]. Recurrent network for system modeling, learn and memorize information in terms of embedded weights [Lee'00].

Each neuron model in the proposed SS-W and MS-W neuron models, as discussed in chapter 2, comprise Sigmoid Activation Function (SAF) and Wavelet Activation Function (WAF). Morlet wavelet function that yield better result with both SS-W and MS-W neuron model is applied to recurrent model. In the series of development of different recurrent network



and neuron model, the presented work is an attempt to proposed different type of recurrent neuron model. Several types of recurrent network can be introduced by combining the output from SAF and WAF with product operator. In Sigmoid-Recurrent Wavelet (S-RW), the output of the wavelet function feedbacks to itself. When the output of Sigmoid function feedbacks to itself, it results a Recurrent Sigmoid–Wavelet (RS-W). In Feedback to Sigmoid from Wavelet (FS-W) neuron, the output of wavelet function, feedback to sigmoid function within a neuron unit. When output of sigmoid function feedbacks to wavelet function, within a neuron unit, it is called as Feedback to Wavelet from Sigmoid (FW-S). A Recurrent Neuron (RN) is also proposed in which the output of the neuron unit is feedback to itself. The idea of using different neuron models is to introduce recurrent network in modeling of dynamic systems and to perform a comparative study of recurrent neuron models consist of sigmoid and wavelet activation function in feed-forward neural network architecture. In proposed recurrent neurons, the SAF can administer the system dynamics by inputting the delayed output of wavelets to it. On the other hand, the delayed output of SAF feedback to WAF ascertains that dynamic of the system is accumulated in WAF. However, when the output of the recurrent neurons feedback to both wavelet and SAF, dynamics of the plant is attributed to SAF and WAF together. Since the convergence analysis plays an important role in the recurrent networks, the Lyapunov stability approach is employed to guarantee the convergence of network.

In this chapter, section 6-2, presents structure of the proposed recurrent S-W neuron models. Universal approximation theories of the proposed recurrent neuron models are described in section 6-3. Gradient Descent learning of learning parameters is draw up in section 6-3. Simulation result is given in section 6-5 and finally the conclusions are relegated in section 6-6.

## 6-2 Structures of Recurrent Neuron Models

In this section, based on proposed SS-W and MS-W neuron model (Fig. 2.7, Fig. 2.8 and Fig. 2.9), five types of recurrent networks are introduced for each SS-W and MS-W neuron model. In the proposed recurrent networks, outputs of SAF and WAF are feedback. Structure of the proposed networks is presented in this section.

### 6-2.1 Sigmoid-Recurrent Wavelet (S-RW) Neuron

In this neuron model, the output of WAF is feedback to itself. Parameter  $Q_w$  is feedback weight for first order dynamic of network. The dynamics of WAF is given in (6.1) and output of SAF is same as given in (3.21). Figure 6.1 and Fig. 6.2 show the architecture of Summation S-RW (SS-RW) and Multiplication S-RW (MS-RW) neuron models, respectively.

$$y_j^\psi(k) = \psi \left( Q_w^j \cdot y_j^\psi(k-1) + \sum_{i=1}^n C_{w_i}^j \cdot x_i(k) \right) \quad (6.1)$$

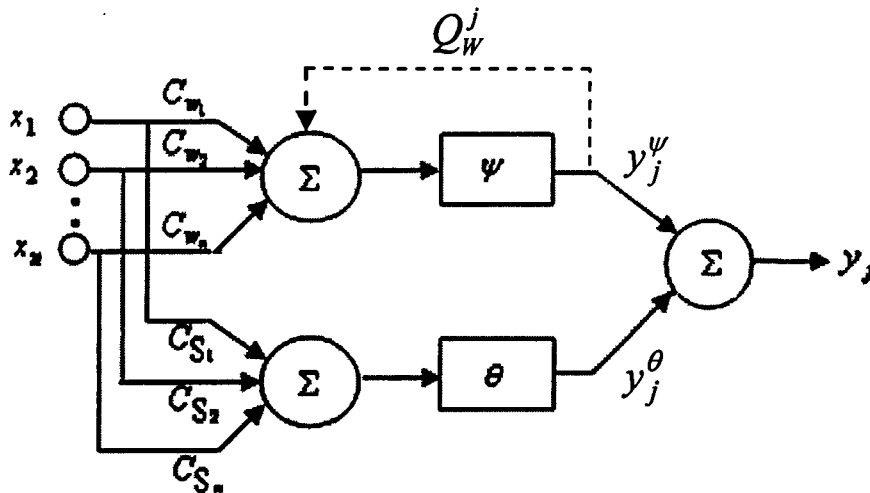


Fig. 6.1. Summation Sigmoid-Recurrent Wavelet (SS-RW) neuron model

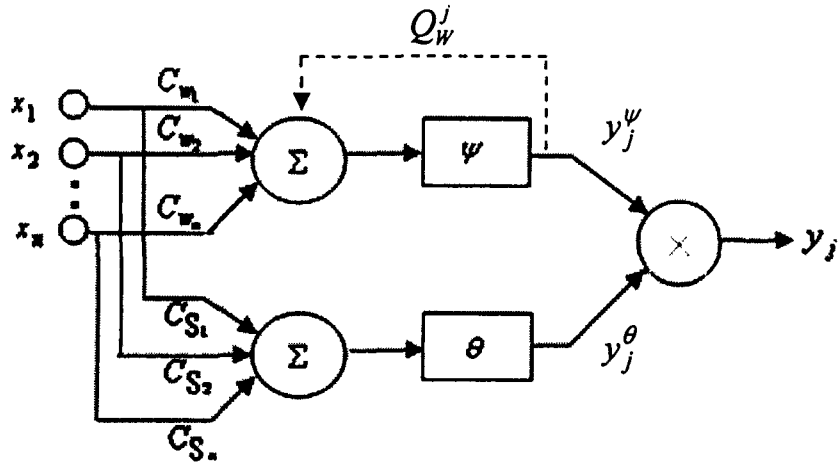


Fig. 6.2. Multiplication Sigmoid-Recurrent Wavelet (MS-RW) neuron model

### 6-2.2 Recurrent Sigmoid-Wavelet (RS-W) Neuron

In this neuron model, the output of SAF is feedback to itself with feedback weight  $Q_S$ . Equation (6.2) represents the dynamics of SAF while (2.22) represents the output of WAF for RS-W neuron models. Summation RS-W (SRS-W) and Multiplication RS-W (MRS-W) neuron models, are shown in Fig. 6.3 and Fig. 6.4, respectively.

$$y_j^\theta(k) = \theta \left( Q_S^j \cdot y_j^\theta(k-1) + \sum_{i=1}^n C_{S_i}^j \cdot x_i(k) \right) \quad (6.2)$$

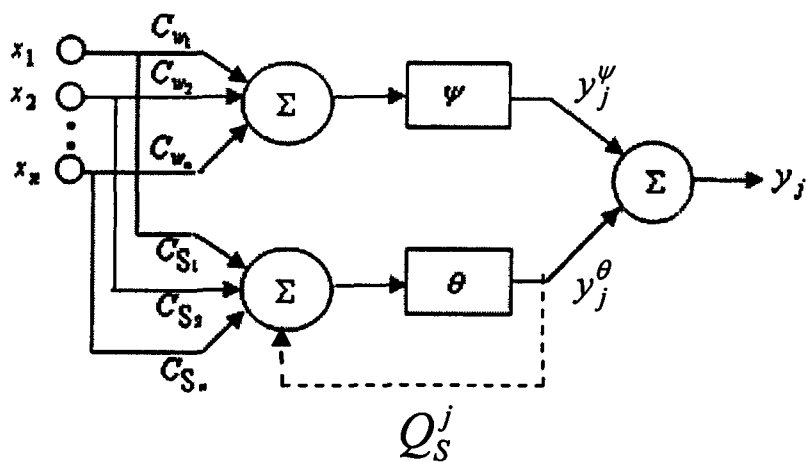


Fig. 6.3. Summation Recurrent Sigmoid-Wavelet (SRS-W) neuron model

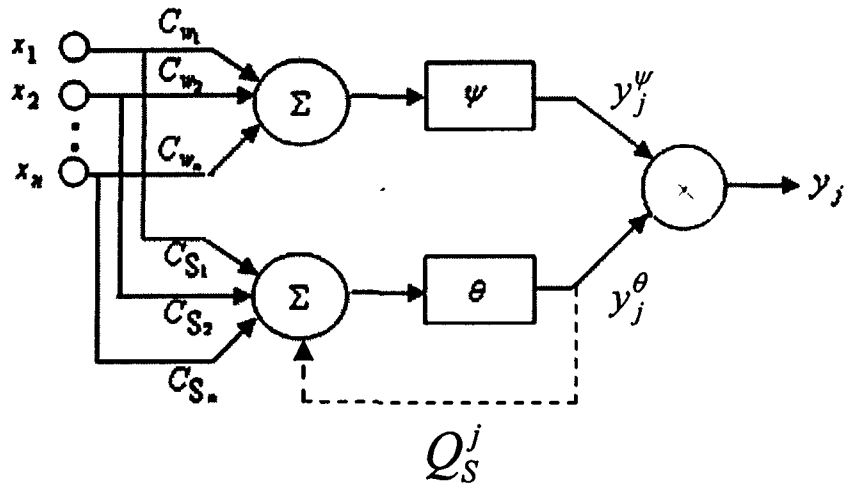


Fig. 6.4. Multiplication Recurrent Sigmoid-Wavelet (MRS-W) neuron model

### 6-2.3 Feedback to Sigmoid from Wavelet (FS-W) Neuron

In Feedback to Sigmoid from Wavelet (FS-W) neuron model, the weighted single delayed output of the WAF is feedback as the input to the SAF.  $Q_{ws}^j$  is the weight to the delayed output of WAF when feedback as the input to the SAF. The output of the SAF is given in (6.3) while the outputs of the WAF as (2.22). Figure 6.5 and Fig. 6.6 show the structure of Summation FS-W (SFS-W) and Multiplication FS-W (MFS-W) neuron models, respectively.

$$y_j^\theta(k) = \theta \left( Q_{ws}^j \cdot y_j^\psi(k-1) + \sum_{i=1}^n C_{S_i}^j \cdot x_i(k) \right) \quad (6.3)$$

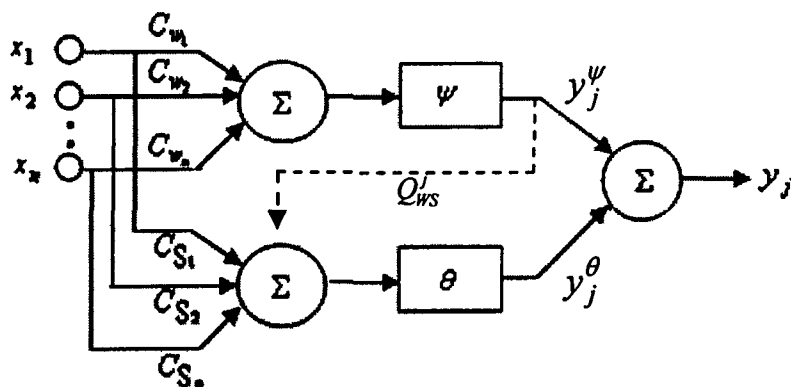


Fig. 6.5. Summation Feedback to Sigmoid from Wavelet (SFS-W) neuron model

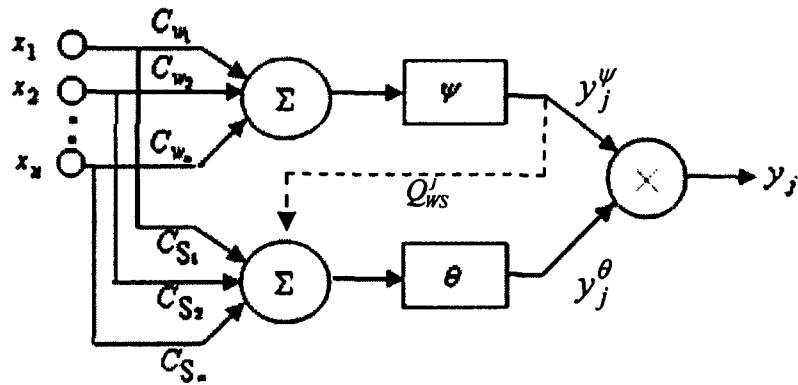


Fig. 6.6. Multiplication Feedback to Sigmoid from Wavelet (MFS-W) neuron model

### 6-2.4 Feedback to Wavelet from Sigmoid (FW-S) Neuron

In Feedback to Wavelet from Sigmoid (FW-S) neuron model, the weighted single delayed output of the SAF is feedback as the input to the WAF.  $Q_{SW}^j$  is the weight to the delayed output of SAF when feedback as the input to the WAF. The output of the WAF is given in (6.4) while the output of the SAF is the same as (2.21). Figure 6.7 and Fig. 6.8 show the structure of Summation FW-S (SFW-S) and Multiplication FW-S (MFW-S) neuron models, respectively.

$$y_j^\psi(k) = \psi \left( Q_{SW}^j \cdot y_j^\theta(k-1) + \sum_{i=1}^n C_{w_i}^j \cdot x_i(k) \right) \quad (6.4)$$

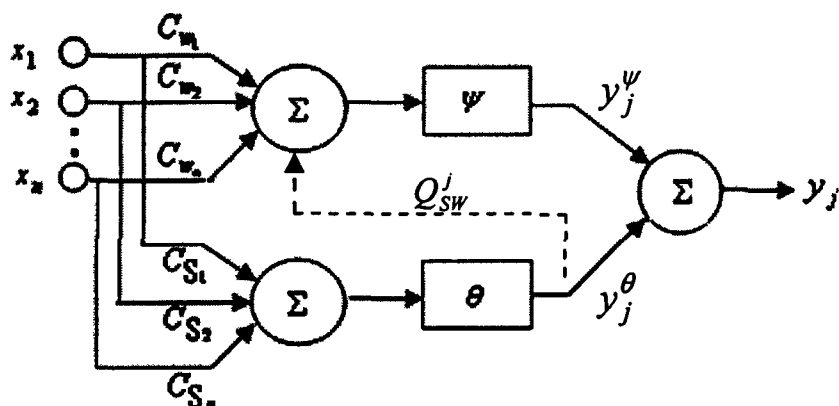


Fig. 6.7. Summation Feedback to Wavelet from Sigmoid (SFW-S) neuron model

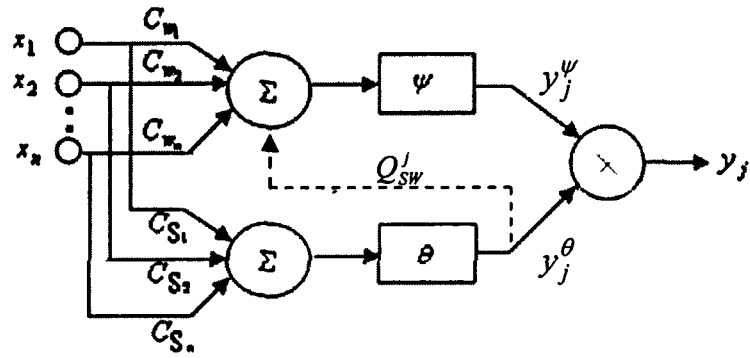


Fig. 6.8. Multiplication Feedback to Wavelet from Sigmoid (MFW-S) neuron model

### 6-2.5 Recurrent Neuron (RN)

Figure 6.9 and Fig. 6.10 show the architecture of the Summation RN (SRN) and Multiplication RN (MRN), respectively. In these neuron models, the weighted delayed output of hidden neuron is feedback itself. It results the feedback of the neuron's delayed output to both SAF and WAF with weight  $Q_j$ . The output of the WAF,  $y_j^\psi$ , and the SAF,  $y_j^\theta$ , are given in equations (6.5-6.6).

$$y_j^\psi(k) = \psi \left( Q_j' \cdot y_j(k-1) + \sum_{i=1}^n C_{w_i}' \cdot x_i(k) \right) \quad (6.5)$$

$$y_j^\theta(k) = \theta \left( Q_j' \cdot y_j(k-1) + \sum_{i=1}^n C_{s_i}' \cdot x_i(k) \right) \quad (6.6)$$

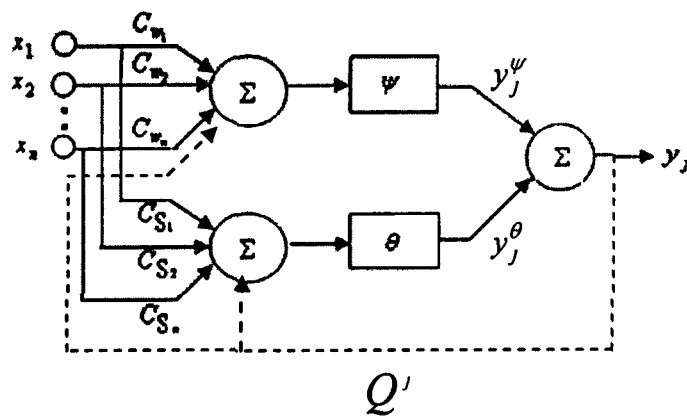


Fig. 6.9. Summation Recurrent Neuron (SRN)

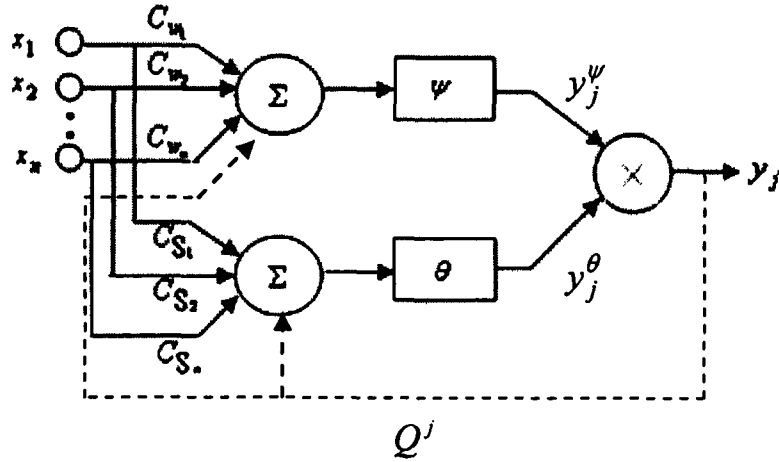


Fig. 6.10. Multiplication Recurrent Neuron (MRN)

### 6-3 Universal approximation of the proposed recurrent neuron models

By applying Stone-Weierstrass Theorem, the proposed SS-W & MS-W neuron models can be shown to be a universal approximate for continuous functions over compact set if it satisfies a certain condition. Then we have the following theorems, Theorems 6.1 to 6.5, for different recurrent network. Approve of the Theorems is presented in Appendix A.

Theorem 6.1: *Universal approximation theorem of recurrent SS-W neuron models*, for any real

function  $h: \mathcal{R}^n \rightarrow \mathcal{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathcal{R}^n$  and for any given  $\varepsilon > 0$  there is an recurrent SS-W network  $f$ , for all recurrent SS-W neuron models, such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

Theorem 6.2: *Universal approximation theorem of MS-RW neuron model*, for any real function

$h: \mathcal{R}^n \rightarrow \mathcal{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathcal{R}^n$  and for any given

$\varepsilon > 0$ , there is an MS-RW neuron model  $f$  that satisfies condition (6.7), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

$$C_W X + Q_W y^\psi (k-1) \neq b + a(2\rho+1)\frac{\pi}{2} \quad (6.7)$$

Where  $C_W = \{C_{W_1}, C_{W_2}, \dots, C_{W_n}\}$ ,  $Q_W = \{Q_{W_1}, Q_{W_2}, \dots, Q_{W_L}\}$ ,  $y^\psi = \{y_1^\psi, y_2^\psi, \dots, y_L^\psi\}$ ,  $X = \{x_1, x_2, \dots, x_n\}$  and  $\rho$  is any integer value.

**Theorem 6.3:** *Universal approximation theorem of MRS-W neuron model*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an MRS-W neuron model  $f$  that satisfies condition (2.28), such as  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

**Theorem 6.4:** *Universal approximation theorem of MFS-W neuron model*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an MFS-W neuron model  $f$  that satisfies condition (2.28), such as  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

**Theorem 6.5:** *Universal approximation theorem of MFW-S neuron model*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an MFW-S neuron model  $f$  that satisfies condition (6.8), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

$$C_W X + Q_{SW} y^\psi (k-1) \neq b + a(2\rho+1)\frac{\pi}{2} \quad (6.8)$$

where  $C_W = \{C_{W_1}, C_{W_2}, \dots, C_{W_n}\}$ ,  $Q_{SW} = \{Q_{SW_1}, Q_{SW_2}, \dots, Q_{SW_L}\}$ ,  $y^\psi = \{y_1^\psi, y_2^\psi, \dots, y_L^\psi\}$ ,  $X = \{x_1, x_2, \dots, x_n\}$  and  $\rho$  is any integer value. Then we have the following result.



Theorem 6.6: *Universal approximation theorem of MRN neuron model*, for any real function

$h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given

$\varepsilon > 0$  there is an RN neuron model  $f$  that satisfies condition (6.9), such that

$\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

$$C_w X + Q y(k-1) \neq b + a(2\rho + 1) \frac{\pi}{2} \quad (6.9)$$

Where  $C_w = \{C_{w_1}, C_{w_2}, \dots, C_{w_n}\}$ ,  $Q = \{Q_1, Q_2, \dots, Q_L\}$ ,  $y = \{y_1, y_2, \dots, y_L\}$ ,

$X = \{x_1, x_2, \dots, x_n\}$  and  $\rho$  is any integer value. Then we have the following result.

## 6-4 Gradient Descent learning of parameters

We apply gradient descent technique as discussed in [chapter 2](#) to modify the parameters

$W$ ,  $C_w$ ,  $C_s$  and delay elements  $Q_w$ ,  $Q_s$  and  $Q$  in different recurrent neuron models. The

parameter update formula for  $p^{\text{th}}$  data set is discussed in (2.18). This equation for the parameters

$W$ ,  $C_w$ ,  $C_s$  in different recurrent neuron model is as follows:

$$\Delta_p W_j(q) = -\eta \frac{\partial J}{\partial W_j} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \frac{\partial \hat{y}}{\partial W_j} \Big|_p \quad (6.10)$$

$$\Delta_p C_{w_j}(q) = -\eta \frac{\partial J}{\partial C_{w_j}} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \frac{\partial \hat{y}}{\partial C_{w_j}} \Big|_p \quad (6.11)$$

$$\Delta_p C_{s_j}(q) = -\eta \frac{\partial J}{\partial C_{s_j}} = \eta \cdot \frac{1}{P \cdot y_r^2} \cdot e \cdot \frac{\partial \hat{y}}{\partial C_{s_j}} \Big|_p \quad (6.12)$$

$\frac{\partial \hat{y}}{\partial W_j}$  in (6.10) for all recurrent neuron models is calculated by (6.13) and (6.14) for SS-W and

MSW, respectively.

$$\frac{\partial \hat{y}(k)}{\partial W_j} = y_j(k) = y_j^o(k) + y_j^\psi(k) \quad (6.13)$$

$$\frac{\partial \hat{y}}{\partial W_j} = y_j(k) = y_j^o(k) \cdot y_j^\psi(k) \quad (6.14)$$

$y_j^o(k)$  and  $y_j^\psi(k)$  change depend on the different recurrent models as discussed in section 6.2.

$\frac{\partial \hat{y}}{\partial W_j}$ ,  $\frac{\partial \hat{y}}{\partial W_j}$  and delay elements  $Q_w$ ,  $Q_s$  and  $Q$  in different recurrent neuron models are

calculated as follow:

#### 6-4.1 Sigmoid-Recurrent Wavelet (S-RW) Neuron

In S-RW network, feedback is only in WAF then (6.15) and (6.16) is applied to evaluate

$\frac{\partial \hat{y}(k)}{\partial C_{S_i}^j}$  in SS-RW and MS-RW neuron models, respectively.

$$\frac{\partial \hat{y}(k)}{\partial C_{S_i}^j} = x_i(k) \cdot W_j \cdot \theta' \left( \sum_{i=1}^n C_{S_i}^j \cdot x_i(k) \right) \quad (6.15)$$

$$\frac{\partial \hat{y}(k)}{\partial C_{S_i}^j(k)} = x_i(k) \cdot W_j(k) \cdot y_j^\psi(k) \cdot \theta' \left( \sum_{i=1}^n C_{S_i}^j(k) \cdot x_i(k) \right) \quad (6.16)$$

By applying chain rule, the following equations are obtained to learn parameters  $C_w$  and

$Q_w$  in recurrent (a) SS-RW and (b) MS-RW neuron models:

**a) S-RW with Summation (SS-RW)**

$$\frac{\partial \hat{y}(k)}{\partial C_{w_i}^j} = x_i(k) \cdot W^j \cdot \psi' \left( \sum_{i=1}^n C_{w_i}^j \cdot x_i(k) + Q_w^j \cdot y_w^j(k-1) \right) \quad (6.17)$$

$$\frac{\partial \hat{y}(k)}{\partial Q_w^j} = W^j \cdot y_w^j(k-1) \cdot \psi' \left( \sum_{i=1}^n C_{w_i}^j \cdot x_i(k) + Q_w^j \cdot y_w^j(k-1) \right) \quad (6.18)$$

**b) S-RW with Multiplication (MS-RW)**

$$\frac{\partial \hat{y}(k)}{\partial C_{w_i}^j(k)} = x_i(k) \cdot W^j(k) \cdot y_j^o(k) \cdot \psi' \left( \sum_{i=1}^n C_{w_i}^j(k) \cdot x_i(k) + Q_w^j(k) \cdot y_j^o(k-1) \right) \quad (6.19)$$

$$\frac{\partial \hat{y}(k)}{\partial Q_w^j(k)} = W^j(k) \cdot y_j^o(k-1) \cdot \psi' \left( \sum_{i=1}^n C_{w_i}^j(k) \cdot x_i(k) + Q_w^j(k) \cdot y_j^o(k-1) \right) \quad (6.20)$$

### 6-4.2 Recurrent Sigmoid–Wavelet (RS-W) Neuron

In this network, the output of SAF is feedback to itself with feedback weight  $Q_S$ .

Therefore (6.21) and (6.22), can be applied to evaluate  $\frac{\partial \hat{y}(k)}{\partial C_{w_i}^j}$  in SRS-W and MRS-W,

respectively.

$$\frac{\partial \hat{y}(k)}{\partial C_{w_i}^j} = x_i(k) \cdot W^j \cdot \psi' \left( \sum_{i=1}^n C_{w_i}^j \cdot x_i(k) \right) \quad (6.21)$$

$$\frac{\partial \hat{y}(k)}{\partial C_{w_i}^j(k)} = x_i(k) \cdot W_j(k) \cdot y_j^o(k) \cdot \psi' \left( \sum_{i=1}^n C_{w_i}^j(k) \cdot x_i(k) \right) \quad (6.22)$$

Learning for  $C_S$  and  $Q_{ws}$  parameters can be achieved by applying chain rule as follows

for recurrent (a) SRS-W and (b) MRS-W neuron models.

**a) RS-W with Summation (SRS-W)**

$$\frac{\partial \hat{y}(k)}{\partial C'_{S_i}} = x_i(k) \cdot W^j \cdot \theta' \left( \sum_{i=1}^n C'_{S_i} \cdot x_i(k) + Q'_{WS} \cdot y'_\psi(k-1) \right) \quad (6.23)$$

$$\frac{\partial \hat{y}(k)}{\partial Q'_{WS}} = W^j \cdot y'_\psi(k-1) \cdot \theta' \left( \sum_{i=1}^n C'_{S_i} \cdot x_i(k) + Q'_{WS} \cdot y'_\psi(k-1) \right) \quad (6.24)$$

**b) RS-W with Multiplication (MRS-W)**

$$\frac{\partial \hat{y}(k)}{\partial C'_{S_i}(k)} = x_i(k) \cdot W^j(k) \cdot y'_\psi(k) \cdot \theta' \left( \sum_{i=1}^n C'_{S_i}(k) \cdot x_i(k) + Q'_S(k) \cdot y'_\psi(k-1) \right) \quad (6.25)$$

$$\frac{\partial \hat{y}(k)}{\partial Q'_S(k)} = W^j(k) \cdot y'_\psi(k-1) \cdot y'_\psi(k) \cdot \theta' \left( \sum_{i=1}^n C'_{S_i}(k) \cdot x_i(k) + Q'_S(k) \cdot y'_\psi(k-1) \right) \quad (6.26)$$

### 6-4.3 Feedback to Sigmoid from Wavelet (FS-W) Neuron

Since there is not any feedback to WAF the updated equation for parameter  $C_{W_j}$  is achieved by (6.21) and (6.22) for SFS-W and MFS-W neuron models, respectively. Update equation for  $C_S$  and  $Q_{WS}$  is achieved by applying chain rule method as follows for recurrent (a) SFS-W and (b) MFS-W neuron models:

**a) FS-W with Summation (SFS-W)**

$$\frac{\partial \hat{y}(k)}{\partial C'_{S_i}} = x_i(k) \cdot W^j \cdot \theta' \left( \sum_{i=1}^n C'_{S_i} \cdot x_i(k) + Q'_{WS} \cdot y'_\psi(k-1) \right) \quad (6.27)$$

$$\frac{\partial \hat{y}(k)}{\partial Q'_{WS}} = W^j \cdot y'_\psi(k-1) \cdot \theta' \left( \sum_{i=1}^n C'_{S_i} \cdot x_i(k) + Q'_{WS} \cdot y'_\psi(k-1) \right) \quad (6.28)$$

**b) FS-W with Multiplication (MFS-W)**

$$\frac{\partial \hat{y}(k)}{\partial C'_s(k)} = x_i(k) \cdot W'(k) \cdot y_j^w(k) \cdot \theta' \left( \sum_{i=1}^n C'_s(k) \cdot x_i(k) + Q'_{ws}(k) \cdot y_j^w(k-1) \right) \quad (6.29)$$

$$\frac{\partial \hat{y}(k)}{\partial Q'_{ws}(k)} = W'(k) \cdot y_j^w(k-1) \cdot y_j^w(k) \cdot \theta' \left( \sum_{i=1}^n C'_s(k) \cdot x_i(k) + Q'_{ws}(k) \cdot y_j^w(k-1) \right) \quad (6.30)$$

**6-4.4 Feedback to Wavelet from Sigmoid (FW-S) Neuron**

Equation (6.15) and (6.16) are applied to update learning parameter  $C_s$  in SS-RW and MS-RW neuron models, respectively. Applying the chain rule method results following equations for  $C_w$  and  $Q_{sw}$  parameters of recurrent (a) SFW-S and (b) MFW-S neuron models:

**a) FW-S with Summation (SFW-S)**

$$\frac{\partial \hat{y}(k)}{\partial C'_{w_i}} = x_i(k) \cdot W' \cdot \psi' \left( \sum_{i=1}^n C'_{w_i} \cdot x_i(k) + Q'_{sw} \cdot y'_\theta(k-1) \right) \quad (6.31)$$

$$\frac{\partial \hat{y}(k)}{\partial Q'_{sw}} = W' \cdot y'_\theta(k-1) \cdot \psi' \left( \sum_{i=1}^n C'_{w_i} \cdot x_i(k) + Q'_{sw} \cdot y'_\theta(k-1) \right) \quad (6.32)$$

**b) FW-S with Multiplication (MFW-S)**

$$\frac{\partial \hat{y}(k)}{\partial C'_{w_i}(k)} = x_i(k) \cdot W'(k) \cdot y_j^o(k) \cdot \psi' \left( \sum_{i=1}^n C'_{w_i}(k) \cdot x_i(k) + Q'_{sw}(k) \cdot y_j^o(k-1) \right) \quad (6.33)$$

$$\frac{\partial \hat{y}(k)}{\partial Q'_{sw}(k)} = W'(k) \cdot y_j^o(k-1) \cdot y_j^o(k) \cdot \psi' \left( \sum_{i=1}^n C'_{w_i}(k) \cdot x_i(k) + Q'_{sw}(k) \cdot y_j^o(k-1) \right) \quad (6.34)$$

**6-4.5 Recurrent Neuron (RN)**

The following equations are drawn by applying chain rule method for updating the parameters  $C_w$ ,  $C_s$  and  $Q_{sw}$  for recurrent (a) SRN and (b) MRN neuron models.

**a) RN with Summation (SRN)**

$$\frac{\partial \hat{y}(k)}{\partial C'_{w_i}} = x_i(k) \cdot W' \cdot \psi' \left( \sum_{i=1}^n C'_{w_i} \cdot x_i(k) + Q' \cdot y'(k-1) \right) \quad (6.35)$$

$$\frac{\partial \hat{y}(k)}{\partial C'_{s_i}} = x_i(k) \cdot W' \cdot \theta' \left( \sum_{i=1}^n C'_{s_i} \cdot x_i(k) + Q' \cdot y'(k-1) \right) \quad (6.36)$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial Q'} = W' \cdot y'(k-1) \cdot & \left[ \psi' \left( \sum_{i=1}^n C'_{w_i} \cdot x_i(k) + Q' \cdot y'(k-1) \right) + \right. \\ & \left. \theta' \left( \sum_{i=1}^n C'_{s_i} \cdot x_i(k) + Q' \cdot y'(k-1) \right) \right] \end{aligned} \quad (6.37)$$

**b) RN with Multiplication (MRN)**

$$\frac{\partial \hat{y}(k)}{\partial C'_{w_i}(k)} = x_i(k) \cdot W'(k) \cdot y_j^\theta(k) \cdot \psi' \left( \sum_{i=1}^n C'_{w_i}(k) \cdot x_i(k) + Q'(k) \cdot y_j(k-1) \right) \quad (6.38)$$

$$\frac{\partial \hat{y}(k)}{\partial C'_{s_i}(k)} = x_i(k) \cdot W'(k) \cdot y_j^\psi(k) \cdot \theta' \left( \sum_{i=1}^n C'_{s_i}(k) \cdot x_i(k) + Q'(k) \cdot y_j(k-1) \right) \quad (6.39)$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial Q'(k)} = W'(k) \cdot y_j(k-1) \cdot & \left[ y_j^\theta(k) \cdot \psi' \left( \sum_{i=1}^n C'_{w_i}(k) \cdot x_i(k) + Q'(k) \cdot y_j(k-1) \right) + \right. \\ & \left. y_j^\psi(k) \cdot \theta' \left( \sum_{i=1}^n C'_{s_i}(k) \cdot x_i(k) + Q'(k) \cdot y_j(k-1) \right) \right] \end{aligned} \quad (6.40)$$

### 6-4.6 Stability analysis of the recurrent neuron models

A small value of learning rate  $\eta$  leads to the lower speed of convergence, while a large value causes the learning procedure unstable. Therefore learning parameter is selected large enough so the convergence speed and stability should be guaranteed. To guarantee stability during the learning procedure, we have applied the Lyapunov stability theorem. The speed of convergence is guaranteed with selecting adaptive learning rate with the lower and upper bound

as mentioned in the stability Theorem (2.1). Following theorems guarantees the convergence stability of the recurrent neuron models. The proof of the all theorems is derived in Appendix C.

**Theorem 6.7:** The convergence and stability of the learning procedure, for recurrent SS-RW and MS-RW neuron models, guaranteed if the upper bound of the learning parameters  $\eta_w$ ,  $\eta_{c_w}$ ,  $\eta_{c_s}$  and  $\eta_{Q_w}$  for recurrent (a) SS-RW and (b) MS-RW neuron models are selected as follows:

a) For recurrent SS-RW neuron model

$$0 < \eta_w < \frac{P \cdot y_r^2}{2} \quad (6.42)$$

$$0 < \eta_{c_w} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.42)$$

$$0 < \eta_{c_s} < 2 \cdot P \cdot y_r^2 \quad (6.43)$$

$$0 < \eta_{Q_w} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.44)$$

b) For recurrent MS-RW neuron model

$$0 < \eta_w < 2 \cdot P \cdot y_r^2 \quad (6.45)$$

$$0 < \eta_{c_w} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.46)$$

$$0 < \eta_{c_s} < 2 \cdot P \cdot y_r^2 \quad (6.47)$$

$$0 < \eta_{Q_w} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.48)$$

*Lemma 6.1:* The range of learning parameter  $\eta_w$  is the same as (6.42) and (6.45) for all summation and multiplication recurrent neuron models, respectively.

*Theorem 6.8:* The convergence and stability of the learning procedure, for recurrent SRS-W and MRS-W neuron models, guaranteed if the upper bound of the learning parameters  $\eta_{C_w}$ ,  $\eta_{C_s}$  and  $\eta_{Q_s}$  for recurrent (a) SRS-W and (b) MRS-W neuron models are selected as follows:

a) For recurrent SRS-W neuron model

$$0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.49)$$

$$0 < \eta_{C_s} < 2 \cdot P \cdot y_r^2 \quad (6.50)$$

$$0 < \eta_{Q_s} < 2 \cdot P \cdot y_r^2 \quad (6.51)$$

b) For recurrent MRS-W neuron model

$$0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.52)$$

$$0 < \eta_{C_s} < 2 \cdot P \cdot y_r^2 \quad (6.53)$$

$$0 < \eta_{Q_s} < 2 \cdot P \cdot y_r^2 \quad (6.54)$$

*Theorem 6.9:* The convergence and stability of the learning procedure, for recurrent SFS-W and MFS-W neuron models, guaranteed if the upper bound of the learning parameters



$\eta_{C_w}$ ,  $\eta_{C_s}$  and  $\eta_{Q_{ws}}$  for recurrent (a) SFS-W and (b) MFS-W neuron models are selected as follows:

a) For recurrent SFS-W neuron model

$$0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.55)$$

$$0 < \eta_{C_s} < 2 \cdot P \cdot y_r^2 \quad (6.56)$$

$$0 < \eta_{Q_{ws}} < 2 \cdot P \cdot y_r^2 \quad (6.57)$$

b) For recurrent MFS-W neuron model

$$0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.58)$$

$$0 < \eta_{C_s} < 2 \cdot P \cdot y_r^2 \quad (6.59)$$

$$0 < \eta_{Q_{ws}} < 2 \cdot P \cdot y_r^2 \quad (6.60)$$

**Theorem 6.10:** The convergence and stability of the learning procedure, for recurrent SFW-S and MFW-S neuron models, guaranteed if the upper bound of the learning parameters  $\eta_{C_w}$ ,  $\eta_{C_s}$  and  $\eta_{Q_{sw}}$  for recurrent (a) SFW-S and (b) MFW-S neuron models are selected as follows:

a) For recurrent SFW-S neuron model

$$0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.61)$$

$$0 < \eta_{C_s} < 2 \cdot P \cdot y_r^2 \quad (6.62)$$

$$0 < \eta_{Q_{SW}} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.63)$$

b) For recurrent MFW-S neuron model

$$0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.64)$$

$$0 < \eta_{C_s} < 2 \cdot P \cdot y_r^2 \quad (6.65)$$

$$0 < \eta_{Q_{SW}} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.66)$$

*Theorem 6.11:* The convergence and stability of the learning procedure, for recurrent SRN and MRN neuron models, guaranteed if the upper bound of the learning parameters  $\eta_{C_w}$ ,  $\eta_{C_s}$  and  $\eta_Q$  for recurrent (a) SRN and (b) MRN neuron models are selected as follows:

a) For recurrent SRN neuron model

$$0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.67)$$

$$0 < \eta_{C_s} < 2 \cdot P \cdot y_r^2 \quad (6.68)$$

$$0 < \eta_Q < \frac{P \cdot y_r^2}{18} \quad (6.69)$$

b) For recurrent MRN neuron model

$$0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{49} \quad (6.70)$$

$$0 < \eta_{C_s} < 2 \cdot P \cdot y_r^2 \quad (6.71)$$

$$0 < \eta_Q < \frac{2 \cdot P \cdot y_r^2}{36} \quad (6.72)$$

## 6-5 Simulation Results

The structure of the proposed recurrent S-W neuron models is determined as discussed in chapter 2. Since this chapter presents a comparative study of different types of recurrent networks, initialization of all the networks should be the same to compare the results and to suggest an acceptable network. By keeping the weight  $Q$  in the recurrent networks equal to zero all recurrent networks will reduce to S-W network. Hence, with a given number of hidden neuron unit, it is possible to initialize all the networks with the same initial weights value excepts the weights  $Q$ . For a given number of hidden layer neuron unit, we initialize the weighted  $C_w$ ,  $C_s$  and  $W$  for recurrent SS-W and MS-W neuron models, the same as chapter 3 and the same initial weights are used for all recurrent network. In all recurrent S-W neuron models, Morlet activation function that yield better performance in chapter 2 is exploited.

### Revisited Example 1: *Linear regression with nonlinear input*

Figure 6.11 shows the performance index for SS-W neuron model and proposed recurrent neuron models. In Fig. 13 the learning pattern for the SS-W model is shown by solid blue line, for SS-RW model it is shown by solid red line, for SRS-W model it is shown by dashed black line, for SFS-W model it is shown by dotted green line, for SFW-S model it is shown by dashed-

dotted magenta line and for SRN model it is shown by solid cyan line. In recurrent SS-W neuron models, as shown in Fig. 13, S-RW model yield better result with  $J=1.185 \times 10^{-6}$ . Next to S-RW, SRN and SFW-S have better performance. It shows that recurrent models have much better performance as compared to the feed-forward SS-W model. Table 6.1 shows the performance index of different recurrent SS-W neuron models.

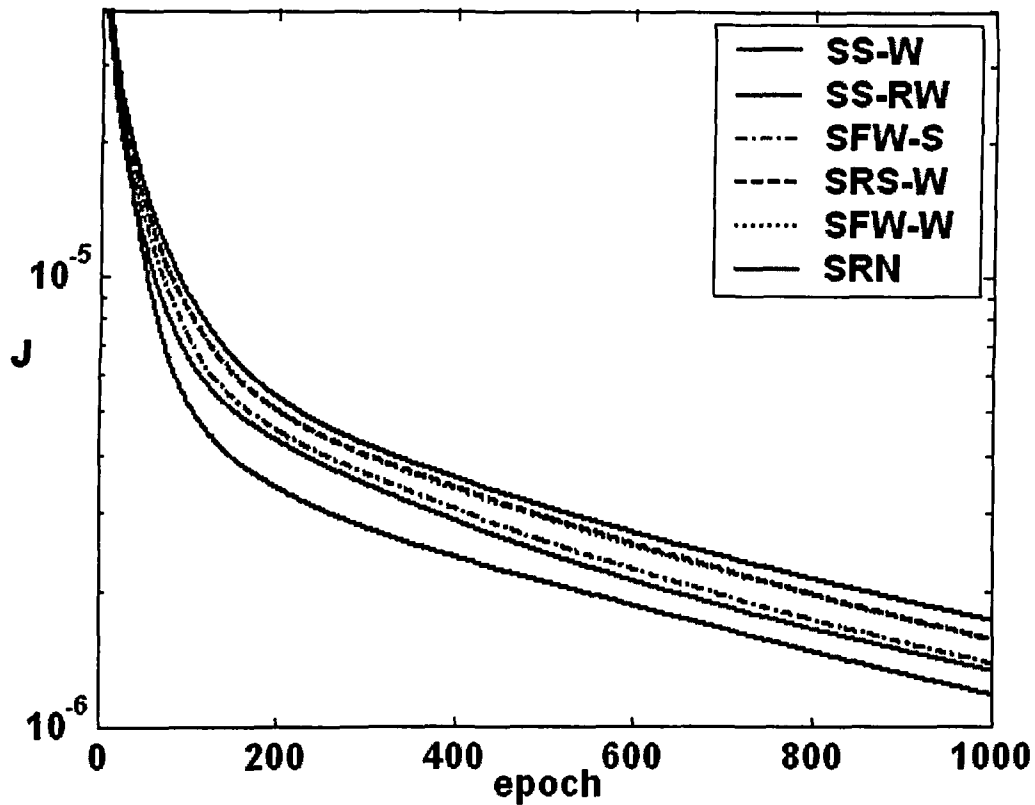


Fig. 6.11. Learning pattern of feed-forward network with recurrent SS-W neuron models for Example 1

The learning parameter  $W$ ,  $C_S$ ,  $C_W$  and delay elements  $Q_W$ ,  $Q_S$ ,  $Q_{SW}$ ,  $Q_{WS}$  and  $Q$  in SS-RW, SRS-W, SFW-S, SFS-W and SRN networks, respectively, are as follow. The columns in  $W$  and delay element and the rows in  $C_S$  and  $C_W$  are equal to the number of hidden neuron from the

conjunction of the SAF and WAF that is equal to  $a \cdot (a+1)/2$ . The numbers of columns in  $C_s$  and  $C_w$  are also equal to the number of inputs.

For recurrent SS-RW neuron model:

$$W^f = [0.7858 \quad -0.0602 \quad 1.2423] \quad Q_w^f = [-0.2738 \quad 0.0797 \quad -0.2738]$$

$$C_s^f = \begin{bmatrix} 0.1215 & 0.0466 & 0.92747 \\ 0.2044 & 0.7577 & 0.53309 \\ 0.1819 & 0.4949 & 0.33326 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.8857 & 0.2208 & 0.1571 \\ 0.0519 & 0.7589 & 0.9126 \\ 0.4286 & -0.0114 & 0.6155 \end{bmatrix}$$

For recurrent SRS-W neuron model:

$$W^f = [0.4955 \quad -0.0874 \quad 1.2831] \quad Q_s^f = [-0.0147 \quad -0.0057 \quad 0.0087]$$

$$C_s^f = \begin{bmatrix} 0.1324 & 0.0559 & 0.9233 \\ 0.2035 & 0.7569 & 0.5313 \\ 0.1977 & 0.5108 & 0.3530 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 1.1292 & 0.2876 & 0.0292 \\ 0.1041 & 0.8346 & 0.9817 \\ 0.5324 & -0.0762 & 0.5315 \end{bmatrix}$$

For recurrent SFW-S neuron model:

$$W^f = [0.5760 \quad -0.1290 \quad 1.2986] \quad Q_{sw}^f = [-0.1631 \quad 0.2359 \quad 0.0122]$$

$$C_s^f = \begin{bmatrix} 0.1314 & 0.0535 & 0.9278 \\ 0.2034 & 0.7570 & 0.5285 \\ 0.1972 & 0.5058 & 0.3531 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 1.0413 & 0.3853 & 0.1459 \\ 0.0936 & 0.8177 & 0.9555 \\ 0.5087 & -0.0964 & 0.5564 \end{bmatrix}$$

For recurrent SFS-W neuron model:

$$W^f = [0.5015 \quad -0.0867 \quad 1.2830] \quad Q_{ws}^f = [-0.0036 \quad 0.0098 \quad 0.0022]$$

$$C_s^f = \begin{bmatrix} 0.1320 & 0.0553 & 0.9227 \\ 0.2035 & 0.7567 & 0.5311 \\ 0.1974 & 0.5106 & 0.3525 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 1.1322 & 0.2847 & 0.0285 \\ 0.1060 & 0.8354 & 0.9824 \\ 0.5325 & -0.0754 & 0.5315 \end{bmatrix}$$

For recurrent SRN neuron model:

$$W^f = [0.5164 \quad -0.0571 \quad 1.2942] \quad Q^f = [-0.2471 \quad 0.0869 \quad 0.0023]$$

$$C_S^f = \begin{bmatrix} 0.1303 & 0.0567 & 0.9299 \\ 0.2033 & 0.7570 & 0.5324 \\ 0.1951 & 0.5126 & 0.3623 \end{bmatrix} \quad C_W^f = \begin{bmatrix} 1.1798 & 0.2485 & 0.0588 \\ 0.1047 & 0.8444 & 0.9911 \\ 0.5047 & -0.1111 & 0.5727 \end{bmatrix}$$

Figure 6.12 shows the performance index for MS-W neuron model and proposed recurrent networks. In Fig. 6.12, the learning pattern for the MS-W network it is shown by solid blue line, for MS-RW network it is shown by solid red line, for MRS-W network it is shown by dashed black line, for MFS-W network by dotted green line, for MFW-S network by dashed-dotted magenta line and for R-N network by solid cyan line. MFW-S network yield better result with  $J=4.834 \times 10^{-7}$ . Next to MFW-S, MRN and MS-RW have better performance. It shows that recurrent networks have much better performance as compared to the feed-forward MS-W network. Table 6.2 shows the performance index of different recurrent MS-W neuron models.

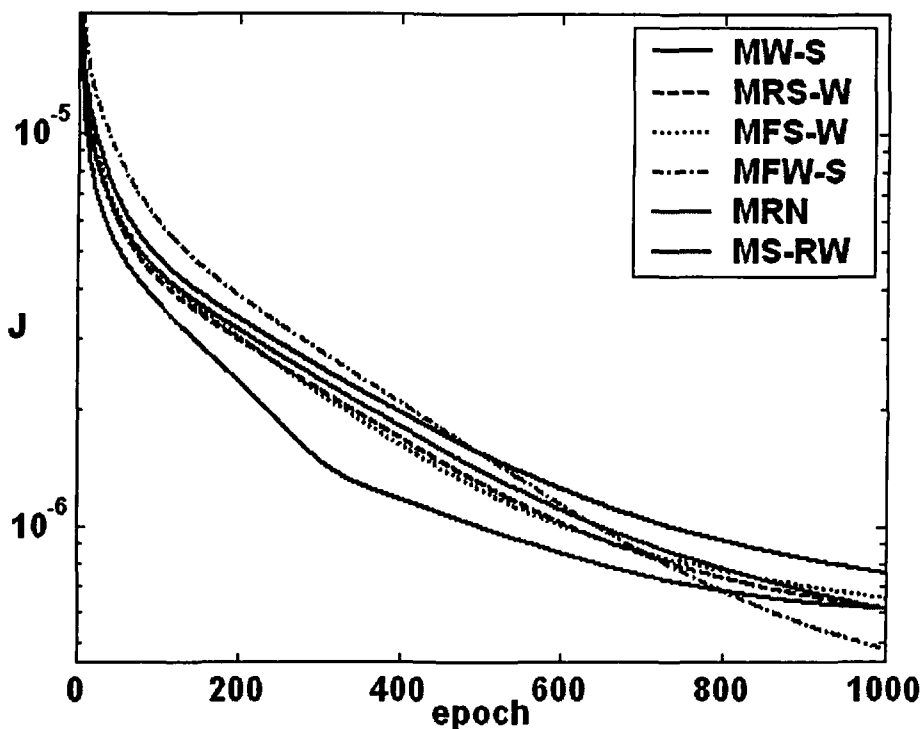


Fig. 6.12. Learning pattern of feed-forward network with recurrent MS-W neuron models for Example 1

The learning parameter  $W$ ,  $C_S$ ,  $C_W$  and delay elements  $Q_W$ ,  $Q_S$ ,  $Q_{SW}$ ,  $Q_{WS}$  and  $Q$  in MS-RW, MRS-W, MFW-S, MFS-W and MRN networks, respectively, are as follow. The columns in  $W$  and delay elements and the number of rows in  $C_S$  and  $C_W$  are equal to the number of the conjunction of the SAF and WAF that is  $a \cdot (a+1)/2$ . The numbers of columns in  $C_S$  and  $C_W$  are equal to the number of inputs.

For recurrent MS-RW neuron model:

$$W^f = \begin{bmatrix} 0.0503 & 0.5608 & 1.1975 \end{bmatrix} \quad Q_W^f = \begin{bmatrix} -0.3649 & -0.1212 & -0.1803 \end{bmatrix}$$

$$C_S^f = \begin{bmatrix} 0.8186 & 0.6630 & 0.3420 \\ 0.6343 & 0.3432 & 0.5925 \\ 0.8934 & 0.3332 & 0.8072 \end{bmatrix} \quad C_W^f = \begin{bmatrix} 0.7199 & 0.2217 & 0.5000 \\ -0.0960 & 0.4452 & -0.0279 \\ 0.3855 & -0.0782 & 0.6968 \end{bmatrix}$$

For recurrent MRS-W neuron model:

$$W^f = \begin{bmatrix} -0.0637 & 0.5714 & 1.1174 \end{bmatrix} \quad Q_S^f = \begin{bmatrix} 0.0168 & 0.0435 & 0.0819 \end{bmatrix}$$

$$C_S^f = \begin{bmatrix} 0.8188 & 0.6608 & 0.3416 \\ 0.6343 & 0.3400 & 0.5878 \\ 0.8847 & 0.3311 & 0.8069 \end{bmatrix} \quad C_W^f = \begin{bmatrix} 0.6078 & 0.3342 & 0.5475 \\ -0.1647 & 0.4310 & -0.0904 \\ 0.4010 & -0.1124 & 0.6301 \end{bmatrix}$$

For recurrent MFW-S neuron model:

$$W^f = \begin{bmatrix} -0.1116 & 0.5607 & 1.0017 \end{bmatrix} \quad Q_{SW}^f = \begin{bmatrix} 0.0235 & -0.8195 & -0.1304 \end{bmatrix}$$

$$C_S^f = \begin{bmatrix} 0.81963 & 0.6613 & 0.3426 \\ 0.6402 & 0.3532 & 0.6162 \\ 0.8849 & 0.2941 & 0.7741 \end{bmatrix} \quad C_W^f = \begin{bmatrix} 0.6121 & 0.2552 & 0.4587 \\ 0.1147 & 0.6284 & 0.2643 \\ 0.5286 & -0.0471 & 0.6943 \end{bmatrix}$$

For recurrent MFS-W neuron model:

$$W^f = \begin{bmatrix} -0.0651 & 0.5712 & 1.1237 \end{bmatrix} \quad Q_{WS}^f = \begin{bmatrix} 0.0190 & -0.0094 & 0.0378 \end{bmatrix}$$

$$C_s^f = \begin{bmatrix} 0.8188 & 0.6609 & 0.3417 \\ 0.6342 & 0.3405 & 0.5868 \\ 0.8866 & 0.3325 & 0.8088 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.6060 & 0.3335 & 0.5457 \\ -0.1651 & 0.4326 & -0.0878 \\ 0.4018 & -0.1115 & 0.6330 \end{bmatrix}$$

For recurrent MRN neuron model:

$$W^f = [-0.0317 \quad 0.5776 \quad 1.1715] \quad Q^f = [-0.1605 \quad -0.0991 \quad -0.0697]$$

$$C_s^f = \begin{bmatrix} 0.8189 & 0.6621 & 0.3423 \\ 0.6393 & 0.3528 & 0.5980 \\ 0.8909 & 0.3392 & 0.8152 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.6690 & 0.3016 & 0.5397 \\ -0.0829 & 0.4576 & -0.0602 \\ 0.3905 & -0.0781 & 0.6354 \end{bmatrix}$$

Figure 6.13 shows the output of system by solid line and output of the network with MFW-S neuron model by dotted line. The error between them is indicated by solid line.

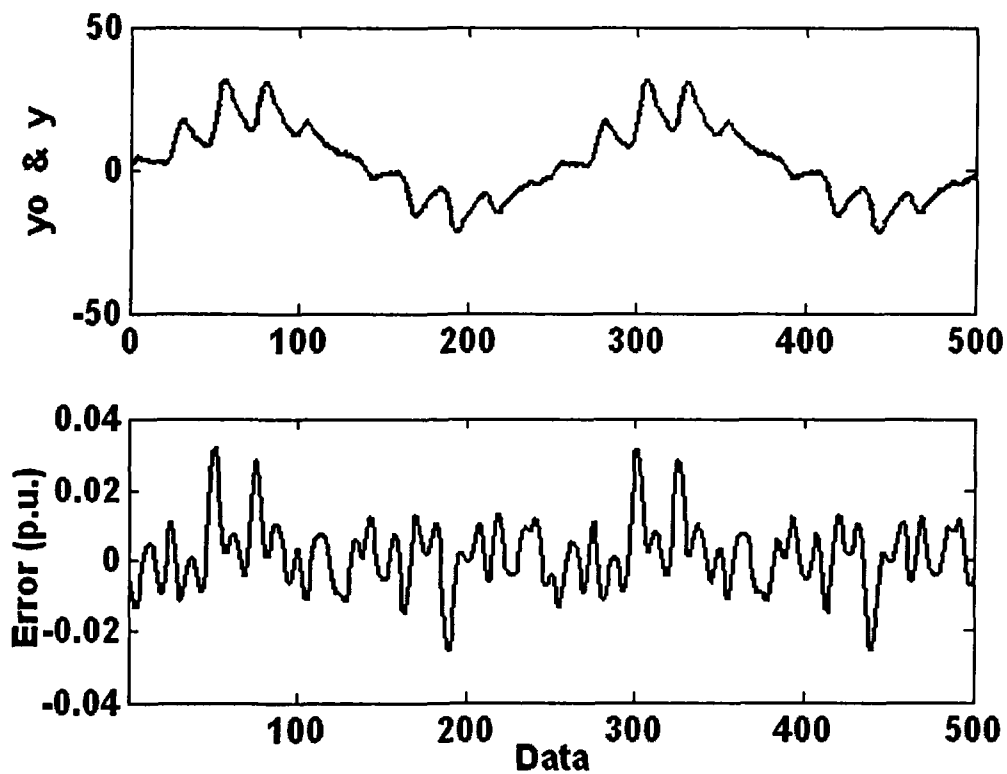


Fig. 6.13. Actual output and network output with MFW-S model and the error for Example 1



## Revisited Example 2: Non-linear regression with random input

In the recurrent SS-W networks, SS-RW model with performance index  $J=1.302 \times 10^{-5}$  yields better result than other recurrent model as shown in Fig. 6.14 and Table 6.1. Next to SS-RW, SR-N and SFW-S have better performance. The learning parameters of different recurrent networks are as follows.

For recurrent SS-RW neuron model:

$$W^f = [0.5566 \quad 0.4771 \quad -0.3027 \quad 1.0627 \quad 0.1647 \quad 0.4851]$$

$$Q_w^f = [-0.1070 \quad -0.3436 \quad -0.1295 \quad 0.1571 \quad 0.0095 \quad -0.1064]$$

$$C_s^f = \begin{bmatrix} 0.7077 & 0.1828 & 0.0759 \\ 0.6543 & 1.0056 & -0.0894 \\ 0.8019 & 0.2871 & 0.9134 \\ 0.9994 & 0.2598 & 0.0091 \\ 0.5487 & 0.8794 & 0.2656 \\ 0.9052 & 0.7431 & 0.5793 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.2598 & 0.0335 & 1.6424 \\ 1.4870 & 0.9909 & 0.8475 \\ 0.8356 & 0.4410 & 0.6125 \\ -0.2087 & 1.2168 & 1.3101 \\ 0.7640 & 0.7533 & 0.2526 \\ 0.9763 & 0.4307 & 0.3870 \end{bmatrix}$$

For recurrent SRS-W neuron model:

$$W^f = [0.5269 \quad 0.2537 \quad -0.3296 \quad 1.0858 \quad 0.2574 \quad 0.5696]$$

$$Q_s^f = [-0.0778 \quad -0.0993 \quad 0.0253 \quad -0.2476 \quad -0.0329 \quad -0.1260]$$

$$C_s^f = \begin{bmatrix} 0.7173 & 0.1765 & 0.0545 \\ 0.6632 & 0.9921 & -0.0881 \\ 0.7978 & 0.2849 & 0.9168 \\ 1.0241 & 0.2473 & -0.0394 \\ 0.5491 & 0.8775 & 0.2550 \\ 0.9138 & 0.7426 & 0.5484 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.4289 & -0.0238 & 1.5433 \\ 1.4794 & 0.8287 & 0.8646 \\ 0.7144 & 0.3212 & 0.6088 \\ -0.1423 & 1.0215 & 1.3574 \\ 0.7773 & 0.8884 & 0.1715 \\ 1.1249 & 0.4935 & 0.3710 \end{bmatrix}$$

For recurrent SFW-S neuron model:

$$W^f = [0.5341 \quad 0.3347 \quad -0.3750 \quad 1.1418 \quad 0.3878 \quad 0.5070]$$

$$Q_{sw}^f = [0.3643 \quad 0.2995 \quad 0.1790 \quad 0.2168 \quad -0.2220 \quad -0.1084]$$

$$C_s^f = \begin{bmatrix} 0.7140 & 0.1733 & 0.0660 \\ 0.6596 & 0.9850 & -0.0827 \\ 0.7942 & 0.2819 & 0.9175 \\ 1.0194 & 0.2320 & -0.0186 \\ 0.5496 & 0.8768 & 0.2416 \\ 0.9089 & 0.7335 & 0.5753 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.3739 & -0.1058 & 1.3028 \\ 1.4434 & 0.7800 & 0.6558 \\ 0.7152 & 0.2892 & 0.5277 \\ -0.1662 & 0.9524 & 1.1292 \\ 0.8440 & 0.9158 & 0.0623 \\ 0.8877 & 0.3338 & 0.3423 \end{bmatrix}$$

For recurrent SFS-W neuron model:

$$W^f = [0.5354 \quad 0.2526 \quad -0.3566 \quad 1.0641 \quad 0.2962 \quad 0.5808]$$

$$Q_{ws}^f = [0.01861 \quad -0.1029 \quad 0.0232 \quad -0.1035 \quad -0.0507 \quad -0.1212]$$

$$C_s^f = \begin{bmatrix} 0.7169 & 0.1790 & 0.0484 \\ 0.6646 & 0.9952 & -0.0867 \\ 0.7976 & 0.2839 & 0.9185 \\ 1.0209 & 0.2392 & -0.0351 \\ 0.5494 & 0.8794 & 0.2484 \\ 0.9146 & 0.7486 & 0.5414 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.4461 & -0.0187 & 1.5227 \\ 1.4699 & 0.8141 & 0.8647 \\ 0.7035 & 0.3190 & 0.6063 \\ -0.1440 & 1.0186 & 1.3652 \\ 0.7792 & 0.9216 & 0.1011 \\ 1.1136 & 0.4745 & 0.3424 \end{bmatrix}$$

For recurrent SRN neuron model:

$$W^f = [0.5671 \quad 0.1588 \quad -0.3664 \quad 1.1152 \quad 0.4033 \quad 0.5583]$$

$$Q^f = [0.1745 \quad -0.0796 \quad 0.1228 \quad 0.1530 \quad -0.2681 \quad -0.1929]$$

$$C_s^f = \begin{bmatrix} 0.7162 & 0.1718 & 0.0495 \\ 0.6573 & 0.9825 & -0.0758 \\ 0.7947 & 0.2834 & 0.9175 \\ 1.0172 & 0.2341 & -0.0250 \\ 0.5485 & 0.8764 & 0.2418 \\ 0.9117 & 0.7383 & 0.5503 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.5252 & -0.0829 & 1.3926 \\ 1.3068 & 0.6994 & 0.8555 \\ 0.7123 & 0.3367 & 0.5597 \\ -0.1217 & 1.0799 & 1.2495 \\ 0.8748 & 0.9394 & 0.0553 \\ 1.0153 & 0.4033 & 0.3887 \end{bmatrix}$$

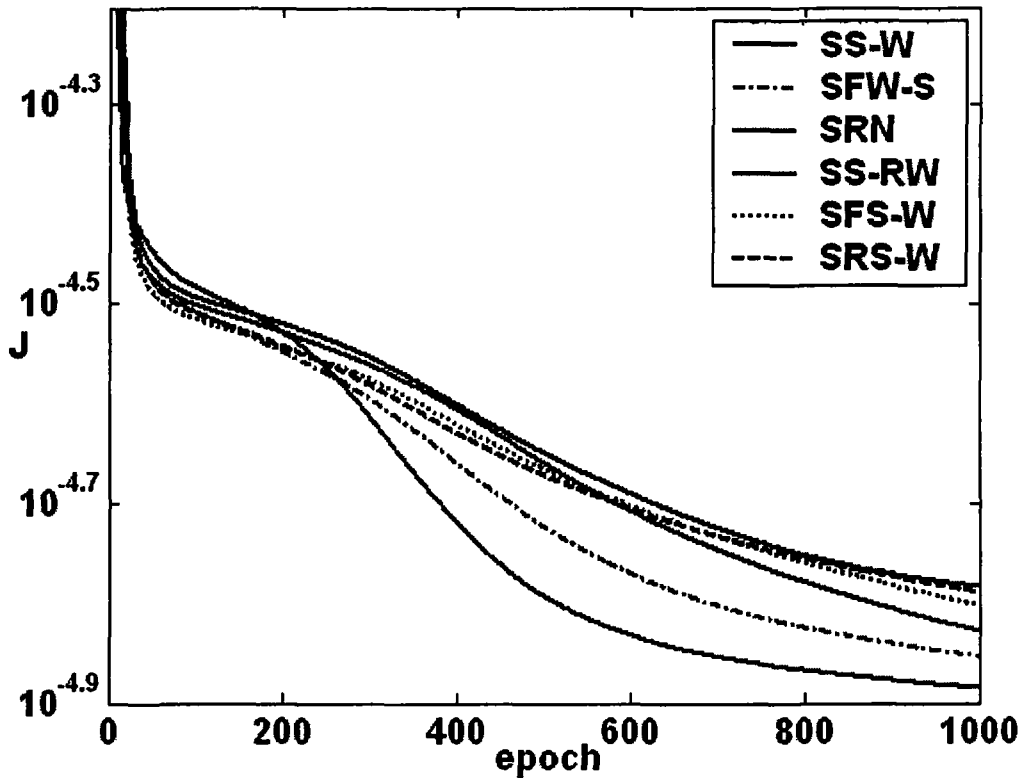


Fig. 6.14. Learning pattern of feed-forward network with recurrent SS-W neuron models for Example 2

As shown in Fig. 6.16 and Table 6.2, MFW-S neuron model with performance index  $J=1.034 \times 10^{-5}$  yields better result than other recurrent model. Next to MFW-S, MS-RW and MRN have better performance. Figure 6.16 shows the output of system and output of the network with MS-RW neuron model and the error between them. The learning parameters for recurrent MS-W neuron models are as follows.

For recurrent MS-RW neuron model:

$$W^f = [0.4311 \quad 0.3929 \quad 0.3828 \quad 1.0594 \quad 0.9677 \quad 0.3993]$$

$$Q_w^f = [0.0093 \quad 0.1076 \quad 0.0165 \quad -0.0962 \quad 0.2079 \quad 0.1346]$$

$$C_s^f = \begin{bmatrix} 0.6873 & 0.2138 & 0.1457 \\ 0.6161 & 0.9781 & 0.0473 \\ 0.8383 & 0.2516 & 0.9249 \\ 0.9856 & 0.1763 & 0.2347 \\ 0.5984 & 0.9089 & 0.3798 \\ 0.8748 & 0.7581 & 0.6400 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.6497 & 0.0340 & 1.2251 \\ 0.9668 & 0.7472 & 0.4461 \\ 0.7452 & 0.3453 & 0.2009 \\ -0.0041 & 0.8706 & 1.0682 \\ 0.4751 & 0.9872 & -0.2128 \\ 0.8913 & 0.4890 & 0.4386 \end{bmatrix}$$

For recurrent MRS-W neuron model:

$$W^f = [0.3902 \quad 0.5236 \quad 0.4225 \quad 1.1127 \quad 1.0827 \quad 0.2837]$$

$$Q_s^f = [0.0065 \quad 0.0604 \quad -0.0421 \quad 0.0502 \quad -0.0393 \quad -0.0120]$$

$$C_s^f = \begin{bmatrix} 0.6946 & 0.2026 & 0.1448 \\ 0.6292 & 0.9859 & 0.0477 \\ 0.8367 & 0.2651 & 0.9264 \\ 0.9834 & 0.1641 & 0.2508 \\ 0.6054 & 0.8865 & 0.4574 \\ 0.8718 & 0.7523 & 0.6385 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.6166 & 0.0096 & 1.2631 \\ 0.9379 & 0.7219 & 0.4026 \\ 0.7151 & 0.3136 & 0.3276 \\ -0.0121 & 1.0353 & 0.9956 \\ 0.5192 & 1.0117 & -0.0977 \\ 0.8942 & 0.4705 & 0.5315 \end{bmatrix}$$

For recurrent MFW-S neuron model:

$$W^f = [0.3398 \quad 0.5024 \quad 0.4007 \quad 1.0600 \quad 1.0523 \quad 0.2724]$$

$$Q_{sw}^f = [0.3212 \quad -0.0324 \quad -0.0450 \quad -0.0268 \quad -0.0222 \quad -0.0887]$$

$$C_s^f = \begin{bmatrix} 0.6900 & 0.1917 & 0.1426 \\ 0.6257 & 0.9892 & 0.0455 \\ 0.8372 & 0.2793 & 0.9156 \\ 0.9706 & 0.1680 & 0.2373 \\ 0.6166 & 0.8860 & 0.4350 \\ 0.8714 & 0.7500 & 0.6404 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.4685 & -0.1246 & 1.1818 \\ 1.0030 & 0.7672 & 0.4588 \\ 0.8335 & 0.3094 & 0.4544 \\ -0.0717 & 1.0539 & 1.0358 \\ 0.5153 & 1.0234 & -0.0828 \\ 0.8932 & 0.4719 & 0.5240 \end{bmatrix}$$

For recurrent MFS-W neuron model:

$$W^f = [0.3922 \quad 0.5125 \quad 0.4203 \quad 1.1142 \quad 1.0943 \quad 0.2814]$$

$$Q_{ws}^f = [0.0424 \quad 0.0462 \quad 0.0097 \quad 0.0429 \quad -0.0708 \quad 0.0104]$$

$$C_s^f = \begin{bmatrix} 0.6945 & 0.2019 & 0.1433 \\ 0.6298 & 0.9858 & 0.0485 \\ 0.8373 & 0.2631 & 0.9253 \\ 0.9895 & 0.1632 & 0.2436 \\ 0.6084 & 0.8923 & 0.4560 \\ 0.8722 & 0.7511 & 0.6385 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.6182 & 0.0175 & 1.2642 \\ 0.9425 & 0.7081 & 0.4014 \\ 0.6996 & 0.3164 & 0.3137 \\ -0.0071 & 1.0423 & 0.9892 \\ 0.5185 & 1.0050 & -0.0886 \\ 0.8956 & 0.4654 & 0.5303 \end{bmatrix}$$

For recurrent MRN neuron model:

$$W^f = [0.4493 \quad 0.4299 \quad 0.3876 \quad 1.0802 \quad 0.9642 \quad 0.3774]$$

$$Q^f = [-0.0497 \quad 0.1402 \quad -0.0012 \quad -0.1486 \quad 0.1668 \quad 0.1778]$$

$$C_s^f = \begin{bmatrix} 0.6870 & 0.2104 & 0.1476 \\ 0.6203 & 0.9812 & 0.0495 \\ 0.8381 & 0.2540 & 0.9263 \\ 0.9893 & 0.1657 & 0.2363 \\ 0.6030 & 0.9164 & 0.3859 \\ 0.8736 & 0.7564 & 0.638 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.6090 & 0.0564 & 1.2406 \\ 0.9576 & 0.7774 & 0.4294 \\ 0.7221 & 0.3710 & 0.2023 \\ 0.0033 & 0.8730 & 1.0577 \\ 0.5098 & 1.0019 & -0.1563 \\ 0.8862 & 0.5007 & 0.4449 \end{bmatrix}$$

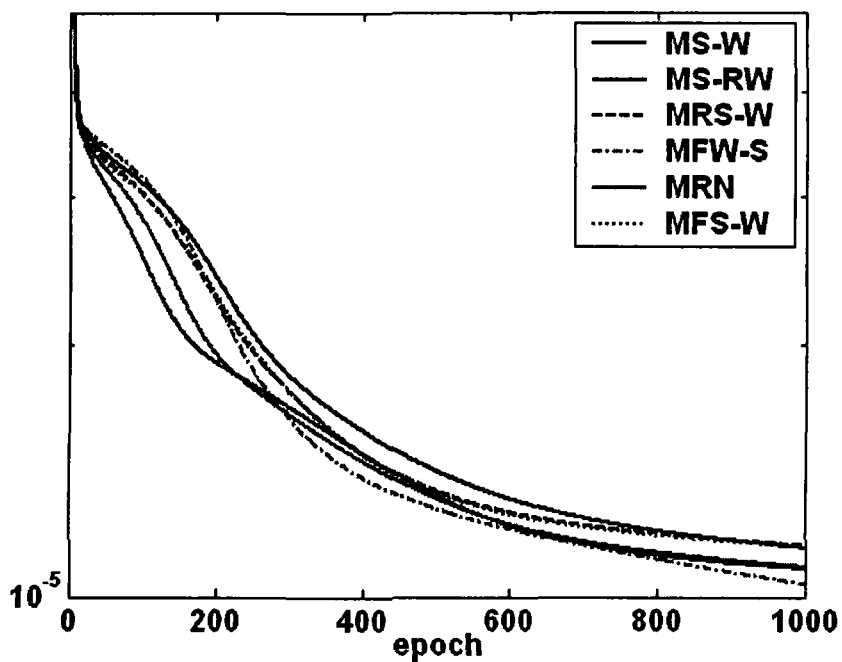


Fig. 6.15. Learning pattern of feed-forward network with recurrent MS-W neuron models for Example 2

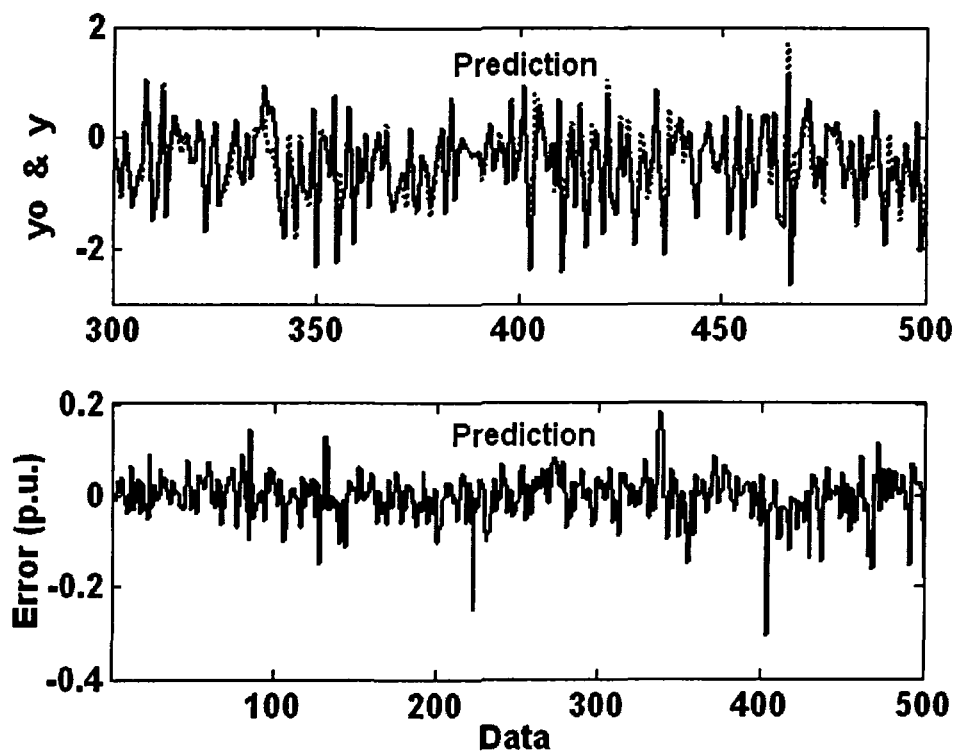


Fig. 6.16. Actual output and network output with MFW-S model and the error for Example 2

### Revisited Example 3: Non-Linear Regression with Non-Linear Input

Figure 6.17 shows that the learning pattern of different recurrent SS-W models and SS-W neuron model. This figure and Table 6.1 confers that SS-RW model acquiesces better performance with performance index  $J=6.780 \times 10^{-5}$ . The learning parameters of the different SS-W recurrent networks are as follows.

For recurrent SS-RW neuron model:

$$W^f = [0.2024 \quad -0.5754 \quad 1.1774] \quad Q_w^f = [-0.3545 \quad 0.3830 \quad -0.4477]$$
$$C_s^f = \begin{bmatrix} 0.4170 & 0.1409 \\ 0.8930 & 0.2077 \\ 0.0985 & 0.2266 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.3538 & 0.9240 \\ 0.1336 & 0.7471 \\ 1.5150 & 0.4963 \end{bmatrix}$$

For recurrent SRS-W neuron model:

$$W^f = [-0.1857 \quad -0.6370 \quad 1.5149] \quad Q_s^f = [-0.0066 \quad -0.0513 \quad 0.2438]$$
$$C_s^f = \begin{bmatrix} 0.4127 & 0.1439 \\ 0.8858 & 0.2146 \\ 0.1525 & 0.2266 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.6561 & 0.8826 \\ -0.1428 & 0.7984 \\ 1.4177 & -0.0874 \end{bmatrix}$$

For recurrent SFW-S neuron model:

$$W^f = [-0.0612 \quad -0.5232 \quad 1.3009] \quad Q_{sw}^f = [0.0729 \quad 0.4031 \quad -0.9716]$$
$$C_s^f = \begin{bmatrix} 0.4158 & 0.1437 \\ 0.9004 & 0.2181 \\ 0.0941 & 0.2066 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.5720 & 0.9767 \\ -0.2987 & 0.7164 \\ 1.5651 & 0.0971 \end{bmatrix}$$

For recurrent SFS-W neuron model:

$$W^f = [-0.1879 \quad -0.6429 \quad 1.5074]$$

$$Q_{ws}^f = [0.0519 \quad 0.2605 \quad 0.2611]$$

$$C_s^f = \begin{bmatrix} 0.4126 & 0.1440 \\ 0.8825 & 0.2117 \\ 0.1524 & 0.2266 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.6641 & 0.8874 \\ -0.1568 & 0.7938 \\ 1.4336 & -0.0770 \end{bmatrix}$$

For recurrent SRN neuron model:

$$W^f = [0.2384 \quad -0.6434 \quad 1.0101]$$

$$Q^f = [-0.5980 \quad 0.8200 \quad -0.3562]$$

$$C_s^f = \begin{bmatrix} 0.4129 & 0.1422 \\ 0.8955 & 0.2035 \\ 0.0941 & 0.2257 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.7107 & 0.9037 \\ 0.2756 & 0.8195 \\ 1.4850 & 0.4119 \end{bmatrix}$$

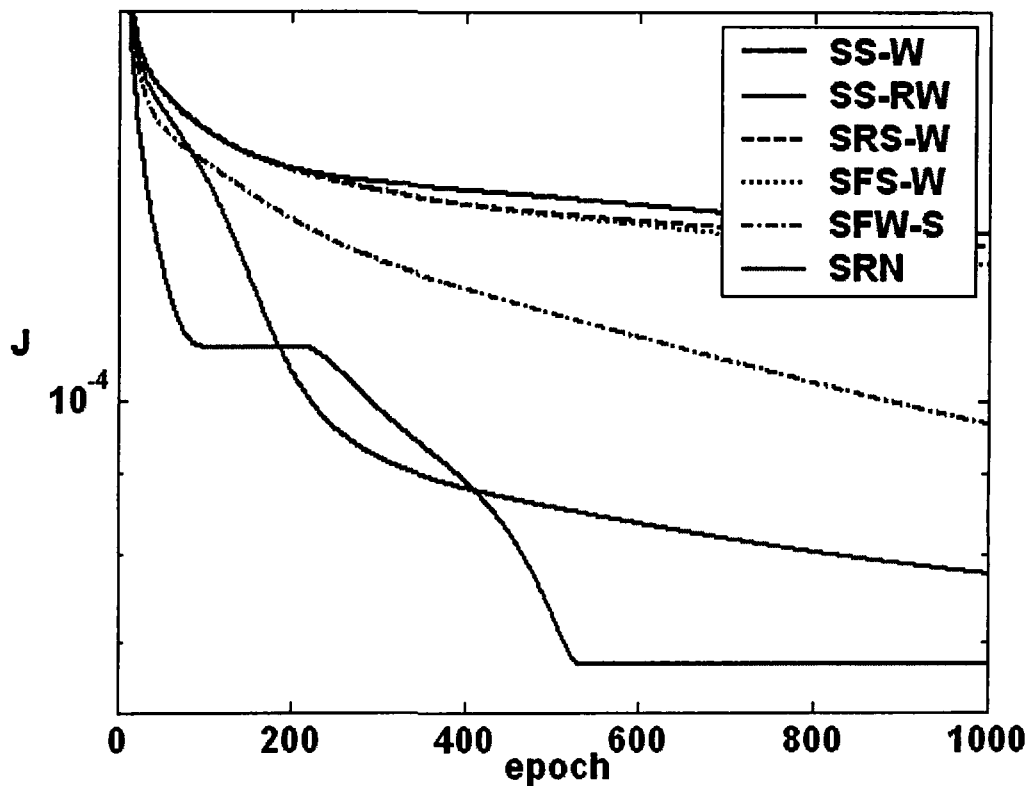


Fig. 6.17. Learning pattern of feed-forward network with recurrent SS-W neuron models for Example 3



Figure 6.18 and Table 6.2 show the learning pattern and performance index of different recurrent MS-W models and MS-W neuron model. MFW-S model acquires better performance with  $J=9.053 \times 10^{-5}$ . Next to MFW-S, MR-N and MS-RW have better performance. Figure 6.19 illustrate system and network output of MFW-S neuron model as well as the error between them. The learning parameters of the different recurrent network are given follows.

For recurrent MS-RW neuron model:

$$W^f = [-0.5088 \quad 0.5206 \quad 0.7630] \quad Q_w^f = [0.0828 \quad -0.6553 \quad -0.2861]$$

$$C_s^f = \begin{bmatrix} 0.1866 & 0.7425 \\ 0.7252 & 0.4753 \\ 0.7185 & 0.5668 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.7804 & 0.1132 \\ 0.5089 & 0.1940 \\ 0.6318 & 0.3624 \end{bmatrix}$$

For recurrent MRS-W neuron model:

$$W^f = [-0.7121 \quad 0.2410 \quad 0.7390] \quad Q_s^f = [0.2001 \quad 0.1464 \quad 0.1617]$$

$$C_s^f = \begin{bmatrix} 0.1744 & 0.6949 \\ 0.7310 & 0.4534 \\ 0.6894 & 0.5671 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 1.0024 & -0.0174 \\ 0.1262 & 0.0311 \\ 0.6552 & 0.2242 \end{bmatrix}$$

For recurrent MFW-S neuron model:

$$W^f = [-0.5266 \quad 0.4847 \quad 0.4822] \quad Q_{sw}^f = [-0.6001 \quad -0.3007 \quad -0.0250]$$

$$C_s^f = \begin{bmatrix} 0.1260 & 0.6279 \\ 0.7089 & 0.4955 \\ 0.6656 & 0.5987 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 1.0142 & 0.2498 \\ 0.2532 & 0.0844 \\ 0.3081 & 0.6824 \end{bmatrix}$$

For recurrent MFS-W neuron model:

$$W^f = [-0.7326 \quad 0.2392 \quad 0.7448]$$

$$Q_{ws}^f = [0.2296 \quad 0.0009 \quad 0.1324]$$

$$C_s^f = \begin{bmatrix} 0.1734 & 0.6964 \\ 0.7346 & 0.4579 \\ 0.6911 & 0.5750 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 1.0189 & -0.0228 \\ 0.1288 & 0.0262 \\ 0.6347 & 0.2358 \end{bmatrix}$$

For recurrent MRN neuron model:

$$W^f = [-0.5290 \quad 0.6739 \quad 0.8909]$$

$$Q^f = [-0.1352 \quad -1.0723 \quad -0.5520]$$

$$C_s^f = \begin{bmatrix} 0.1685 & 0.7340 \\ 0.7717 & 0.5126 \\ 0.7251 & 0.5862 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.7934 & -0.0181 \\ 0.4401 & 0.2042 \\ 0.5856 & 0.2934 \end{bmatrix}$$

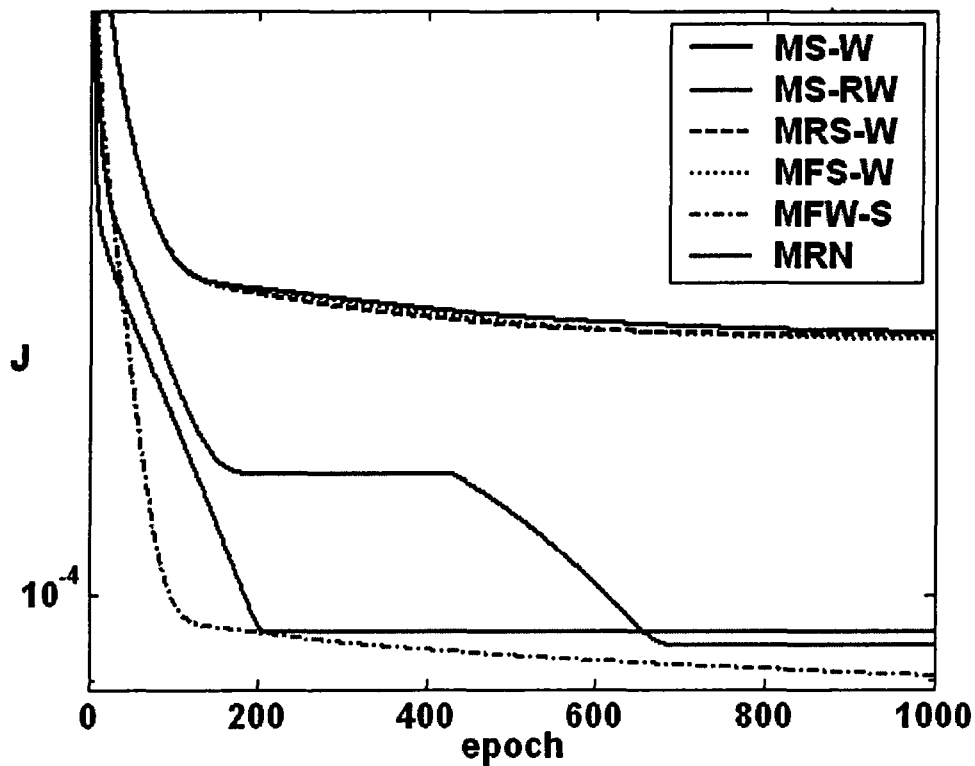


Fig. 6.18. Learning pattern of feed-forward network with recurrent MS-W neuron models for Example 3

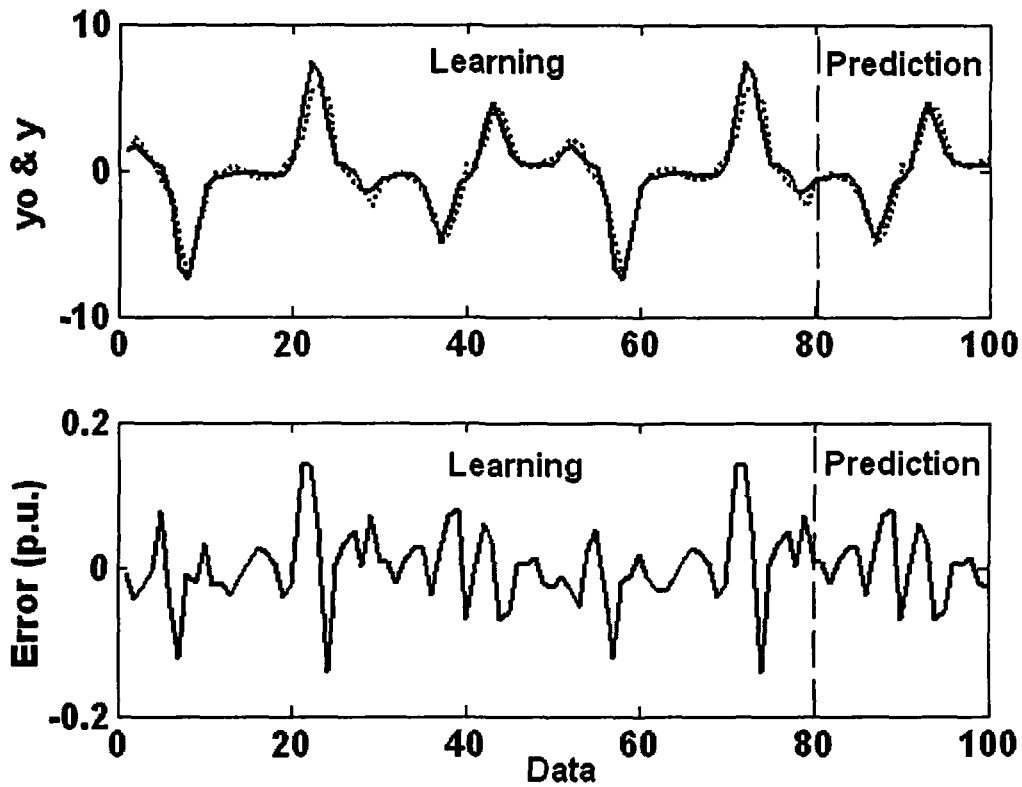


Fig. 6.19. Actual output and network output with SS-RW model and the error for Example 3

#### Revisited Example 4: *Non-linear Regression of Input and output*

Figure 6.20 and Table 6.1 show that the SRN model yields better result than other recurrent models with performance index  $J=9.824 \times 10^{-7}$ . Next to SRN model, SS-RW and SFW-S models are better. Learning parameters of the recurrent summation network are as follows.

For recurrent SS-RW neuron model:

$$W^f = [-0.2980 \quad 0.2765 \quad 1.1226]$$

$$Q_w^f = [0.1266 \quad -0.0947 \quad -0.2375]$$

$$C_s^f = \begin{bmatrix} 0.1673 & 0.6126 \\ 0.2540 & 0.3389 \\ 0.2492 & 0.3203 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.5524 & 0.5587 \\ 0.9275 & 0.9231 \\ 0.4877 & 0.6794 \end{bmatrix}$$

For recurrent SRS-W neuron model:

$$W^f = [-0.2935 \quad 0.5270 \quad 1.0867]$$

$$Q_{S}^f = [0.2318 \quad -0.4205 \quad -0.9303]$$

$$C_{S}^f = \begin{bmatrix} 0.1773 & 0.6167 \\ 0.2257 & 0.3360 \\ 0.1665 & 0.2765 \end{bmatrix}$$

$$C_{W}^f = \begin{bmatrix} 0.6096 & 0.7516 \\ 0.9606 & 1.1120 \\ 0.7749 & 0.4434 \end{bmatrix}$$

For recurrent SFW-S neuron model:

$$W^f = [-0.3127 \quad 0.4535 \quad 1.0243]$$

$$Q_{SW}^f = [0.3264 \quad -0.1772 \quad -0.4966]$$

$$C_{S}^f = \begin{bmatrix} 0.1646 & 0.6139 \\ 0.2562 & 0.3413 \\ 0.2534 & 0.2965 \end{bmatrix}$$

$$C_{W}^f = \begin{bmatrix} 0.4005 & 0.4878 \\ 0.9198 & 1.0168 \\ 0.7770 & 0.5688 \end{bmatrix}$$

For recurrent SFS-W neuron model:

$$W^f = [-0.2768 \quad 0.4240 \quad 1.1041]$$

$$Q_{WS}^f = [0.1110 \quad -0.3262 \quad -0.5268]$$

$$C_{S}^f = \begin{bmatrix} 0.1589 & 0.6148 \\ 0.2655 & 0.3409 \\ 0.2752 & 0.3009 \end{bmatrix}$$

$$C_{W}^f = \begin{bmatrix} 0.5811 & 0.7354 \\ 1.0302 & 0.9997 \\ 0.6747 & 0.4743 \end{bmatrix}$$

For recurrent SRN neuron model:

$$W^f = [-0.3768 \quad 0.3959 \quad 1.0940]$$

$$Q^f = [0.2964 \quad -0.1282 \quad -0.2628]$$

$$C_{S}^f = \begin{bmatrix} 0.1541 & 0.6183 \\ 0.2638 & 0.3289 \\ 0.2775 & 0.2730 \end{bmatrix}$$

$$C_{W}^f = \begin{bmatrix} 0.4475 & 0.6136 \\ 1.0185 & 0.9459 \\ 0.6938 & 0.5948 \end{bmatrix}$$

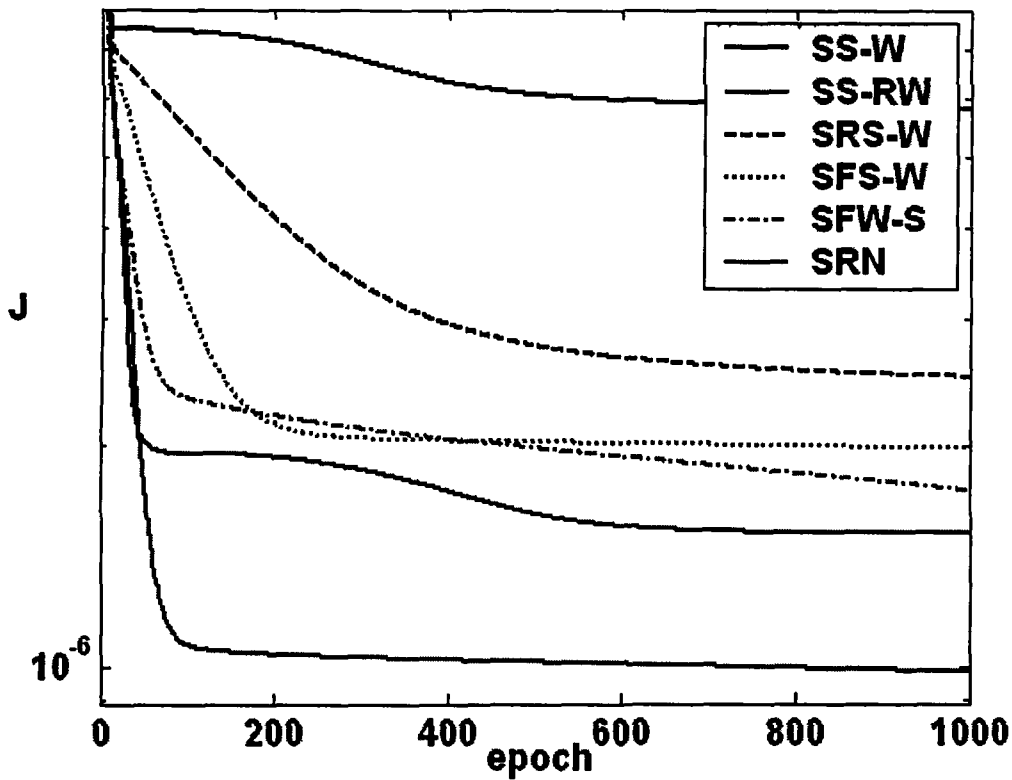


Fig. 6.20. Learning pattern of feed-forward network with recurrent SS-W neuron models for Example 4

Figure 6.21 shows that in recurrent MS-W neuron models, the MFW-S network has better result than other recurrent networks. As shown in Table 6.2 the performance index of this model is  $J=1.320 \times 10^{-6}$ . Next to MFW-S network, MS-RW and MRN networks are better. Network output (with SRN neuron model) and the system output with error between them is shown in Fig. 6.22. Learning parameters of the recurrent MW-S network are as follows.

For recurrent MS-RW neuron model:

$$W^f = [0.2244 \quad 0.6028 \quad 1.2509] \quad Q_w^f = [-0.0840 \quad -0.1840 \quad -0.0720]$$

$$C_s^f = \begin{bmatrix} 0.2641 & 0.2703 \\ 0.1149 & 0.6295 \\ 0.7313 & 1.0391 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.4903 & 0.5522 \\ 0.5389 & 0.4530 \\ 0.4755 & 0.7396 \end{bmatrix}$$

For recurrent MRS-W neuron model:

$$W^f = [0.4615 \quad 0.7514 \quad 2.0308]$$

$$Q_s^f = [0.0662 \quad 0.2127 \quad -0.2387]$$

$$C_s^f = \begin{bmatrix} 0.4307 & 0.4525 \\ 0.3826 & 0.8448 \\ 0.3789 & 0.7182 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.0453 & 0.6066 \\ 0.6875 & 0.6239 \\ 0.5311 & 0.6046 \end{bmatrix}$$

For recurrent MFW-S neuron model:

$$W^f = [0.4547 \quad 0.7641 \quad 1.8686]$$

$$Q_{sw}^f = [-0.2560 \quad 0.1754 \quad -0.2684]$$

$$C_s^f = \begin{bmatrix} 0.4637 & 0.5243 \\ 0.3286 & 0.8680 \\ 0.4899 & 0.7130 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.1276 & 0.7714 \\ 0.6232 & 0.3968 \\ 0.5051 & 0.9219 \end{bmatrix}$$

For recurrent MFS-W neuron model:

$$W^f = [0.5257 \quad 0.7500 \quad 1.9448]$$

$$Q_{ws}^f = [0.0712 \quad 0.1454 \quad -0.2262]$$

$$C_s^f = \begin{bmatrix} 0.4272 & 0.4572 \\ 0.3511 & 0.8261 \\ 0.5446 & 0.8501 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.0329 & 0.6211 \\ 0.7003 & 0.6371 \\ 0.5227 & 0.6181 \end{bmatrix}$$

For recurrent MRN neuron model:

$$W^f = [0.3634 \quad 0.5970 \quad 1.3249]$$

$$Q^f = [-0.1444 \quad -0.3730 \quad -0.0414]$$

$$C_s^f = \begin{bmatrix} 0.2726 & 0.2749 \\ 0.1143 & 0.6018 \\ 0.7449 & 1.0679 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.4217 & 0.5909 \\ 0.6335 & 0.4248 \\ 0.4458 & 0.8089 \end{bmatrix}$$

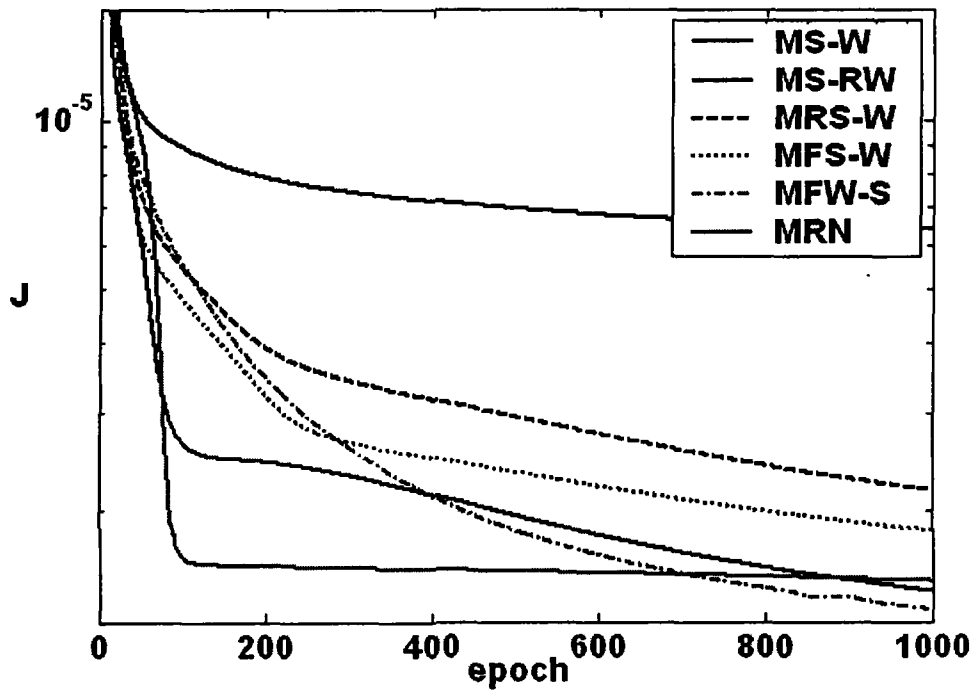


Fig. 6.21. Learning pattern of feed-forward network with recurrent MS-W neuron models for Example 4

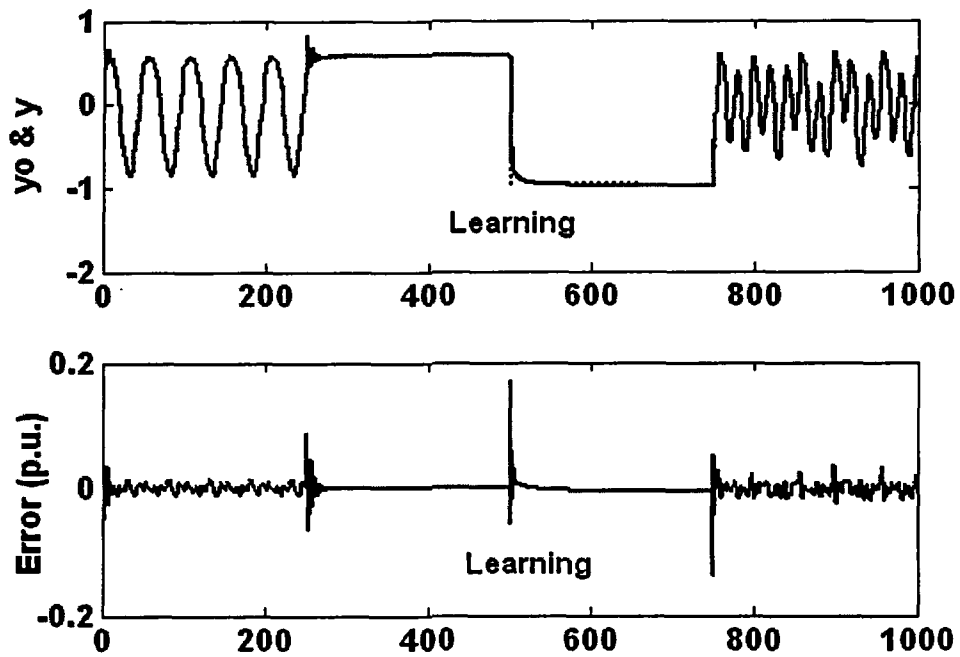


Fig. 6.22. Actual output and network output with SRN model and the error for Example 4

### Revisited Example 5: Gas Furnace Data

From the Fig. 6.23 SS-RW yields better result than other recurrent SS-W neuron models with performance index, as shown in Table 6.1,  $J=1.405 \times 10^{-7}$ . Next to SS-RW model, SRN and SFW-S have better performances. Learning parameters of the recurrent SS-W neuron model are as follows.

For recurrent SS-RW neuron model:

$$W^f = [0.8219 \quad 0.1574 \quad 0.6369 \quad 0.3447 \quad 0.6124 \quad 0.9757]$$

$$Q_w^f = [-0.0703 \quad -0.0791 \quad 0.1707 \quad 0.1572 \quad 0.2236 \quad -0.0063]$$

$$C_s^f = \begin{bmatrix} 0.0995 & -0.0228 & 0.8470 \\ 0.1944 & 0.7387 & 0.5276 \\ 0.1683 & 0.4155 & 0.2103 \\ 0.5875 & 0.9160 & 0.6828 \\ 0.2413 & 0.4361 & 0.8471 \\ 0.1543 & 0.3752 & 0.0313 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.0733 & 0.5494 & 0.8444 \\ 0.0764 & 0.7942 & 0.8905 \\ 0.8380 & 0.1956 & -0.3959 \\ 0.5290 & 0.4882 & 0.3944 \\ 0.6120 & 0.9367 & 1.0686 \\ 0.8808 & 1.0045 & -0.3664 \end{bmatrix}$$

For recurrent SRS-W neuron model:

$$W^f = [0.9527 \quad -0.0335 \quad 0.5899 \quad 0.3333 \quad 0.6765 \quad 1.0553]$$

$$Q_s^f = [-0.0066 \quad 0.0031 \quad -0.0021 \quad -0.00002 \quad -0.0016 \quad -0.0098]$$

$$C_s^f = \begin{bmatrix} 0.0899 & -0.0320 & 0.8731 \\ 0.1993 & 0.7434 & 0.5275 \\ 0.1672 & 0.4146 & 0.2316 \\ 0.5852 & 0.9139 & 0.6920 \\ 0.2345 & 0.4296 & 0.8660 \\ 0.1474 & 0.3690 & 0.0672 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.2004 & 0.6780 & 0.6519 \\ 0.0449 & 0.7639 & 0.9310 \\ 0.7755 & 0.1303 & -0.5126 \\ 0.5264 & 0.4848 & 0.3451 \\ 0.5273 & 0.8503 & 0.9611 \\ 0.9061 & 1.0261 & -0.4920 \end{bmatrix}$$

For recurrent SFW-S neuron model:

$$W^f = [0.9405 \quad 0.0007 \quad 0.6091 \quad 0.3448 \quad 0.6692 \quad 1.0368]$$



$$Q_{sw}^f = [-0.0036 \quad 0.0363 \quad -0.1348 \quad 0.1009 \quad 0.0955 \quad 0.0290]$$

$$C_s^f = \begin{bmatrix} 0.0918 & -0.0302 & 0.8692 \\ 0.1982 & 0.7424 & 0.5282 \\ 0.1677 & 0.4150 & 0.2285 \\ 0.5855 & 0.9141 & 0.6906 \\ 0.2359 & 0.4309 & 0.8629 \\ 0.1495 & 0.3709 & 0.0610 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.1825 & 0.6602 & 0.6793 \\ 0.0513 & 0.7699 & 0.9286 \\ 0.7884 & 0.1434 & -0.5027 \\ 0.5308 & 0.4892 & 0.3474 \\ 0.5449 & 0.8676 & 0.9571 \\ 0.8987 & 1.0194 & -0.4705 \end{bmatrix}$$

For recurrent SFS-W neuron model:

$$W^f = [0.9542 \quad -0.0348 \quad 0.5881 \quad 0.3329 \quad 0.6786 \quad 1.0550]$$

$$Q_{ws}^f = [-0.0354 \quad 0.0078 \quad -0.0181 \quad -0.0088 \quad -0.0304 \quad -0.0680]$$

$$C_s^f = \begin{bmatrix} 0.0902 & -0.0318 & 0.8731 \\ 0.1993 & 0.7434 & 0.5276 \\ 0.1675 & 0.4148 & 0.2316 \\ 0.5854 & 0.9140 & 0.6920 \\ 0.2348 & 0.4298 & 0.8659 \\ 0.1479 & 0.3694 & 0.0671 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.2057 & 0.6830 & 0.6501 \\ 0.0448 & 0.7638 & 0.9309 \\ 0.7745 & 0.1295 & -0.5087 \\ 0.5257 & 0.4843 & 0.3440 \\ 0.5273 & 0.8503 & 0.9580 \\ 0.9047 & 1.0250 & -0.4903 \end{bmatrix}$$

For recurrent SRN neuron model:

$$W^f = [0.9277 \quad 0.0360 \quad 0.5890 \quad 0.3370 \quad 0.6569 \quad 1.0244]$$

$$Q^f = [-0.1518 \quad 0.0106 \quad 0.0054 \quad 0.0456 \quad 0.1385 \quad -0.0249]$$

$$C_s^f = \begin{bmatrix} 0.0918 & -0.0295 & 0.8638 \\ 0.1977 & 0.7420 & 0.5280 \\ 0.1678 & 0.4156 & 0.2236 \\ 0.5860 & 0.9149 & 0.6880 \\ 0.2371 & 0.4325 & 0.8581 \\ 0.1494 & 0.3716 & 0.0533 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.1274 & 0.6110 & 0.7297 \\ 0.0552 & 0.7734 & 0.9238 \\ 0.7818 & 0.1364 & -0.4985 \\ 0.5265 & 0.4839 & 0.3659 \\ 0.5626 & 0.8855 & 0.9937 \\ 0.8976 & 1.0161 & -0.4478 \end{bmatrix}$$

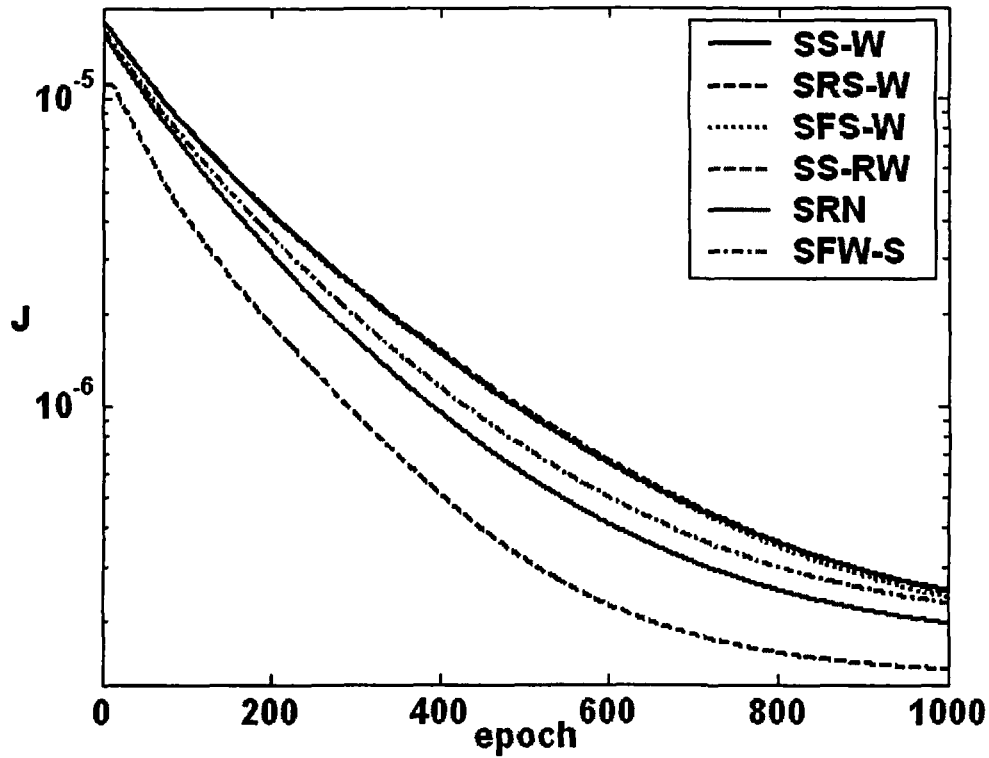


Fig. 6.23. Learning pattern of feed-forward network with recurrent SS-W neuron models for Example 5

From the Fig. 6.24 MFW-S yield better learning pattern. The performance index of this model as shown in Table 6.2 is  $J=9.738 \times 10^{-8}$ . Next to MFW-S network, MS-RW has better performances. Figure 6.25 shows actual and network with MFW-S network as well as error. Learning parameters of the MFW-S recurrent networks are as follows.

For recurrent MS-RW neuron model:

$$W^f = [0.8039 \quad 0.1272 \quad 0.6507]$$

$$Q_w^f = [0.0514 \quad 0.0022 \quad 0.0068]$$

$$C_s^f = \begin{bmatrix} 0.1331 & 0.1680 & 0.6760 \\ 0.0681 & 0.9113 & 0.2672 \\ 0.1288 & 0.1396 & 0.3051 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.1997 & 0.2138 & -0.0373 \\ 0.6465 & 0.6624 & 0.6043 \\ 0.0566 & 0.3145 & 1.0313 \end{bmatrix}$$

For MRS-W recurrent network:

$$W^f = [0.9120 \quad 0.0835 \quad 0.8625]$$

$$Q_s^f = [-0.0052 \quad -0.0019 \quad 0.0284]$$

$$C_s^f = \begin{bmatrix} 0.1151 & 0.1452 & 0.7416 \\ 0.0677 & 0.9107 & 0.2665 \\ 0.1156 & 0.1225 & 0.4071 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.3250 & 0.3746 & 0.0745 \\ 0.6611 & 0.6803 & 0.5598 \\ -0.0238 & 0.2133 & 0.7597 \end{bmatrix}$$

For recurrent MFW-S network:

$$W^f = [0.9029 \quad 0.0812 \quad 0.8557]$$

$$Q_{sw}^f = [0.2783 \quad -0.0235 \quad -0.0680]$$

$$C_s^f = \begin{bmatrix} 0.1134 & 0.1454 & 0.7408 \\ 0.0676 & 0.9107 & 0.2666 \\ 0.1151 & 0.1236 & 0.4046 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.2865 & 0.3188 & -0.1471 \\ 0.6626 & 0.6805 & 0.5600 \\ -0.0270 & 0.2194 & 0.7761 \end{bmatrix}$$

For recurrent MFS-W neuron model:

$$W^f = [0.9116 \quad 0.0880 \quad 0.8594]$$

$$Q_{ws}^f = [-0.0228 \quad -0.0029 \quad 0.0239]$$

$$C_s^f = \begin{bmatrix} 0.1149 & 0.1449 & 0.7415 \\ 0.0677 & 0.9107 & 0.2667 \\ 0.1155 & 0.1224 & 0.4055 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.3246 & 0.3743 & 0.0732 \\ 0.6613 & 0.6807 & 0.5597 \\ -0.0254 & 0.2114 & 0.7591 \end{bmatrix}$$

For recurrent MRN neuron model:

$$W^f = [0.8999 \quad 0.1467 \quad 0.7803]$$

$$Q^f = [0.2818 \quad -0.0196 \quad 0.0087]$$

$$C_s^f = \begin{bmatrix} 0.1129 & 0.1448 & 0.7214 \\ 0.0678 & 0.9108 & 0.2691 \\ 0.1128 & 0.1211 & 0.3742 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.3206 & 0.3603 & -0.1192 \\ 0.6648 & 0.6838 & 0.5724 \\ -0.0265 & 0.2192 & 0.7740 \end{bmatrix}$$

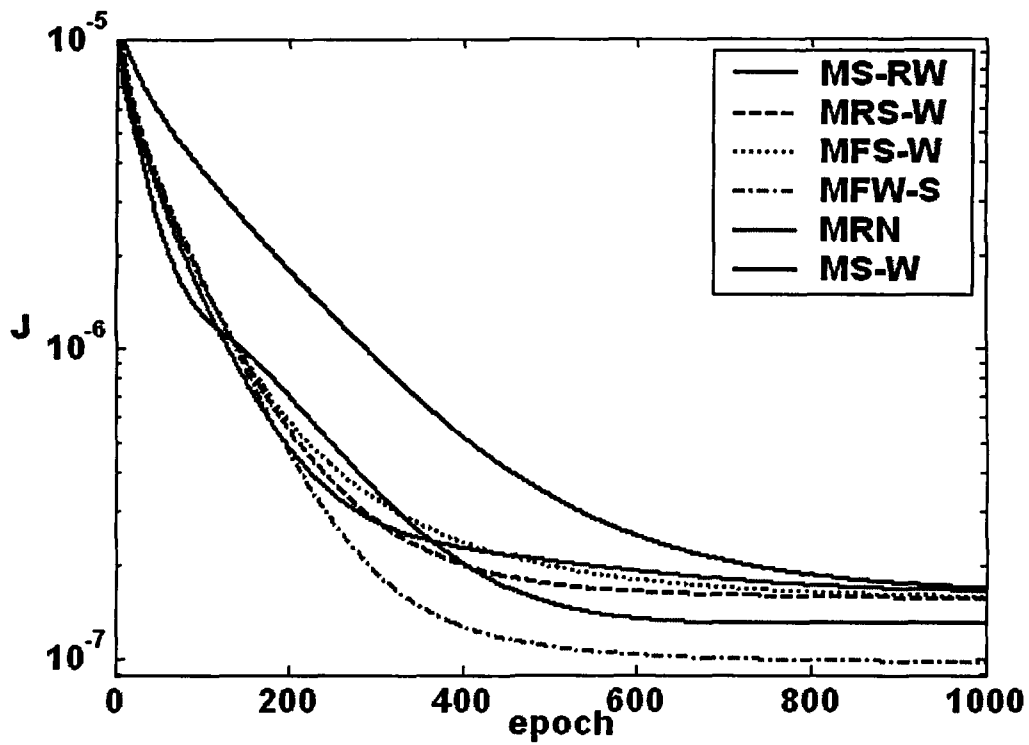


Fig. 6.24. Learning pattern of feed-forward network with recurrent MS-W neuron models for Example 5

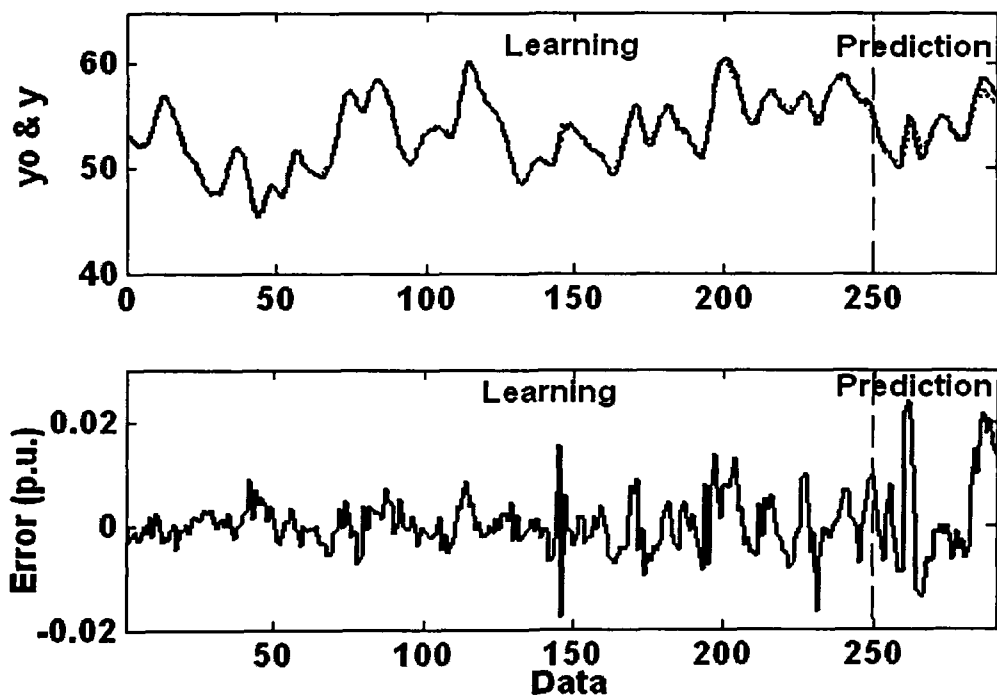


Fig. 6.25. Actual output and network output with MFW-S model and the error for Example 5

### Revisited Example 6: Human Operation at a Chemical Plant

Figure 6.26 shows that the learning pattern of the recurrent SS-W network with SS-RW neuron yields better result with performance index  $J=7.832 \times 10^{-6}$ . Table 6.1 show the performance index of the different recurrent SS-W models. Learning parameters of the SS-W recurrent networks are as follows.

For recurrent SS-RW neuron model:

$$W^f = [-0.1907 \quad 0.0266 \quad 0.8936 \quad 0.0965 \quad 0.7763 \quad 0.3453]$$

$$Q_w^f = [0.0247 \quad -0.0414 \quad -0.0684 \quad -0.0485 \quad 0.1958 \quad -0.1399]$$

$$C_s^f = \begin{bmatrix} 0.0150 & 0.4910 \\ 0.7679 & 0.2224 \\ 0.9707 & 0.6945 \\ 0.9900 & 0.3306 \\ 0.7888 & 0.9991 \\ 0.4386 & 0.7456 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.3797 & 0.1124 \\ 0.7833 & 0.5154 \\ 0.6804 & 0.3814 \\ 0.4611 & 0.3790 \\ 0.5674 & 0.6367 \\ 0.7942 & 0.9453 \end{bmatrix}$$

For recurrent SRS-W neuron model:

$$W^f = [-0.2109 \quad 0.0160 \quad 0.9114 \quad 0.0659 \quad 0.8823 \quad 0.2137]$$

$$Q_s^f = [-0.0080 \quad 0.0117 \quad 0.0769 \quad 0.0162 \quad 0.0550 \quad 0.0366]$$

$$C_s^f = \begin{bmatrix} 0.0150 & 0.4913 \\ 0.7679 & 0.2267 \\ 0.9707 & 0.7162 \\ 0.9900 & 0.3359 \\ 0.7887 & 1.0230 \\ 0.4386 & 0.7516 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.3798 & 0.0465 \\ 0.7833 & 0.5024 \\ 0.6805 & 0.3709 \\ 0.4610 & 0.3788 \\ 0.5674 & 0.8701 \\ 0.7942 & 0.8780 \end{bmatrix}$$

For recurrent SFW-S neuron model:

$$W^f = [-0.1887 \quad 0.0224 \quad 0.8852 \quad 0.0839 \quad 0.7864 \quad 0.2678]$$

$$Q_{sw}^f = [-0.0059 \quad -0.0157 \quad 0.0519 \quad -0.0075 \quad 0.2344 \quad 0.0122]$$

$$C_s^f = \begin{bmatrix} 0.0150 & 0.4909 \\ 0.7679 & 0.2228 \\ 0.9707 & 0.6967 \\ 0.9900 & 0.3304 \\ 0.7887 & 1.0029 \\ 0.4386 & 0.7447 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.3797 & 0.0722 \\ 0.7833 & 0.5227 \\ 0.6803 & 0.3750 \\ 0.4611 & 0.3849 \\ 0.5673 & 0.7241 \\ 0.7942 & 0.9070 \end{bmatrix}$$

For recurrent SFS-W neuron model:

$$W^f = [-0.2061 \quad 0.0172 \quad 0.9038 \quad 0.0703 \quad 0.8761 \quad 0.2137]$$

$$Q_{ws}^f = [-0.0007 \quad 0.0059 \quad 0.0181 \quad 0.0055 \quad 0.0295 \quad 0.0245]$$

$$C_s^f = \begin{bmatrix} 0.0150 & 0.4896 \\ 0.7679 & 0.2240 \\ 0.9707 & 0.7146 \\ 0.9900 & 0.3337 \\ 0.7887 & 1.0212 \\ 0.4386 & 0.7522 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.3798 & 0.0505 \\ 0.7833 & 0.5099 \\ 0.6804 & 0.3761 \\ 0.4611 & 0.3817 \\ 0.5673 & 0.8755 \\ 0.7942 & 0.8826 \end{bmatrix}$$

For recurrent SRN neuron model:

$$W^f = [-0.2225 \quad 0.0437 \quad 0.9071 \quad 0.0700 \quad 0.8463 \quad 0.2475]$$

$$Q^f = [0.0135 \quad -0.0717 \quad 0.00004 \quad -0.0497 \quad 0.0396 \quad -0.0468]$$

$$C_s^f = \begin{bmatrix} 0.0150 & 0.4886 \\ 0.7679 & 0.2238 \\ 0.9707 & 0.7099 \\ 0.9900 & 0.3323 \\ 0.7888 & 1.0165 \\ 0.4386 & 0.7500 \end{bmatrix}$$

$$C_w^f = \begin{bmatrix} 0.3797 & 0.0773 \\ 0.7833 & 0.5154 \\ 0.6805 & 0.3875 \\ 0.4610 & 0.3822 \\ 0.5674 & 0.8420 \\ 0.7942 & 0.8976 \end{bmatrix}$$

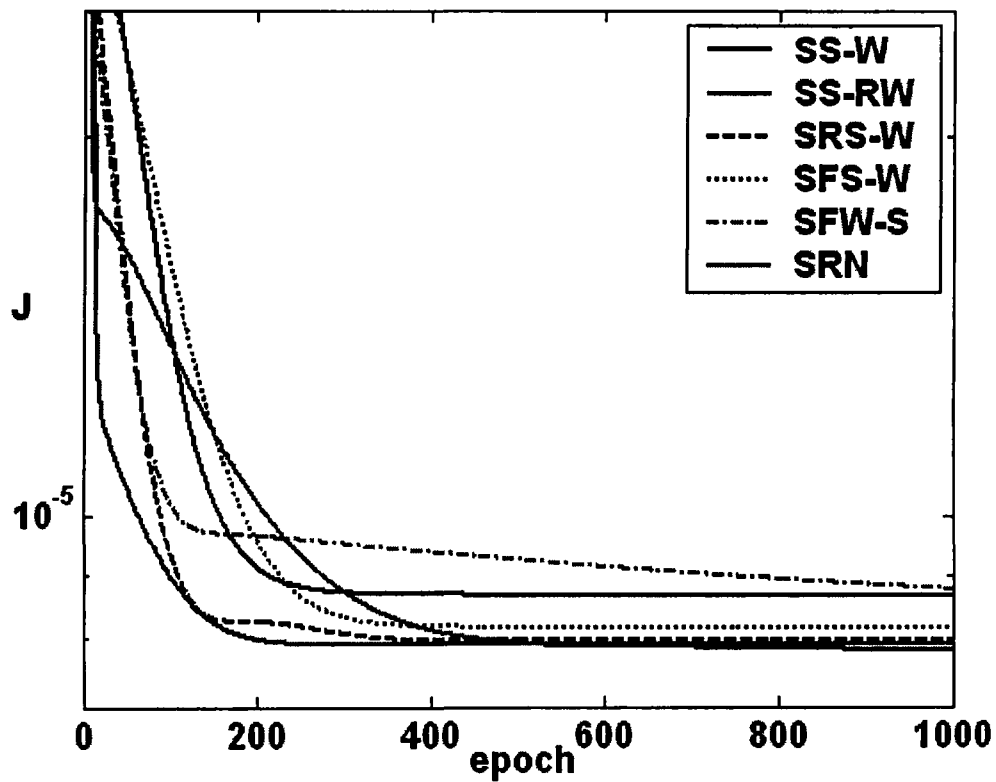


Fig. 6.26. Learning pattern of feed-forward network with recurrent SS-W neuron models for Example 6

Figure 6.27 shows that the learning pattern of the recurrent MS-W network with MS-RW neuron yields better result with performance index  $J=6.190 \times 10^{-6}$  as shown in Table 6.2. Learning parameters of the MS-W recurrent network are as follows.

For recurrent MS-RW neuron model:

$$W^f = [0.1292 \quad 0.6108 \quad 0.5087 \quad 0.2991 \quad 0.9034 \quad 0.9322]$$

$$Q_w^f = [0.0169 \quad -0.1169 \quad -0.1769 \quad -0.0720 \quad -0.0636 \quad -0.0936]$$

$$C_s^f = \begin{bmatrix} 0.7035 & 0.5454 \\ 0.4848 & 0.8529 \\ 0.1146 & 0.6661 \\ 0.6648 & 1.0111 \\ 0.3653 & 1.0134 \\ 0.1401 & 0.0308 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.3717 & 0.6686 \\ 0.4253 & 0.3327 \\ 0.5943 & 0.3298 \\ 0.5657 & 0.6159 \\ 0.7161 & 1.0572 \\ 0.5112 & 0.9514 \end{bmatrix}$$

For recurrent MRS-W neuron model:

$$W^f = [0.1048 \quad 0.6058 \quad 0.4824 \quad 0.3036 \quad 0.9293 \quad 0.9226]$$

$$Q_s^f = [-0.0140 \quad 0.0428 \quad -0.0186 \quad 0.0216 \quad 0.0437 \quad -0.0469]$$

$$C_s^f = \begin{bmatrix} 0.7035 & 0.5461 \\ 0.4848 & 0.8514 \\ 0.1146 & 0.6642 \\ 0.6648 & 1.0121 \\ 0.3653 & 1.0193 \\ 0.1401 & 0.0252 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.3717 & 0.6996 \\ 0.4254 & 0.2656 \\ 0.5943 & 0.3397 \\ 0.5658 & 0.6045 \\ 0.7160 & 1.0946 \\ 0.5112 & 0.9534 \end{bmatrix}$$

For recurrent MFW-S neuron model:

$$W^f = [0.1319 \quad 0.5972 \quad 0.4949 \quad 0.2975 \quad 0.9156 \quad 0.9215]$$

$$Q_{sw}^f = [0.0084 \quad -0.1283 \quad 0.0672 \quad -0.0650 \quad 0.0761 \quad 0.0050]$$

$$C_s^f = \begin{bmatrix} 0.7035 & 0.5477 \\ 0.4849 & 0.8525 \\ 0.1146 & 0.6628 \\ 0.6648 & 1.0127 \\ 0.3653 & 1.0073 \\ 0.1401 & 0.0249 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.3716 & 0.7698 \\ 0.4252 & 0.3133 \\ 0.5945 & 0.3249 \\ 0.5657 & 0.6241 \\ 0.7163 & 1.1123 \\ 0.5113 & 0.9183 \end{bmatrix}$$

For recurrent MFS-W neuron model:

$$W^f = [0.1249 \quad 0.6001 \quad 0.4934 \quad 0.3011 \quad 0.9587 \quad 0.9189]$$

$$Q_{ws}^f = [-0.0112 \quad 0.0111 \quad -0.0033 \quad 0.0092 \quad 0.0378 \quad -0.0287]$$



$$C_s^f = \begin{bmatrix} 0.7035 & 0.5487 \\ 0.4848 & 0.8578 \\ 0.1146 & 0.6604 \\ 0.6648 & 1.0154 \\ 0.3653 & 1.0124 \\ 0.1401 & 0.0201 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.3717 & 0.7456 \\ 0.4253 & 0.2714 \\ 0.5944 & 0.3473 \\ 0.5657 & 0.6087 \\ 0.7161 & 1.1211 \\ 0.5113 & 0.9358 \end{bmatrix}$$

For recurrent MRN neuron model:

$$W^f = [0.1110 \quad 0.6070 \quad 0.4954 \quad 0.3041 \quad 0.9330 \quad 0.9216]$$

$$Q^f = [-0.0396 \quad -0.0330 \quad -0.1087 \quad -0.0232 \quad 0.0249 \quad 0.0255]$$

$$C_s^f = \begin{bmatrix} 0.7035 & 0.5456 \\ 0.4848 & 0.8545 \\ 0.1146 & 0.6641 \\ 0.6648 & 1.0126 \\ 0.3653 & 1.0176 \\ 0.1401 & 0.0248 \end{bmatrix} \quad C_w^f = \begin{bmatrix} 0.3717 & 0.7407 \\ 0.4253 & 0.2945 \\ 0.5944 & 0.3480 \\ 0.5657 & 0.6094 \\ 0.7162 & 1.0879 \\ 0.5112 & 0.9560 \end{bmatrix}$$

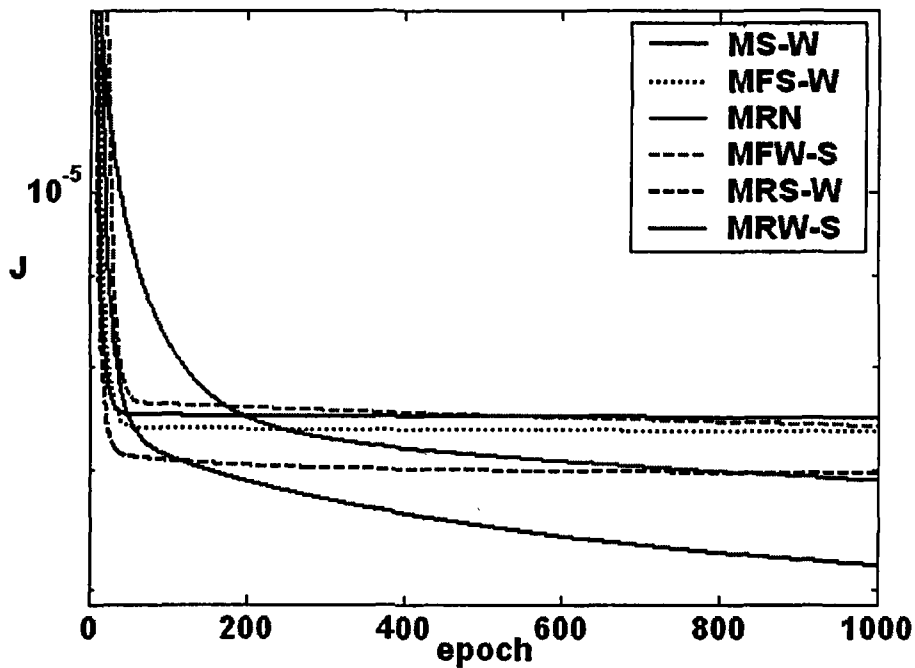


Fig. 6.27. Learning pattern of feed-forward network with recurrent MS-W neuron models for Example 6

Figure 6.28 shows the actual control action of the operator and the network output with error for MS-RW neuron model.

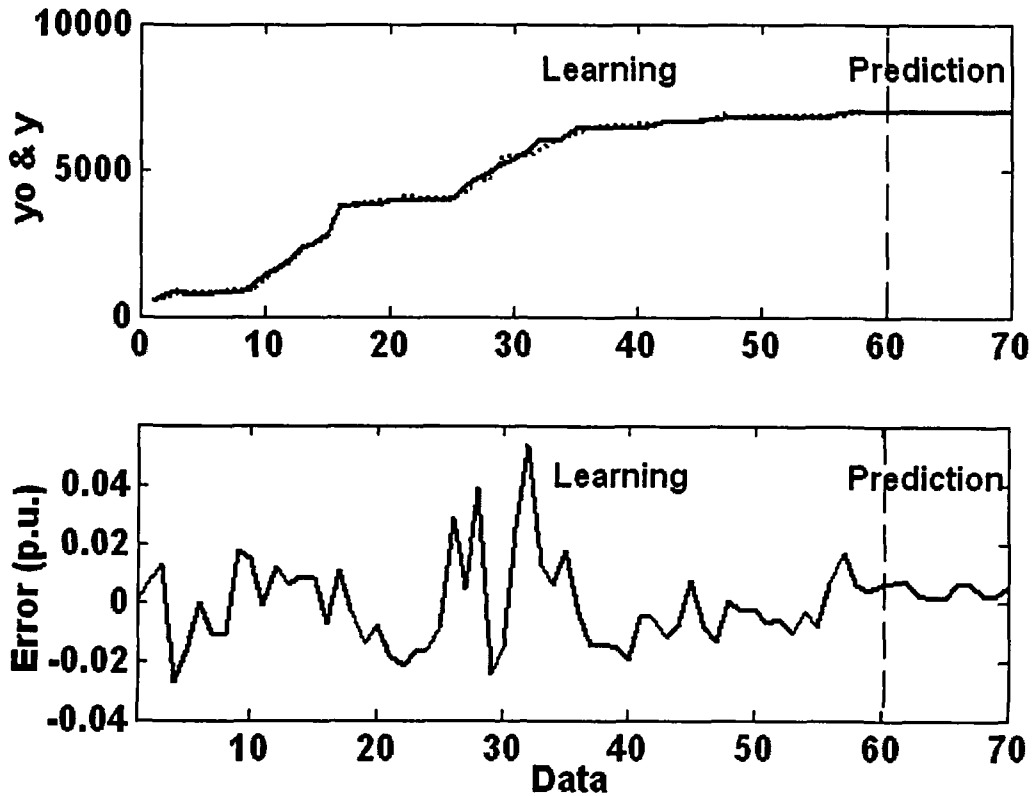


Fig. 6.28. Actual output and network output with MS-RW model and the error for Example 6

Table 6.1: Performance Index for Recurrent SS-W neuron models

Examples	S. F. ( $\alpha$ )	SS-W	SS-RW	SRS-W	SFS-W	SFW-S	SRN
Example 1	$\alpha=2$	$1.734 \times 10^{-6}$	<b><math>1.185 \times 10^{-6}</math></b>	$1.740 \times 10^{-6}$	$1.733 \times 10^{-6}$	$1.676 \times 10^{-6}$	$1.838 \times 10^{-6}$
Example 2	$\alpha=3$	$1.648 \times 10^{-5}$	<b><math>1.302 \times 10^{-5}</math></b>	$1.626 \times 10^{-5}$	$1.576 \times 10^{-5}$	$1.400 \times 10^{-5}$	$1.487 \times 10^{-5}$
Example 3	$\alpha=2$	$1.283 \times 10^{-4}$	<b><math>6.780 \times 10^{-5}</math></b>	$1.261 \times 10^{-4}$	$1.225 \times 10^{-4}$	$9.689 \times 10^{-5}$	$7.770 \times 10^{-5}$
Example 4	$\alpha=2$	$5.814 \times 10^{-6}$	$1.515 \times 10^{-6}$	$2.492 \times 10^{-6}$	$1.985 \times 10^{-6}$	$1.738 \times 10^{-6}$	<b><math>9.824 \times 10^{-7}</math></b>
Example 5	$\alpha=3$	$1.676 \times 10^{-7}$	<b><math>1.405 \times 10^{-7}</math></b>	$1.576 \times 10^{-7}$	$1.571 \times 10^{-7}$	$1.413 \times 10^{-7}$	$1.668 \times 10^{-7}$
Example 6	$\alpha=3$	$8.674 \times 10^{-6}$	<b><math>7.832 \times 10^{-6}</math></b>	$7.980 \times 10^{-6}$	$8.165 \times 10^{-6}$	$8.765 \times 10^{-6}$	$7.918 \times 10^{-6}$

Table 6.2: Performance Index for Recurrent MS-W neuron models

Examples	S. F. ( $a$ )	MS-W	MS-RW	MRS-W	MFS-W	MFW-S	MRN
Example 1	$a=2$	$7.585 \times 10^{-7}$	$6.112 \times 10^{-7}$	$6.150 \times 10^{-7}$	$6.510 \times 10^{-7}$	<b><math>4.834 \times 10^{-7}</math></b>	$6.102 \times 10^{-7}$
Example 2	$a=3$	$1.145 \times 10^{-5}$	$1.080 \times 10^{-5}$	$1.151 \times 10^{-5}$	$1.145 \times 10^{-5}$	<b><math>1.034 \times 10^{-5}</math></b>	$1.086 \times 10^{-5}$
Example 3	$a=2$	$1.361 \times 10^{-4}$	$9.559 \times 10^{-5}$	$1.356 \times 10^{-4}$	$1.351 \times 10^{-4}$	<b><math>9.053 \times 10^{-5}</math></b>	$9.405 \times 10^{-5}$
Example 4	$a=2$	$6.394 \times 10^{-6}$	$1.428 \times 10^{-6}$	$2.177 \times 10^{-6}$	$1.840 \times 10^{-6}$	<b><math>1.320 \times 10^{-6}</math></b>	$1.492 \times 10^{-6}$
Example 5	$a=2$	$1.685 \times 10^{-7}$	$1.296 \times 10^{-7}$	$1.560 \times 10^{-7}$	$1.579 \times 10^{-7}$	<b><math>9.738 \times 10^{-8}</math></b>	$1.653 \times 10^{-7}$
Example 6	$a=3$	$7.479 \times 10^{-6}$	<b><math>6.190 \times 10^{-6}</math></b>	$7.041 \times 10^{-6}$	$7.344 \times 10^{-6}$	$7.396 \times 10^{-6}$	$6.895 \times 10^{-6}$

Tables 6.1 & 6.2, show the performance index of different recurrent neurons as well as SS-W & MS-W neuron models, respectively. Scaling Factor (S.F.),  $a$ , which calculated in chapter 3, is shown in second column. In these two tables, the performance index of best model, for each example is Bold.

## 6-6 Conclusions

In this chapter, based on proposed SS-W and MS-W neuron models in chapter2, five recurrent neuron models namely SS-RW, SRS-W, SFW-S, SFS-W, SR-N and MS-RW, MRS-W, MFW-S, MFS-W, MR-N for SS-W and MS-W neuron models, respectively, are proposed. In proposed recurrent neuron models, the systems' dynamic is saved in sigmoid or wavelet activation function. Therefore, they predict the system dynamics well.

One important point should be noted that when the wavelet activation function used as memory element these recurrent neuron models in the feed-forward network yield better result. It means that sigmoid activation function do not have the same capability as of wavelet activation

function to save the systems' dynamics. Therefore, the dynamic of the system can be predicted well by recurrent model with feedback to wavelet activation function.

The results show that if the dynamic is accumulated in wavelet section of the recurrent models, namely SS-RW, SR-N, SFW-S or MS-RW, MR-N, MFW-S, in SS-W or MS-W neuron models, respectively, the performance will be better than in sigmoid function, i.e., SRS-W , SFS-W or MRS-W , MFS-W, in SS-W or MS-W, respectively.

Among the recurrent neurons model, in which feedback is to wavelet activation function, in recurrent SS-W neuron models, SS-RW model has best performance index and in recurrent MS-W neuron models, MFW-S model has best performance index.

# Chapter 7

## **CASE STUDY: Indian Summer Monsoon Rainfall (June-September)**

### **7.1 Introduction**

The agricultural economy of India is closely linked to the performance of summer monsoon rainfall all over India. The ability to understand and predict circulation and rainfall during the Asian summer monsoon on various time-scales is of prime importance to the economy of several nations of this region because of its affect on agriculture, drinking water, transportation, health, power, and the very livelihood of billions of people living in the monsoon region. To mitigate this and also with increasing population, effective planning and management of water resources is necessary [Pal'99, Singhrattna1'04].

Indian summer monsoon is one of the major components of the tropical circulation and its simulation using numerical models is one of the most challenging aspects because of its complex interactions between orography, convection and surface processes [Kang'05, Pal'99].

The major drought of 2002 [Gadgil'05, Gadgil'02, Kalsi'04, Sikka'03], with the all-India Summer Monsoon Rainfall (ISMR) (June–September) being 19% less than the long-

term average, led to considerable concern in the meteorological community since none of the predictions had suggested a large deficit in the ISMR. This was irrespective of the predictions, whether it were based on empirical models used in the country for generating operational/experimental forecasts, or generated in the different centers in the world using the atmospheric general circulation models. Fortunately, the unanticipated failure of the Indian monsoon in the summer of 2002 was followed by the summer monsoon of 2003 for which the ISMR was 2% more than the average [Rajeevan'04]. However, the relief was short-lived since the summer monsoon of 2004 has again been a drought (defined as a summer monsoon season for which the deficit in ISMR is larger than 10% of the long-term average), with the ISMR being 87% of the average. As in 2002, neither the forecast of the India Meteorological Department (IMD) for the ISMR nor the predictions from the international centers using atmospheric General Circulation Models (GCM), suggested that there would be a drought. Clearly, it is far more important to generate accurate predictions of droughts/excess rainfall seasons than of fluctuations within 10% of the average.

Different forecasting methods were suggested by researchers [Pal'99, Gowariker'89, Krishnamurti'98] and also predictability of monsoon were considered by some researchers [Kang'05, Palmer'94]. Here our attempt is the prediction of Indian summer monsoon rainfall, between June to September, by using previous rainfall data. The data used in this chapter is available in the website of Indian Institute of Tropical Meteorology department [IITM].

The chapter is organized as follow: In section 7.2 forecasting ability of the Indian summer rainfall data is considered. Application of the proposed networks and neuro-fuzzy models in forecasting of rainfall data are presented in section 7.3 and finally conclusions are derived in section 7.4.

## 7.2 Forecasting Ability of Rainfall Data

One of the main question in time series analysis is the ability of the time series data (here rainfall data) to predict future change in rainfall data. There are several unknown things, which have effect on rainfall. During last decades, several methods are developed to analyze, predict ability of the data in time series, same as Rescaled Range Analysis [Hurst'51], Correlation Dimension Estimate [Brock'92, Isham'93] and Largest Lyapunov Exponents [Benettin'80, Oseledec'68].

Rescaled Range (RR) analysis is a robust statistical method used to evaluate the degree of the presence of noise in a process, which has capability to identify a random series from a non-random one. It can also be used to determine the average length of non-periodic cycles. The computation procedure for Rescaled Range is as follows [Hurst'51, Khaloozadeh'04]:

$$R_N = \text{Max}[X_{t,N}] - \text{Min}[X_{t,N}] ; 1 \leq t \leq N \quad (7.1)$$

$$X_{t,N} = \sum_t^N (X_t - m) \quad (7.2)$$

where  $m$  is the mean value of original time series ( $X_t$ ).  $N$  is the number of observations and  $R$  captures the maximum and minimum cumulative deviations of the observations function of the number of observations  $X_t$  of the time series from its mean  $m$ , and it is a function of  $N$ ,  $R$  is defined using the following relation:

$$\frac{R_N}{S} = N^H \quad (7.3)$$

Where  $S$ , the standard deviation of  $X_t$ , and  $0 \leq H \leq 1$  are given by (7.4) and (7.5), respectively.

$$S = \left[ \sum_{i=1}^N (X_i - m)^2 / N - 1 \right]^{1/2} \quad (7.4)$$

$$H = \frac{\log(R_N/S)}{\log(N)} \quad (7.5)$$

$H$  is known as the Hurst exponent, which shows the similarity between two successive events. An estimate of  $H$  can be obtained by calculating the slope of  $\log(R_N/S)$  versus  $\log(N)$ . The largest value for  $H$  shows the mean orbital period of the process.

The value of the Hurst exponent for rainfall data is  $H=0.6436$ , which denotes long-memory effect in time series. Validity of the Hurst exponent is tested by randomly interchanging the order of data points in the original time series and calculating the Hurst exponent for a new series. In fact, for the long memory effect the order of data is important so that a new series should have a lower  $H$  estimate. The average  $H$  estimate obtained is much lower than the original series (0.4815).

Various  $\log(R_N/S)$  values are regressed on their corresponding  $\log(N)$  values and the resulting slope is the estimate for  $H$ . Fig. 7.1 shows the  $H$  estimate for original series and shuffling series versus  $N$ , respectively.

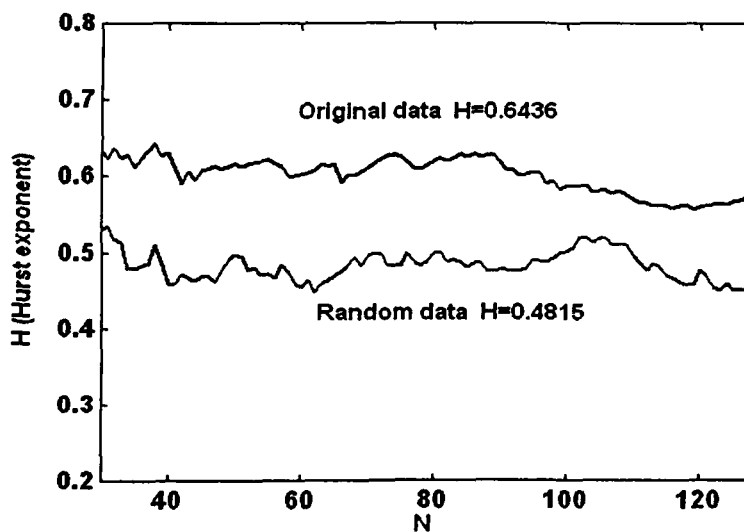


Fig. 7.1.  $H$  estimate for the original and random rainfall data



### 7.3 Simulation Results

The monthly rainfall data from 1871 to 2003 for period June to September has been taken and their average is considered in this section. There is 133 available rainfall data. Here we predict the present data,  $x(t)$ , by using five past inputs,  $x(t-1)$ ,  $x(t-2)$ ,  $x(t-3)$ ,  $x(t-4)$  &  $x(t-5)$ . Therefore, a time series with 128 data is available. The models \ networks are trained with 100 data and tested by remaining 28 data. In this section, the best-proposed networks in earlier chapters are applied to predict average of Indian summer monsoon rainfall data

#### Wavelet Neural Network (WNN):

Figures 7.2, shows the performance index of WNN (WANN in chapter 2) with different scaling factor ' $a$ '. The structure determination of the networks are started from ' $a=1$ '. We increase scaling factor one by one. By increasing scaling factor from 6 to 7 in WNN, the performance index does not improve. So scaling factors ' $a=6$ ' is selected for this network. The performance index for this model, with scaling factor  $a=6$ , is  $J=4.8031 \times 10^{-4}$ .

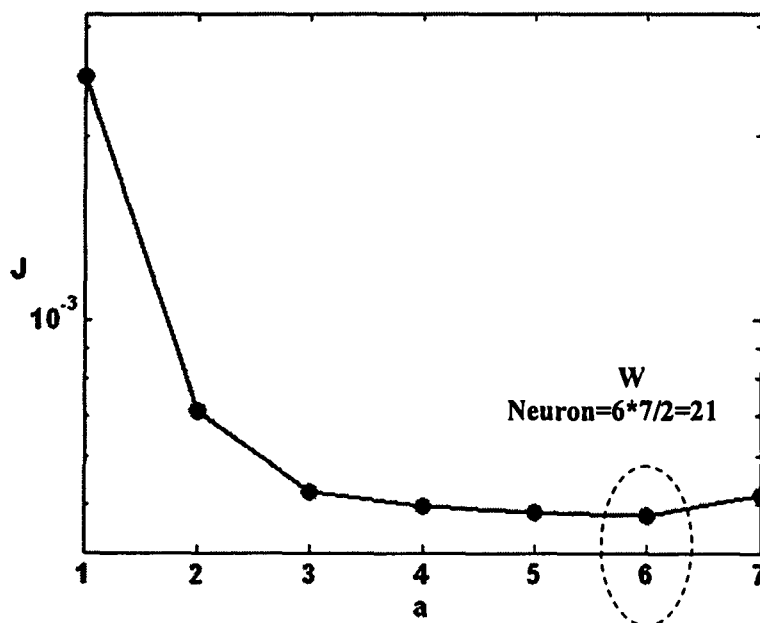


Fig. 7.2. Performance index of Wavelet Neural Network (WNN) with different scaling factor  $a$

### Summation & Multiplication Sigmoid-Wavelet (SS\_W & MS-W) Neuron networks:

Figures 7.3 and 7.4 show the performance index of feed-forward SS-W and MS-W neuron networks, respectively, with different scaling factor 'a'. The structure determination of the networks are started from 'a=1'. We increase scaling factor one by one. For every scaling factor, the model is being learned. By increasing scaling factor from 3 to 4 and from 4 to 5 for SS-W and MS-W neuron models, respectively, the performance index does not improve. Therefore, scaling factors "a=3" and "a=4" are selected for SS-W and MS-W neuron networks, respectively. The performance indexes of SS-W and MS-W neuron networks are equal to  $J=4.7597 \times 10^{-4}$  and  $J=4.7981 \times 10^{-4}$ , respectively.

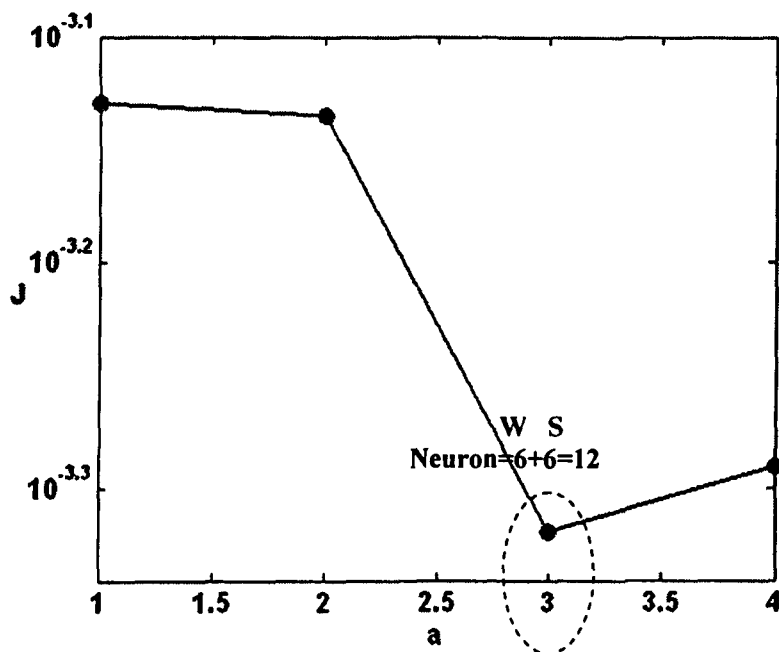


Fig. 7.3. Performance index of feed-forward SS-W neuron model with different scaling factor  $a$

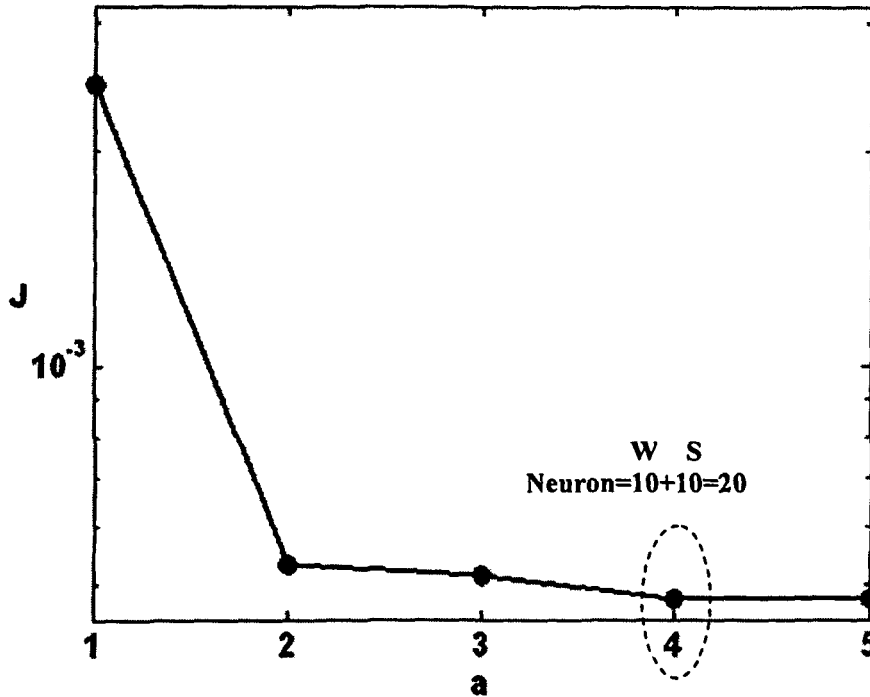


Fig. 7.4. Performance index of feed-forward MS-W neuron model with different scaling factor  $a$

**Wavelet Neuro-Fuzzy (WNF) model:**

By applying modified clustering and cluster validity function [Azeem'03a, Xie'87], five rules are obtained. The performance index of different networks for rainfall data has been listed in Table 7.1. WNF model yields better performance. In this model, we have applied genetic algorithm with 100 populations. We have fed the initial parameters for GA randomly. Figure 7.5 shows the maximum fitness of to each generation. The initial solution for GD is obtained over 30 generation. The value of performance index  $J$ , obtained by GA for initialization of the parameter, is  $8.0885 \times 10^{-4}$ .

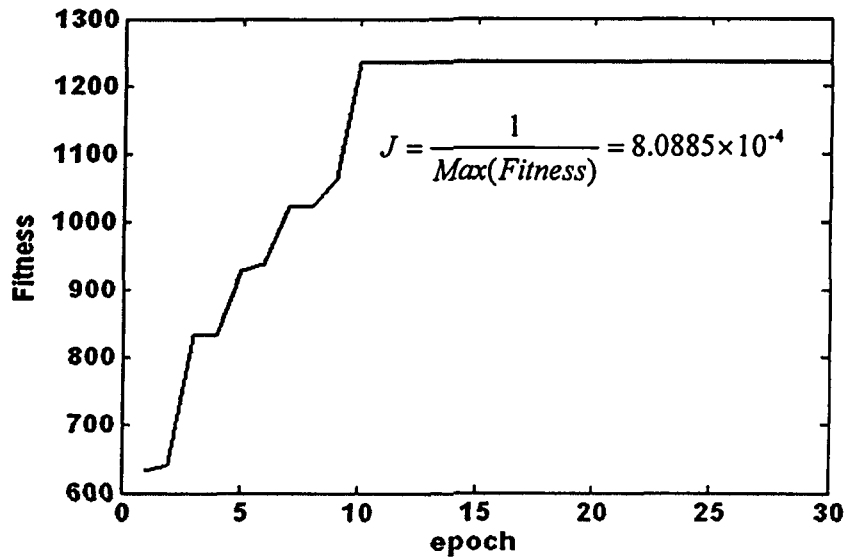


Fig. 7.5. Maximum fitness of GA up to each generation for rainfall data

Figure 7.6 shows learning pattern of WNF model, after initialization by Genetic Algorithm, with Gradient Descent. The performance index after training with 4000 epochs is  $J=1.5697 \times 10^{-4}$ .

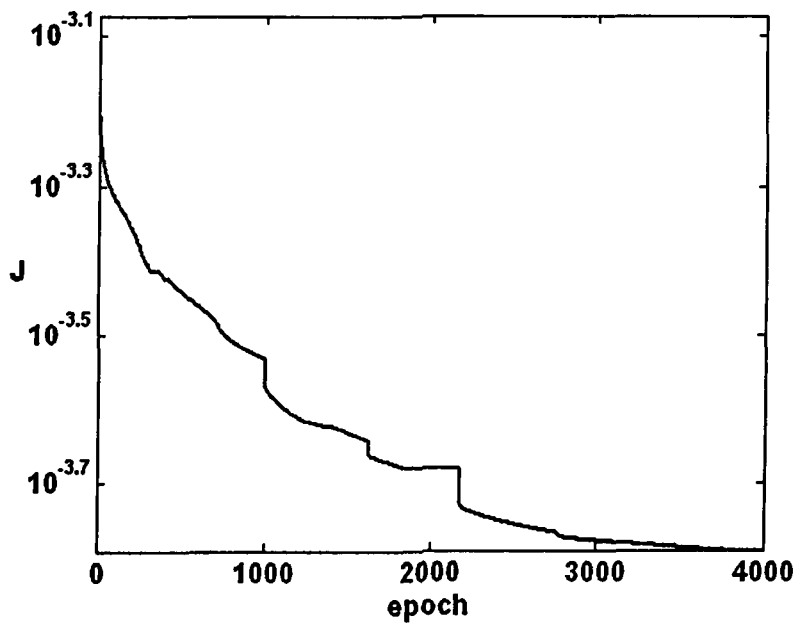


Fig. 7.6. Learning pattern of WNF model by Gradient Descent for rainfall data

The learned premise variable membership functions  $A_1^{1f}$ - $A_1^{5f}$ ,  $A_2^{1f}$ - $A_2^{5f}$ ,  $A_3^{1f}$ - $A_3^{5f}$ ,  $A_3^{1f}$ - $A_3^{5f}$  and  $A_3^{1f}$ - $A_3^{5f}$  for inputs  $x(t-1)$ ,  $x(t-2)$ ,  $x(t-3)$ ,  $x(t-4)$  &  $x(t-5)$  are shown in Fig. 7.7 to 7.11, respectively. The fuzzy rules corresponding to the learned WNF are listed below:

$$R^{1f} : \text{if } x(t-1) \text{ is } A_1^{1f} \wedge x(t-2) \text{ is } A_2^{1f} \wedge x(t-3) \text{ is } A_3^{1f} \wedge x(t-4) \text{ is } A_4^{1f} \\ \wedge x(t-5) \text{ is } A_5^{1f} \quad \text{then } x(t) \text{ is } \hat{Y}_{WNN}^{1f}(X)$$

$$R^{2f} : \text{if } x(t-1) \text{ is } A_1^{2f} \wedge x(t-2) \text{ is } A_2^{2f} \wedge x(t-3) \text{ is } A_3^{2f} \wedge x(t-4) \text{ is } A_4^{2f} \\ \wedge x(t-5) \text{ is } A_5^{2f} \quad \text{then } x(t) \text{ is } \hat{Y}_{WNN}^{2f}(X)$$

$$R^{3f} : \text{if } x(t-1) \text{ is } A_1^{3f} \wedge x(t-2) \text{ is } A_2^{3f} \wedge x(t-3) \text{ is } A_3^{3f} \wedge x(t-4) \text{ is } A_4^{3f} \\ \wedge x(t-5) \text{ is } A_5^{3f} \quad \text{then } x(t) \text{ is } \hat{Y}_{WNN}^{3f}(X)$$

$$R^{4f} : \text{if } x(t-1) \text{ is } A_1^{4f} \wedge x(t-2) \text{ is } A_2^{4f} \wedge x(t-3) \text{ is } A_3^{4f} \wedge x(t-4) \text{ is } A_4^{4f} \\ \wedge x(t-5) \text{ is } A_5^{4f} \quad \text{then } x(t) \text{ is } \hat{Y}_{WNN}^{4f}(X)$$

$$R^{5f} : \text{if } x(t-1) \text{ is } A_1^{5f} \wedge x(t-2) \text{ is } A_2^{5f} \wedge x(t-3) \text{ is } A_3^{5f} \wedge x(t-4) \text{ is } A_4^{5f} \\ \wedge x(t-5) \text{ is } A_5^{5f} \quad \text{then } x(t) \text{ is } \hat{Y}_{WNN}^{5f}(X)$$

where  $\hat{Y}_{WNN}^{1f}(X)$ ,  $\hat{Y}_{WNN}^{2f}(X)$ , ...,  $\hat{Y}_{WNN}^{5f}(X)$  are the outputs of learned MS-W neuron models in consequent parts of  $R^1$  to  $R^5$ , respectively. The learned parameters  $C_s$ ,  $C_w$  and  $W^1, \dots, W^5$ , for  $R^1$  to  $R^5$ , are as follow.

$$C_s = \begin{bmatrix} 0.767 & 0.886 & 0.050 & 0.736 & 0.255 \\ 0.142 & 0.298 & 0.772 & 0.552 & 0.209 \\ 0.830 & 0.010 & 0.580 & 0.444 & 0.913 \\ 0.528 & 0.478 & 0.301 & 0.472 & 0.603 \\ 0.570 & 0.542 & 0.645 & 0.151 & 0.236 \\ 0.987 & 0.202 & 0.641 & 0.073 & 0.508 \\ 0.487 & 0.853 & 0.526 & 0.532 & 0.808 \\ 0.090 & 0.202 & 0.395 & 0.459 & 0.664 \\ 0.114 & 0.265 & 0.420 & 0.347 & 0.876 \\ 0.557 & 0.515 & 0.903 & 0.767 & 0.216 \end{bmatrix}$$

$$C_w = \begin{bmatrix} 0.629 & 0.682 & 0.659 & 0.226 & 0.542 \\ 0.332 & 0.721 & 0.500 & 0.826 & 0.558 \\ 0.657 & 0.334 & 0.318 & 0.809 & 0.792 \\ 0.295 & 0.112 & 0.791 & 0.871 & 0.994 \\ 0.756 & 0.657 & 0.881 & 0.421 & 0.864 \\ 0.708 & 1.018 & 1.052 & 0.926 & 0.848 \\ 0.144 & 0.706 & 0.505 & 0.108 & 0.737 \\ 1.015 & 0.669 & 0.790 & 0.817 & 0.869 \\ 0.459 & 0.118 & 0.024 & 0.381 & 0.379 \\ 0.008 & 0.752 & 0.601 & 0.654 & 0.204 \end{bmatrix}$$

$$W = \begin{bmatrix} W^1 \\ W^2 \\ W^3 \\ W^4 \\ W^5 \end{bmatrix} = \begin{bmatrix} 0.556 & 0.351 & 0.050 & 0.067 & 0.833 & 0.581 & 0.530 & 0.882 & 0.608 & 0.293 \\ 0.581 & 0.593 & 0.357 & 0.528 & 0.355 & 0.500 & 0.602 & 0.868 & 0.790 & 0.569 \\ 0.227 & 0.820 & 0.025 & 0.878 & 0.013 & 0.651 & 0.648 & 0.819 & 0.679 & 0.598 \\ 0.618 & 0.822 & 0.060 & 0.713 & 0.979 & 0.358 & 0.133 & 0.875 & 0.938 & 0.441 \\ 0.538 & 0.818 & 0.216 & 0.367 & 0.523 & 0.961 & 0.131 & 0.445 & 0.395 & 0.396 \end{bmatrix}$$

Each hidden neuron in MS-W neuron model is conjunction of sigmoid and wavelet function. Rows and column in  $C_S$  and  $C_W$  are corresponding to the number of hidden neurons in MS-W neuron model and the number of inputs, respectively. The number of rows in  $W$  is equal to the number of rules whereas number of column is equal to the number of hidden neuron in MS-W neuron model.

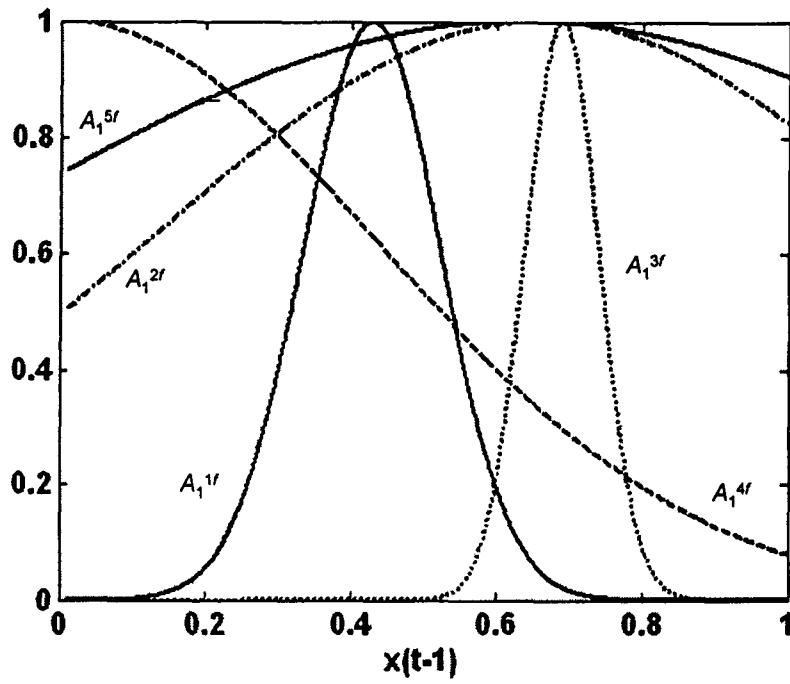


Fig. 7.7. Learned membership function, obtained by GA & GD, of the normalized input  $x(t-1)$  for rainfall data

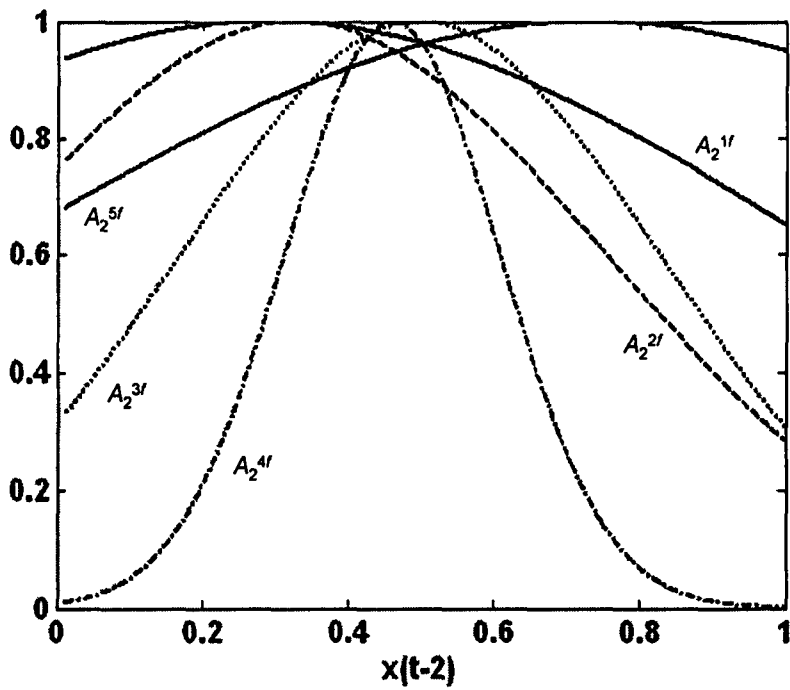


Fig. 7.8. Learned membership function, obtained by GA & GD, of the normalized input  $x(t-2)$  for rainfall data

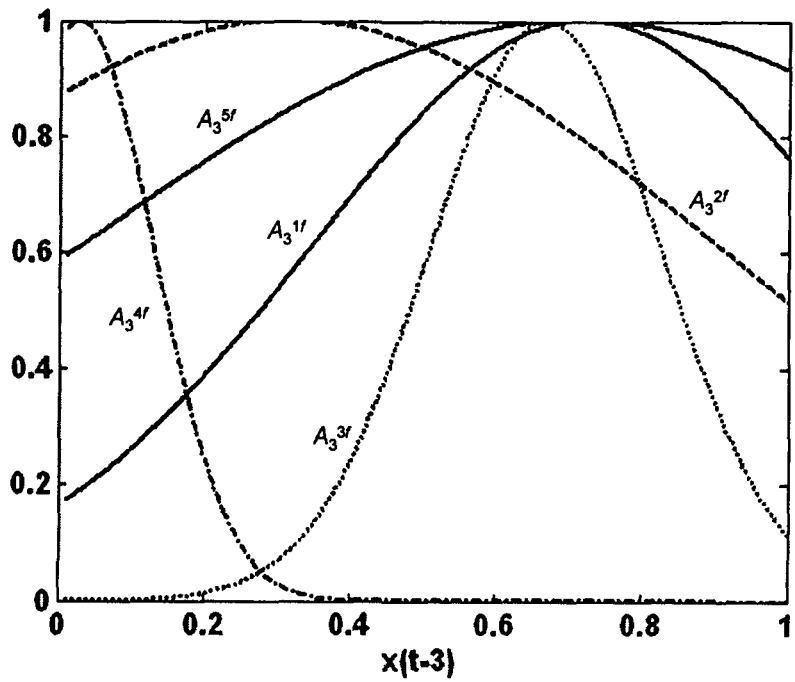


Fig. 7.9. Learned membership function, obtained by GA & GD, of the normalized input  $x(t-3)$  for rainfall data

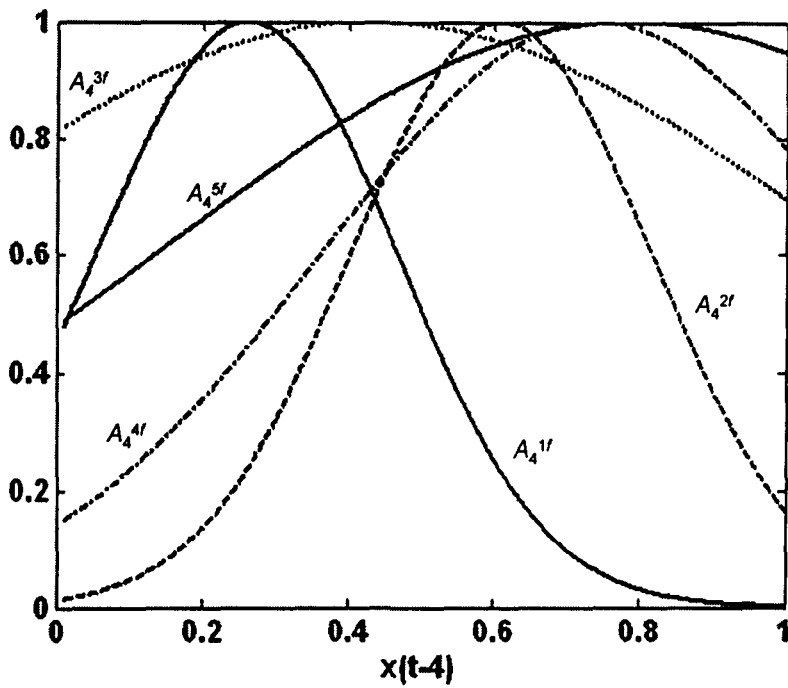


Fig. 7.10. Learned membership function, obtained by GA & GD, of the normalized input  $x(t-4)$  for rainfall data



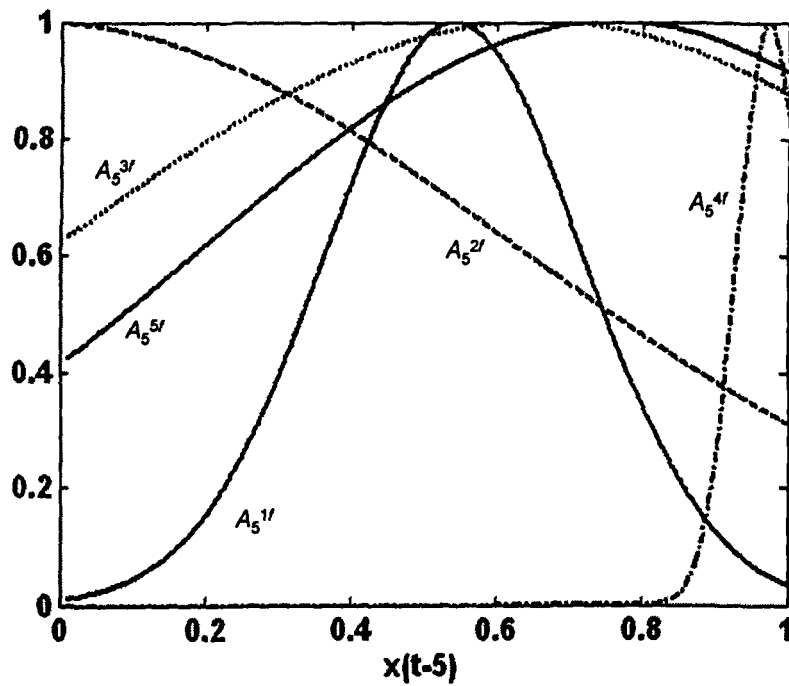


Fig. 7.11. Learned membership function, obtained by GA & GD, of the normalized input  $x(t-5)$  for rainfall data

Figure 7.12 shows the actual rainfall data and WNF model output and the model error. In this figure, actual output of the plant is solid line and the model output is dot line. The error also is solid line. The horizontal-dash line shows the 13 percent of the error.

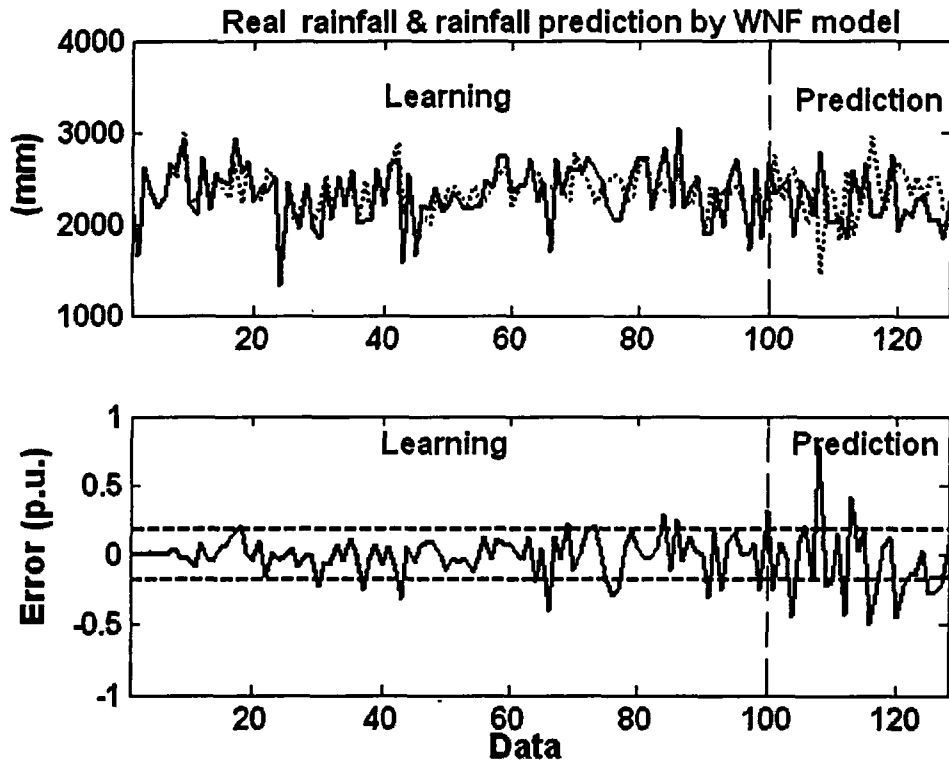


Fig. 7.12. Actual output and model output with WNF model and the error for Indian monsoon rainfall data

Table 7-1: Performance Index ( $J$ ) with different networks for rainfall data

Model	Number of Hidden Neuron	Performance Index ( $J$ ) (Training)	Performance Index ( $J$ ) (Prediction)
NN	25	$1.0523 \times 10^{-3}$	$4.2587 \times 10^{-3}$
WNN	21 (a=6)	$4.8031 \times 10^{-4}$	$9.0682 \times 10^{-4}$
SS-W	12 (a=3)	$4.7981 \times 10^{-4}$	$2.8863 \times 10^{-4}$
MS-W	20 (a=4)	$4.7597 \times 10^{-4}$	$6.2543 \times 10^{-4}$
SS-RW	12 (a=3)	$3.0719 \times 10^{-4}$	$6.1013 \times 10^{-4}$
MFW-S	20 (a=4)	$3.1463 \times 10^{-4}$	$4.1925 \times 10^{-4}$
NF	M=5	$2.3469 \times 10^{-4}$	$5.700 \times 10^{-4}$
WNF	M=5 a=4	$1.5697 \times 10^{-4}$	$7.6275 \times 10^{-4}$

Table 7.1 shows the performance index of different model / network for rainfall data. First column in this table is the name of the models. The second layer presents the number of hidden layer and rules in the networks and fuzzy models, respectively. For wavelet networks, the number of scaling factor ( $\alpha$ ) also is shown. Third and fourth columns, are performance index of rainfall data for training and prediction section. The best result in both columns is Bold.

## 7.4 Conclusions

In this chapter, we have considered Indian monsoon rainfall data. In the short term, this requires a good idea of the upcoming monsoon season rainfall, i.e. good seasonal forecast. In the long term, it needs realistic projections of scenarios of future variability and change.

Ability of rainfall data has been checked by rescaled range analysis. By using last five years rainfalls as inputs and the present data as an output, we have applied the proposed network / models to predict present data. The results show that Wavelet Neuro-Fuzzy (WNF) model yields better performance than others do. However, the Multiplication Feedback to Wavelet from Sigmoid (MFW-S) has better performance with testing data (for prediction).



# Chapter 8

## Conclusions & Future work

### 8-1 Conclusions

In this presented work, two types of WNN networks (i.e., SS-W and MS-W neuron model) are introduced. The application of proposed neuron models in recurrent network and neuro-fuzzy models also considered. Three wavelet functions namely Mexican hat, Morlet and Sinc wavelet function are tested in present wavelet functions and also proposed wavelet neuron models network.

SS-W and MS-W neuron models are single hidden layer networks. Each neuron in the hidden layer comprised of WAF and SAF. When the summation operator is used to combine them, it results in Summation Sigmoid-Wavelet (SS-W) neuron network, whereas the product operator results in Multiplication Sigmoid-Wavelet (MS-W) neuron network.

Three types of WAF, namely Mexican, Morlet and Sinc are tested in the S-W model. The comparative result of different wavelets shows that Morlet activation function yields better performance in either SS-W or MS-W neuron models.

The proposed SS-W or MS-W neuron models, have better performance than WNN network with WAF only and NN with SAF only, even with fewer number of hidden layer neurons. MS-W neuron model yields better performance in comparison to SS-W neuron model.

Based on MS-W neuron model that yields better result than SS-W neuron model, WNF model is proposed. The consequent parts of each rule in WNF model is localized by MS-W neuron model.

Two new configurations, namely CS-P and PS-P, for parameter identification of TSK neuro-fuzzy model have been compared with two existing configurations, namely P and S-P. The comparative studies have been performed on five different examples. The result shows that CS-P configuration yields best results, for all the examples, among all the four configurations. It is because of the local models of TSK neuro-fuzzy model, obtained from CS-P configuration, catering the actual dynamics of the system in the space that covers TSK neuro-fuzzy model dynamics.

GA initializes the proposed WNF model and learning parameter of the proposed model is learned by CS-P configuration, which has better performance. Result shows that MWNF has better performance than MS-W neuron model. The propose MWNF model also yields better result than TSK neuro-fuzzy model.

In this thesis, based on SS-W and MS-W neuron models five recurrent neuron models, namely S-RW, RS-W, FW-S, FS-W and R-N, have been proposed. These proposed neuron model are used in the hidden layer of a standard one hidden layered feed-forward network. Their performance is evaluated by modeling of dynamic system. In proposed recurrent neuron model, the systems' dynamic is saved in sigmoid or wavelet activation functions. Therefore, they predict the system dynamics well. The results show that if the dynamic accumulated in wavelet section of the recurrent models, namely FW-S, S-RW and R-N, the performance are better than in sigmoid function, i.e., RS-W and FS-W. Among the recurrent neurons model, in which feedback is to wavelet activation function, FW-S model has best performance index.

In this thesis, application of proposed network \ models in forecasting of Indian summer monsoon is considered. WNF model has better performance in training whereas the MFW-S recurrent neuron yields better performance in forecasting of the rainfall data.

## **8-2 Future work**

In the present work, we supposed that the wavelet parameters scaling factor and shifting are fixed. It is research topic to work on adaptive wavelet network, which wavelet parameters and weights are tuning.

In the wavelet neuro-fuzzy models, the procedure of the learning is already sleepy. This is because of initialization of the consequent part is based on the GA. If the GA method does not convergence to a good solution, it will take huge time. Therefore, an initialization method based on the wavelet parameters and fuzzy models is necessary.

Forecasting of rainfall data not only depends on last behavior of data, it also depends of another parameters same as pressure, humidity, etc. It is a complete work if we used those data also in prediction of rainfall. In this thesis, we have used previous rainfall data to predict future rainfall. For an accurate long-range prediction, Multi-step forecasting can be extended to this work. It is an ideal work if it predicts monthly rainfall and then applies this local's model to predict all data. In this work, we have selected the five last data to predict the present rainfall data. A methodology of input selection can be applied to find out best-input candidate.

# Appendix A

## Universal Approximation Theory

*Definition A-1:* A sequence of functions  $\{f_n\}$ ,  $n = 1, 2, 3, \dots$ , convergence uniformly on  $K$  to a function  $f$  if for every  $\varepsilon > 0$  there is an integer  $N$  such that  $n \geq N$  implies  $|f_n(x) - f(x)| \leq \varepsilon$  for all  $x \in K$ .

In order to prove universal approximation of proposed neuron models the Stone-Weierstrass Theorem is applied.

*Universal Approximation Theorem: (Stone-Weierstrass Theorem)*

Let  $K$  be a set of real continuous functions on a compact set  $K$ . The uniform closure of  $K$  consists of all real continuous function on  $K$ , if:

- (a)  $K$  is an algebra
- (b)  $K$  separates point on  $K$
- (c)  $K$  vanishes at no point of  $K$

*Definition A-2:* A real functions family  $\mathcal{A}$  defined on a set  $K$  is an algebra if: (i)  $f + g \in \mathcal{A}$  (ii)  $f g \in \mathcal{A}$  and (iii)  $cf \in \mathcal{A}$  are satisfied, where  $f \in \mathcal{A}$ ,  $g \in \mathcal{A}$ , and  $c$  is a complex constant. i.e.,  $\mathcal{A}$  is closed under addition, multiplication, and scalar multiplication. For example, the set of all polynomials is an algebra.

*Definition A-3:* A family  $K$  is uniformly closed if  $f \in K$  whenever  $f_n \in K$ ,  $n = 1, 2, \dots$  and  $f_n \rightarrow f$  uniformly on  $K$ .

*Definition A-4:* The uniform closure of  $K$ , denoted by  $B$  is the set of all functions that are limits of uniformly convergent sequences of members of  $K$ . By the Stone-



Weierstrass theorem, it is known that the set of continuous function on  $[a, b]$  is the uniform closure of the set of polynomials on  $[a, b]$ .

*Definition A-5:*  $K$  separates points on a set  $K$  if for every  $x, y$  in  $K, x \neq y$ , there is a function  $f$  in  $K$  such that  $f(x) \neq f(y)$ .

*Definition A-6:*  $K$  vanishes at no point of  $K$  if for each  $x$  in  $K$ , there is a function  $f$  in  $A$  such that  $f(x) \neq 0$ .

By applying conditions (a) to (c) in Stone-Weierstrass Theorem and using definitions A-2 to A-5 the universal approximation of the proposed networks is achieved.

### **A-1 Proof of Universal Approximation Theorems of SS-W and MS-W neuron models**

Suppose function in (3.5) been the output of feed-forward network. (3.6) and (3.9) evaluate the output of each neuron in SS-W and MS-W neuron model, respectively.

*Lemma A-1:* Let  $F^n$  be the family of  $\hat{Y}$  defined in (3.5). Then  $F^n \subset k$ , where  $K$  is a compact set.

*Proof:* The output of the SAF (3.7) is bounded by (A.1).

$$0 < y_j^\theta(s) = \frac{1}{1 + \exp(s)} < 1 \quad (\text{A.1})$$

where  $S = \sum_{i=1}^n C_{S_i}^j \cdot x_i$ . The outputs of the WAF (3.8) for Mexican hat, Morlet and Sinc wavelet functions are also bounded by (A.2) to (A.4), respectively.

$$\text{Suppose } Z = \sum_{i=1}^n C_{W_i}^j \cdot x_i \text{ and } G = \left( \frac{Z-b}{a} \right).$$

*Mexican hat wavelet function:* To find lower and upper bounded of the  $y_j^\psi(z)$

for Mexican hat function differential of the  $y_j^\psi(z) = e^{-G^2}(1-2G^2)$  is taken equal zero.

$$\frac{\partial}{\partial G} y_j^\psi = -2Ge^{-G^2} - 2 \left[ -2Ge^{-G^2} G^2 + 2Ge^{-G^2} \right] = 0 \Rightarrow \begin{cases} G = 0 \\ G = \pm \sqrt{\frac{3}{2}} \end{cases}$$

The upper bounded of the  $y_j^\psi(z)$  is known by  $G = 0$  and lower bounded by

$$G = \pm \sqrt{\frac{3}{2}} :$$

$$-0.4463 = e^{-G^2}(1-2G^2) < e^{-3/2} \left( 1 - 2 \cdot \frac{3}{2} \right) < y_j^\psi(z) < 1 \quad (\text{A.2})$$

*Morlet wavelet function:* Since  $0 < e^{-G^2} < 1$  and  $-1 < \cos G < 1$  therefore

$$-1 < y_j^\psi(z) = e^{-G^2} \cos G < 1 \quad (\text{A.3})$$

*Sinc wavelet function:* Since  $\lim_{G \rightarrow \infty} \frac{\sin(\pi G)}{\pi G} = 0$  and  $\lim_{G \rightarrow 0} \frac{\sin(\pi G)}{\pi G} = 1$ , therefore

this function is bounded between  $[0, 1]$ , but this function in  $G = 0$  should be stated equal 1.

$$0 < y_j^\psi(z) = \sin(\pi G)/(\pi G) < 1 \quad (\text{A.4})$$

From (A.2) to (A.4), the continuous function  $W_j \cdot (y_j^\theta(s) + y_j^\psi(z))$  or

$W_j \cdot y_j^\theta(s) \cdot y_j^\psi(z)$  in (3.5) for SS-W and MS-W neuron models, respectively,

with the Mexican hat, Morlet and Sinc wavelet functions is closed. That is,

$$F^n \subset k.$$

**Lemma A-2:** Let  $F^n$  be the family of  $\hat{Y}$  defined in (3.5). Then  $F^n$  is an algebra.

*Proof:* let  $f, g \in F^n$  as shown in (3.5). Then we can write

$$f(x) = \sum_{j=1}^{L_1} W_j^1 \cdot y_j^1 \quad (\text{A.5})$$

$$g(x) = \sum_{j=1}^{L_2} W_j^2 \cdot y_j^2 \quad (\text{A.6})$$

where  $W_j^1$  and  $W_j^2 \in \mathfrak{R}$ ,  $\forall j$ . Therefore for SS-W neuron model we have:

$$\begin{aligned} f + g &= \sum_{j=1}^{L_1} W_j^1 \cdot y_j^1 + \sum_{j=1}^{L_2} W_j^2 \cdot y_j^2 \\ &= \sum_{j=1}^{L_1+L_2} W_j \cdot y_j \end{aligned} \quad (\text{A.7})$$

Since  $W_j \in \mathfrak{R}$ , then  $f + g \in F^n$ , That is,  $F^n$  is closed under addition.

Similarly for MS-W neuron model,

$$\begin{aligned} f \cdot g &= \sum_{j=1}^{L_1} W_j^1 \cdot y_j^1 \cdot \sum_{j=1}^{L_2} W_j^2 \cdot y_j^2 \\ &= W_1^T Y_1 \cdot W_2^T Y_2 = W_1^T (Y_1 \cdot Y_2) W_2 = W_1^T ((Y_1^\theta \cdot Y_1^\psi) \cdot (Y_2^\theta \cdot Y_2^\psi)) W_2 \\ &= W_1^T ((Y_1^\theta \cdot Y_2^\theta) \cdot (Y_1^\psi \cdot Y_2^\psi)) W_2 = W_1^T (Y_1^\theta \cdot Y_2^\theta) W_2 = W_1^T Y W_2 \end{aligned} \quad (\text{A.8})$$

where  $W_1 = [w_1^1 \ \dots \ w_{L_1}^1]^T \in \mathfrak{R}^{L_1}$ ,  $W_2 = [w_1^2 \ \dots \ w_{L_2}^2]^T \in \mathfrak{R}^{L_2}$ . Since, the

product of SAF or WAF is neither also SAF nor WAF, thus  $f \cdot g \in F^n$ . That

is,  $F^n$  is closed under multiplication. Finally, for arbitrary  $c \in \mathfrak{R}$

$$c \cdot f = \sum_{j=1}^{L_1} (c \cdot W_j^1) \cdot y_j^1 = \sum_{j=1}^{L_1} W_j^c \cdot y_j^1 \quad (\text{A.9})$$

Since  $W_j^c = c \cdot W_j^1 \in \mathfrak{R}$ ,  $\forall j$  thus  $c \cdot f \in F^n$ . That is,  $F^n$  is closed under scalar multiplication.

From *Definition A-2* and above discussion, we conclude that  $F^n$  in both SS-W and MS-W neuron models, is algebra.

**Lemma A-3:**  $F^n$  separates points on  $K$ .

*Proof:* From *Definition A-3* we show that for  $\forall x^0, y^0 \in F^n$  if  $x^0 \neq y^0$ , there is a

function  $f \in F^n$  such that  $f(x^0) \neq f(y^0)$ .

$$f(x^0) = \sum_{j=1}^L W_j \cdot y_j(x^0) \quad (\text{A.10})$$

$$f(y^0) = \sum_{j=1}^L W_j \cdot y_j(y^0) \quad (\text{A.11})$$

Suppose  $Z^{x^0} = \sum_{i=1}^n C_{W_i}^j \cdot x^0_i$  &  $S^{x^0} = \sum_{i=1}^n C_{S_i}^j \cdot x^0_i$  and  $Z^{y^0} = \sum_{i=1}^n C_{W_i}^j \cdot y^0_i$  &

$S^{y^0} = \sum_{i=1}^n C_{S_i}^j \cdot y^0_i$ . From (2.11-2.13) for SS-W and MS-W neuron model with

different wavelet function we have:

*SS-W neuron model with Mexican wavelet function:*

$$y_j^\theta(x^0) + y_j^\psi(x^0) = \frac{1}{1 + e^{-S^{x^0}}} + e^{-\left(\frac{Z^{x^0} - b}{a}\right)} \cdot \left(1 - 2\left(\frac{Z^{x^0} - b}{a}\right)\right) \quad (\text{A.12})$$

$$y_j^\theta(y^0) + y_j^\psi(y^0) = \frac{1}{1 + e^{-S^{y^0}}} + e^{-\left(\frac{Z^{y^0} - b}{a}\right)} \cdot \left(1 - 2\left(\frac{Z^{y^0} - b}{a}\right)\right) \quad (\text{A.13})$$

*MS-W neuron model with Mexican wavelet function:*

$$y_j^\theta(x^0) \cdot y_j^\psi(x^0) = \frac{1}{1 + e^{-S^{x^0}}} \cdot e^{-\left(\frac{Z^{x^0} - b}{a}\right)} \cdot \left(1 - 2\left(\frac{Z^{x^0} - b}{a}\right)\right) \quad (\text{A.14})$$

$$y_j^\theta(y^0) \cdot y_j^\psi(y^0) = \frac{1}{1 + e^{-S^{y^0}}} \cdot e^{-\left(\frac{Z^{y^0} - b}{a}\right)} \cdot \left(1 - 2\left(\frac{Z^{y^0} - b}{a}\right)\right) \quad (\text{A.15})$$

Thus, from (A.12) to (A.15), it is easy to verify that for SS-W and MS-W neuron

models with *Mexican hat* wavelet function,  $f(x^0) \neq f(y^0)$  if  $x^0 \neq y^0$ .

*SS-W neuron model with Morlet wavelet function:*

$$y_j^\theta(x^0) + y_j^\psi(x^0) = \frac{1}{1 + e^{-s x^0}} + e^{-\left(\frac{z^{x^0} - b}{a}\right)} \cos\left(\frac{Z^{x^0} - b}{a}\right) \quad (\text{A.16})$$

$$y_j^\theta(y^0) + y_j^\psi(y^0) = \frac{1}{1 + e^{-s y^0}} + e^{-\left(\frac{z^{y^0} - b}{a}\right)} \cos\left(\frac{Z^{y^0} - b}{a}\right) \quad (\text{A.17})$$

*MS-W neuron model with Morlet wavelet function:*

$$y_j^\theta(x^0) \cdot y_j^\psi(x^0) = \frac{1}{1 + e^{-s x^0}} \cdot e^{-\left(\frac{z^{x^0} - b}{a}\right)} \cos\left(\frac{Z^{x^0} - b}{a}\right) \quad (\text{A.18})$$

$$y_j^\theta(y^0) \cdot y_j^\psi(y^0) = \frac{1}{1 + e^{-s y^0}} \cdot e^{-\left(\frac{z^{y^0} - b}{a}\right)} \cos\left(\frac{Z^{y^0} - b}{a}\right) \quad (\text{A.19})$$

Thus, from (A.16) to (A.19), it is easy to verify that for SS-W and MS-W neuron models with *Morlet* wavelet function  $f(x^0) \neq f(y^0)$  if  $x^0 \neq y^0$ .

*SS-W neuron model with Sinc wavelet function:*

$$y_j^\theta(x^0) + y_j^\psi(x^0) = \frac{1}{1 + e^{-s x^0}} + \frac{\sin\left(\pi\left(\frac{Z^{x^0} - b}{a}\right)\right)}{\pi\left(\frac{Z^{x^0} - b}{a}\right)} \quad (\text{A.20})$$

$$y_j^\theta(y^0) + y_j^\psi(y^0) = \frac{1}{1 + e^{-s y^0}} + \frac{\sin\left(\pi\left(\frac{Z^{y^0} - b}{a}\right)\right)}{\pi\left(\frac{Z^{y^0} - b}{a}\right)} \quad (\text{A.21})$$

*MS-W neuron model with Sinc wavelet function:*

$$y_j^\theta(x^0) \cdot y_j^\psi(x^0) = \frac{1}{1 + e^{-s x^0}} \cdot \frac{\sin\left(\pi\left(\frac{Z^{x^0} - b}{a}\right)\right)}{\pi\left(\frac{Z^{x^0} - b}{a}\right)} \quad (\text{A.22})$$

$$y_j^\theta(y^0) \cdot y_j^\psi(y^0) = \frac{1}{1 + e^{-s y^0}} \cdot \frac{\sin\left(\pi\left(\frac{Z y^0 - b}{a}\right)\right)}{\pi\left(\frac{Z y^0 - b}{a}\right)} \quad (\text{A.23})$$

Thus, from (A.20) to (A.23), it is easy to verify that for SS-W and MS-W neuron models with *Sinc* wavelet function  $f(x^0) \neq f(y^0)$  if  $x^0 \neq y^0$ .

From above discussion and *Definition A-3*,  $F^n$ , in both proposed SS-W and MS-W neuron models, separates points on  $K$ .

**Lemma 4:**  $F^n$  vanished at no point of  $K$ .

*Proof:* if we choose  $W_j \neq 0$  ( $j=1, 2, \dots, L$ ) in (3.5) then  $f(x) = \hat{Y} \neq 0$  unless in those point that  $y_i$  are equal zero. In SS-W neuron model  $y_i = y_i^\theta + y_i^\psi$ . Since  $y_i^\theta(x) \neq 0$  for all  $x$ ,  $f(x) = \hat{Y} \neq 0$ . In MS-W neuron model network  $y_i = y_i^\theta \cdot y_i^\psi$ . Since  $y_i^\theta(x) > 0$  the condition is limited to  $y_i^\psi(x) \neq 0$ . In following we consider this condition for different wavelet function.

*Mexican wavelet function:*

$$\left(1 - 2\left(\frac{Z-b}{a}\right)^2\right) \cdot e^{-\left(\frac{Z-b}{a}\right)^2} \neq 0 \Rightarrow 1 - 2\left(\frac{Z-b}{a}\right)^2 \neq 0 \quad (\text{A.24})$$

Then  $Z$  should be selected as:

$$Z \neq b \pm a \frac{\sqrt{2}}{2} \quad (\text{A.25})$$

*Morlet wavelet function:*

$$y_j^\psi(Z) = e^{-\left(\frac{Z-b}{a}\right)^2} \cos\left(\frac{Z-b}{a}\right) \neq 0 \Rightarrow \cos\left(\frac{Z-b}{a}\right) \neq 0 \quad (\text{A.26})$$

Then  $Z$  should be selected as:

$$Z \neq b + a(2k+1)\frac{\pi}{2} \quad (\text{A.27})$$

*Sinc wavelet function:*

$$\frac{\sin\left(\pi\left(\frac{Z-b}{a}\right)\right)}{\pi\left(\frac{Z-b}{a}\right)} \neq 0 \Rightarrow \sin\left(\pi\left(\frac{Z-b}{a}\right)\right) \neq 0 \quad (\text{A.28})$$

Then  $Z$  should be selected as:

$$Z \neq b + ka \quad (\text{A.29})$$

From above discussion and Stone-Weierstrass Theorem, we can easily approve the following *Universal Approximation Theorems*.

**Theorem 3.1:** *Universal approximation theorem of SS-W neuron model*, for any real function

$h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an SS-W network  $f$ , with Mexican hat, Morlet or Sinc WAF, such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

**Proof:** From *Lemma 1* to 4, it is easy to show that SS-W neuron model is universal approximation.

**Theorem 3.2:** *Universal approximation theorem of MS-W neuron model with Mexican hat*

*WAF*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an MS-W network  $f$ , with Mexican hat WAF that satisfies condition (A.30), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

$$C_w X \neq b \pm a \frac{\sqrt{2}}{2} \quad (\text{A.30})$$

where  $C_w = \{C_{w_1}, C_{w_2}, \dots, C_{w_n}\}$ ,  $X = \{x_1, x_2, \dots, x_n\}$ .

Proof: MS-W neuron model with Mexican hat is universal approximation if satisfy A.25 that is (A.30).

Theorem 3.3: *Universal approximation theorem of MS-W neuron model with Morlet WAF,*

for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an MS-W network  $f$ , with Morlet WAF that satisfies condition (A.31), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

$$C_w X \neq b + a(2\rho + 1) \frac{\pi}{2} \quad (\text{A.31})$$

where  $C_w = \{C_{w_1}, C_{w_2}, \dots, C_{w_n}\}$ ,  $X = \{x_1, x_2, \dots, x_n\}$  and  $\rho$  is any integer value.

Proof: MS-W neuron model with Morlet is universal approximation if satisfy A.27 that is (A.31).

Theorem 3.4: *Universal approximation theorem of MS-W neuron model with Sinc WAF,* for

any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an MS-W network  $f$ , with Sinc WAF that satisfies condition (A.32), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

$$C_w X \neq b + \rho a \quad (\text{A.32})$$

where  $C_w = \{C_{w_1}, C_{w_2}, \dots, C_{w_n}\}$ ,  $X = \{x_1, x_2, \dots, x_n\}$  and  $\rho$  is any integer value.

Proof: MS-W neuron model with Sinc is universal approximation if satisfy A.29 that is (A.32).



## A-2 Proof of Universal Approximation Theorems of Recurrent SS-W and MS-W neuron models with Morlet wavelet function

Theorem 6.1: *Universal approximation theorem of recurrent SS-W neuron models*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an recurrent SS-W network  $f$ , for all recurrent SS-W neuron models, such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

Proof: From *Lemma 1* to 4, it is easy to show that all recurrent SS-W neuron models are universal approximation.

Theorem 6.2: *Universal approximation theorem of MS-RW neuron model*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$ , there is an MS-RW neuron model  $f$  that satisfies condition (A.33), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

$$C_w X + Q_w y^\psi (k-1) \neq b + a(2\rho+1)\frac{\pi}{2} \quad (\text{A.33})$$

Where  $C_w = \{C_{w_1}, C_{w_2}, \dots, C_{w_n}\}$ ,  $Q_w = \{Q_{w_1}, Q_{w_2}, \dots, Q_{w_l}\}$ ,  $y^\psi = \{y_1^\psi, y_2^\psi, \dots, y_l^\psi\}$ ,  $X = \{x_1, x_2, \dots, x_n\}$  and  $\rho$  is any integer value.

Proof: The variable  $Z$  in (A.27) in MS-RW neuron model is  $Z = C_w X + Q_w y^\psi (k-1)$ .

Therefore (A.33) is draw out.

Theorem 6.3: *Universal approximation theorem of MRS-W neuron model*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an MRS-W neuron model  $f$  that satisfies condition (A.30), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

Proof: The variable  $Z$  in (A.27) in MRS-W neuron model is same as (A.27) or (A.30).

**Theorem 6.4:** *Universal approximation theorem of MFS-W neuron model*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an MFS-W neuron model  $f$  that satisfies condition (A.30), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

Proof: The variable  $Z$  in (A.27) in MFS-W neuron model is same as (A.27) or (A.30).

**Theorem 6.5:** *Universal approximation theorem of MFW-S neuron model*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an MFW-S neuron model  $f$  that satisfies condition (A.34), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

$$C_w X + Q_{sw} y^\psi (k-1) \neq b + a(2\rho+1)\frac{\pi}{2} \quad (\text{A.34})$$

where  $C_w = \{C_{w_1}, C_{w_2}, \dots, C_{w_n}\}$ ,  $Q_{sw} = \{Q_{sw_1}, Q_{sw_2}, \dots, Q_{sw_L}\}$ ,  $y^\psi = \{y_1^\psi, y_2^\psi, \dots, y_L^\psi\}$ ,

$X = \{x_1, x_2, \dots, x_n\}$  and  $\rho$  is any integer value. Then we have the following result.

Proof: The variable  $Z$  in (A.27) in MFW-S neuron model is  $Z = C_w X + Q_{sw} y^\psi (k-1)$ .

Therefore (A.34) is draw out.

**Theorem 6.6:** *Universal approximation theorem of MRN neuron model*, for any real function  $h: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$  which is continuous on a compact set  $\mathcal{A} \subset \mathfrak{R}^n$  and for any given  $\varepsilon > 0$  there is an RN neuron model  $f$  that satisfies condition (A.35), such that  $\sup_{x \in \mathcal{A}} \|f(x) - h(x)\| < \varepsilon$ . Here  $\|\cdot\|$  can be any norm.

$$C_w X + Q y(k-1) \neq b + a(2\rho+1)\frac{\pi}{2} \quad (\text{A.35})$$

Where  $C_w = \{C_{w_1}, C_{w_2}, \dots, C_{w_n}\}$ ,  $Q = \{Q_1, Q_2, \dots, Q_L\}$ ,  $y = \{y_1, y_2, \dots, y_L\}$ ,

$X = \{x_1, x_2, \dots, x_n\}$  and  $\rho$  is any integer value. Then we have the following

result.

Proof: The variable  $Z$  in (A.27) in MRN neuron model is  $Z = C_w X + Q y(k-1)$ . Therefore

(A.35) is draw out.

## Appendix B

### Modified Mountain Clustering

The purpose of clustering is to do natural grouping of large set of data, producing a concise representation of system's behavior. [Azeem'03a] have proposed a, simple and easy to implement, mountain clustering algorithm for estimating the number and location of cluster centers. The proposed modified mountain clustering in unit hypercube (normalized space of data) is as follows:

We assume that each data point has potential to become a cluster center and calculate its potential by:

$$Q_i^1 = \sum_{j=1}^n \exp\left(-\alpha \|x_i - x_j\|^2\right), \quad i = 1, 2, \dots, n \quad (\text{B.1})$$

Where  $\alpha = \frac{4}{r_a^2}$  and n is number of data.

$\| \cdot \|$  denote the Euclidian distance and  $r_a$  is a positive constant, which defines the neighborhood of datum. A data point with many neighboring data points will have a high potential value and data points outside radial distance  $r_a$  have a little influence on the potential. After the potential of every data point has been evaluated, the data with highest potential is selected as first center:

$$x_i^* \leftarrow Q_i^* = \max_{i=1}^n (Q_i^1) \quad (\text{B.2})$$

For the selection of second cluster center, the potential value of each datum is revised in order to deduce the effect of mountain function around the first cluster center as follows:

$$Q_i^2 = Q_i^1 - Q_1^* \exp\left(-\beta \|x_i - x_1^*\|^2\right) \quad (\text{B.3})$$

Where  $\beta = \frac{4}{r_b^2}$  and second cluster center will be:

$$x_2^* \Leftarrow Q_2^* = \max_{i=1}^{n-1} (Q_i^2) \quad (\text{B.4})$$

$r_b$  is a positive constant, which defines the neighborhood of cluster center. Thus, we subtract an amount of potential from each data point as a function of its distance from the first cluster center. It is evident from the above equation that the data points near the first cluster center have greatly reduced potential value and are unlikely to be selected as the next cluster center. After revision of the potential value of each datum, second cluster center is selected with highest remaining potential. Similarly, for the selection of  $k^{\text{th}}$  cluster center, revision of the potential value for each datum is done by:

$$Q_i^{k-1} = Q_i^{k-1} - Q_{k-1}^* \exp\left(-\beta \|x_i - x_{k-1}^*\|^2\right) \quad (\text{B.5})$$

Where  $x_{k-1}^*$  is location of  $(k-1)^{\text{th}}$  cluster center and  $Q_{k-1}^*$  is its potential value and the  $k^{\text{th}}$  cluster will be:

$$x_k^* \Leftarrow Q_k^* = \max_{i=1}^{n-k+1} (Q_i^k) \quad (\text{B.6})$$

To stop the procedure we use the criterion  $Q_k^*/Q_1^* < \delta$  ( $\delta$  is a small fraction).

Number of resulting cluster centers and distance between them are highly dependent upon the mountain clustering parameters, i.e., the neighborhood of datum or radius of

influence  $r_a$ , neighborhood of clusters  $r_b$ , gray region parameter  $\delta$ . A brief analysis of their choice is presented as follows:

a) Neighborhood of datum  $r_a$

Smaller the value of  $r_a$ , smaller the values of potential value of first centers and  $r_b$  which result in the large number of cluster centers and vice versa. The number of cluster centers approaches the number of data points thereby defeating the purpose of clustering. The maximum value for  $r_a$  is half of the principal diagonal of unit hypercube, i.e.,  $r_{a_{max}} = \sqrt{(n+1)}/2$  and the minimum value may be taken as  $r_{a_{min}} = 0.2r_{a_{max}}$ .

b) Neighborhood of cluster centers

The spaces among the resulting clusters are highly depending upon the value of  $r_b$ . To avoid obtaining closely spaced cluster centers, set  $r_b$  to be somewhat greater than  $r_a$ .

c) Gray region of parameters  $\delta_u$  and  $\delta_l$

The number of cluster centers is also depends upon the position and range of gray region. Small value of  $\delta$  results in a large number of cluster centers vice versa. It is difficult to establish a single value for  $\delta$  that works well for all data. A good choice of the upper and lower limits is to take  $\delta_u = 0.15$  and  $\delta_l = 0.0$ .

The number of optimum clusters for the data set  $\{x_p, y_p\}_{p=1}^P$  is decided by the validity function  $S$  which is the ratio of compactness to separation, [Xie'87]

$$S = \frac{\sum_{m=1}^M \sum_{p=1}^P \beta_{pm}^2 \|\bar{x}_{py} - \bar{c}_{my}\|^2}{P \cdot \min_{i \neq p} \|\bar{c}_{iy} - \bar{c}_{py}\|^2} ; \text{ for each } M = M_{min}, \dots, M_{max} \quad (B.7)$$

where  $\beta_{pm} = \frac{\mu_{A^m}}{\sum_{m=1}^M \mu_{A^m}}$ . The membership function  $\mu_{A^m}$  represents the degree of

association of  $p^{\text{th}}$  data to the  $m^{\text{th}}$  cluster center and is defined as:

$$\mu_{A^m} = \exp\left\{-\frac{(x_p - \bar{x}_{mp})^2}{(\sigma_{mp})^2}\right\} \quad (\text{B.8})$$

(B.8) assures the most valid fuzzy clustering of the data set.

Denoting the optimal candidate at each  $M$  by  $\Omega_M$  the solution to the following minimization problem

$$\min_{M_{\min} \leq M \leq M_{\max}} \left( \min_{\Omega_M} S \right) \quad (\text{B.9})$$

is ensured to yield the most valid fuzzy clustering of the data set.  $S$  has a tendency to decrease eventually when  $M$  is very large. So, the values of  $S$  are meaningless when  $M$  gets close to  $P$ . Since in practice the feasible number of clusters  $M$  is much smaller than the number of data points  $P$ , we apply two heuristic methods for the determination of  $M_{\max}$  for both small and large values of  $P$ .

In the first method, we plot the optimal values of  $S$  for  $M = M_{\min}$  to  $P-1$  when  $P$  takes small values, and select the starting point of monotonically decreasing tendency of  $S$  at  $M_{\max}$ .

In the second method, we need not compute  $S$  for very large  $M$  when  $P$  takes large values. It is almost always the case that  $M$  at the stop-value is  $\ll P$ . We can choose  $M_{\max}$  a priori, e.g., say  $M_{\max} = P/3$  which is not likely to reach the starting point of the decreasing tendency.

## Appendix C

### Stability and Convergence Analysis of Gradient Descent Learning Algorithm

#### C-1 Lyapunov method in analysis of stability

Consider a dynamic system, which satisfies:

$$\dot{x} = f(x, t) \quad x(t_0) = x_0 \quad x \in R \quad (C.1)$$

The equilibrium point  $x^* = 0$  is stable (in the sense of Lyapunov) at  $t = t_0$  if for any  $\varepsilon > 0$  there exists a  $\delta(t_0, \varepsilon) > 0$  such that

$$\|x(t_0)\| < \delta \quad \Rightarrow \quad \|x(t)\| < \varepsilon, \quad \forall t \geq t_0 \quad (C.2)$$

#### C-1.1 Lyapunov Stability Theorem

Let  $V(x, t)$  be a non-negative function with derivative  $\dot{V}$  along the trajectories of the system, then

- The origin of the system is locally stable (in the sense of Lyapunov) if  $V(x, t)$  is locally positive definite and  $-\dot{V}(x, t) \leq 0$  locally in  $x$  and for all  $t$ .
- The origin of the system is globally uniformly asymptotically stable if  $V(x, t)$  is positive definite and exrescent, and  $-\dot{V}(x, t)$  is positive definite.

To approve stability analysis based on GD learning algorithm we can define discreet function as:

$$V(k) = E(k) = \frac{1}{2} \cdot [e(k)]^2 \quad (C.3)$$



Change of Lyapunov function is:

$$\Delta V(k) = V(k+1) - V(k) = \frac{1}{2} \cdot [e^2(k+1) - e^2(k)] \quad (C.4)$$

From

$$e(k+1) = e(k) + \Delta e(k) \Rightarrow e^2(k+1) = e^2(k) + \Delta^2 e(k) + 2 \cdot e(k) \cdot \Delta e(k) \quad (C.5)$$

Then

$$\Delta V(k) = \Delta e(k) \cdot \left[ e(k) + \frac{1}{2} \cdot \Delta e(k) \right] \quad (C.6)$$

Difference of error is:

$$\Delta e(k) = e(k+1) - e(k) \approx \left[ \frac{\partial e(k)}{\partial v} \right]^T \cdot \Delta v \quad (C.7)$$

where  $\theta$  is learning parameter and  $e(k) = \hat{y}(k) - y(k)$  is error between output of plant and present output of network.

As discussed in section 2-5:

$$\Delta v = -\eta \cdot \frac{\partial J}{\partial v} \quad (C.8)$$

By using equations (C.7, 2.18) and putting them in (C.6):

$$\Delta V(k) = \left[ \frac{\partial e(k)}{\partial v} \right]^T \cdot \Delta v \cdot \left\{ e(k) + \frac{1}{2} \cdot \left[ \frac{\partial e(k)}{\partial v} \right]^T \cdot \Delta v \right\}$$

$$\text{Or } \Delta V(k) = \left[ \frac{\partial e(k)}{\partial v} \right]^T \cdot \left( -\eta \cdot \frac{\partial E(k)}{\partial v} \right) \cdot \left\{ e(k) + \frac{1}{2} \cdot \left[ \frac{\partial e(k)}{\partial v} \right]^T \cdot \left( -\eta \cdot \frac{\partial E(k)}{\partial v} \right) \right\}$$

$$\text{or } \Delta V(k) = \left[ \frac{\partial e(k)}{\partial v} \right]^T \cdot (-\eta) \cdot \frac{1}{P \cdot y_r^2} \cdot e(k) \cdot \frac{\partial \hat{y}(k)}{\partial v} \cdot \left\{ e(k) + \frac{1}{2} \cdot \left[ \frac{\partial e(k)}{\partial v} \right]^T \cdot (-\eta) \cdot \frac{1}{P \cdot y_r^2} \cdot e(k) \cdot \frac{\partial \hat{y}(k)}{\partial v} \right\}$$

$$\text{or } \Delta V(k) = e^2(k) \cdot \left\{ - \left[ \frac{\partial \hat{y}(k)}{\partial v} \right]^T \cdot \eta \cdot \frac{1}{P \cdot y_r^2} \cdot \frac{\partial \hat{y}(k)}{\partial v} + \frac{1}{2} \cdot \left[ \frac{\partial \hat{y}(k)}{\partial v} \right]^T \cdot \left[ \frac{\partial \hat{y}(k)}{\partial v} \right]^T \cdot \eta^2 \cdot \frac{1}{(P \cdot y_r^2)^2} \cdot \left( \frac{\partial \hat{y}(k)}{\partial v} \right)^2 \right\}$$

$$\Delta V(k) = -e^2(k) \cdot \frac{1}{2} \cdot \frac{\eta}{P \cdot y_r^2} \cdot \left( \frac{\partial \hat{y}(k)}{\partial v} \right)^2 \cdot \left\{ 2 - \frac{\eta}{P \cdot y_r^2} \cdot \left( \frac{\partial \hat{y}(k)}{\partial v} \right)^2 \right\} \quad (C.9)$$

where  $y_r = \left( \max_{p=1}^P y(p) - \min_{p=1}^P y(p) \right)$ ,

Therefore

$$\Delta V(k) = -\lambda \cdot e^2(k) \quad (C.10)$$

$$\text{Where } \lambda = \frac{1}{2} \cdot \frac{\eta}{P \cdot y_r^2} \cdot \left( \frac{\partial \hat{y}(k)}{\partial v} \right)^2 \cdot \left\{ 2 - \frac{\eta}{P \cdot y_r^2} \cdot \left( \frac{\partial \hat{y}(k)}{\partial v} \right)^2 \right\}$$

From the Laypunov stability theorem, the stability is guaranteed if  $V(k)$  be positive and  $V(k)$  be negative. From (C.3),  $V(k)$  is already positive. The condition of stability is depending on  $V(k)$  to be negative. Therefore, we consider  $\lambda > 0$  for all models.

Because  $\frac{1}{2} \cdot \frac{\eta}{P \cdot y_r^2} \cdot \left( \frac{\partial \hat{y}(k)}{\partial v} \right)^2 > 0$  then the convergence condition is limited to:

$$2 - \frac{\eta}{P \cdot y_r^2} \cdot \left( \frac{\partial \hat{y}(k)}{\partial v} \right)^2 > 0 \Rightarrow \frac{\eta}{P \cdot y_r^2} \cdot \left( \frac{\partial \hat{y}(k)}{\partial v} \right)^2 < 2 \Rightarrow \eta < (2 \cdot P \cdot y_r^2) / \left( \frac{\partial \hat{y}(k)}{\partial v} \right)^2 \quad (C.11)$$

Maximum of learning rate  $\eta$  changes in a fixed range. Since  $2 \cdot P \cdot y_r^2$  is not depending of the model, the value of  $\eta_{Max}$  guarantees the convergence can be found out by minimizing the

term of  $\left| \frac{\partial \hat{y}(k)}{\partial v_i} \right|$ . Therefore,

$$0 < \eta < \eta_{Max} \quad (C.12)$$

$$\text{where } \eta_{Max} = (2 \cdot P \cdot y_r^2) / \text{Max} \left( \frac{\partial \hat{y}(k)}{\partial v} \right)^2$$

## C-2 (Chapter 4): Convergence theorems of TSK neuro-fuzzy model

Theorem 4.1: The asymptotic learning convergence of S-P and CS-P configurations are guaranteed if the learning rate for different learning parameters follows the upper bound as mentioned below:

$$0 < \eta_w < 2 \cdot P \cdot y_r^2 \quad (\text{C.13})$$

$$0 < \eta_\sigma < \frac{2 \cdot P \cdot y_r^2}{\max_m |w(\mathbf{X})|^2 \cdot \left(\frac{2}{\sigma_{\min}^3}\right)^2} \quad (\text{C.14})$$

$$0 < \eta_{\bar{x}} < \frac{2 \cdot P \cdot y_r^2}{\max_m |w(\mathbf{X})|^2 \cdot \left(\frac{2}{\sigma_{\min}^2}\right)^2} \quad (\text{C.15})$$

Proof: (C.12) for NF models can be written as:

$$0 < \eta_\nu < \frac{2 \cdot P \cdot y_r^2}{\left| \frac{\partial \hat{Y}_{NF}}{\partial \nu} \right|_{\max}^2} \quad (\text{C.16})$$

Because  $\beta_m = \frac{\mu_{A^m}(\mathbf{X})}{\sum_{m=1}^M \mu_{A^m}(\mathbf{X})} \leq 1$  for all  $m$  and since local models have same variables i.e.  $\mathbf{X}$ ,

therefore, from (4.26-4.29), equations (C.13 to C.15) easily can be derived.

**Theorem 4.2:** The asymptotic learning convergence of P and PS-P configurations are guaranteed if the learning rate for different learning parameters follows the upper bound as mentioned below:

$$0 < \eta_w < 2 \cdot P \cdot y_r^2 \quad (\text{C.17})$$

$$0 < \eta_\sigma < \frac{2 \cdot P \cdot y_r^2}{\max_m |w(\mathbf{X}')|^2 \cdot \left(\frac{2}{\sigma_{\min}^3}\right)^2} \quad (\text{C.18})$$

$$0 < \eta_{\bar{x}} < \frac{2 \cdot P \cdot y_r^2}{\max_m |w(\mathbf{X}')|^2 \cdot \left(\frac{2}{\sigma_{\min}^2}\right)^2} \quad (\text{C.19})$$

Proof: Because  $\beta_m = \frac{\mu_{A^m}(\mathbf{X})}{\sum_{m=1}^M \mu_{A^m}(\mathbf{X})} \leq 1$  and since local models have same variables i.e.  $\mathbf{X}'$ ,

therefore, from (4.26-4.29), equations (C.17 to C.19) easily can be derived.

### C-3 (Chapter 5): Convergence theorems of WNF model

Theorem 5.1: The asymptotic learning convergence is guaranteed if the learning rate for different learning parameters follows the upper bound as mentioned below:

$$0 < \eta_\sigma < \frac{2 \cdot P \cdot y_r^2}{\left| \hat{Y}_{WNN} \right|_{\max}^2 \cdot \left( \frac{2}{\sigma_{\min}^3} \right)^2} \quad (\text{C.20})$$

$$0 < \eta_{\bar{x}} < \frac{2 \cdot P \cdot y_r^2}{\left| \hat{Y}_{WNN} \right|_{\max}^2 \cdot \left( \frac{2}{\sigma_{\min}^2} \right)^2} \quad (\text{C.21})$$

$$0 < \eta_w < \frac{2 \cdot P \cdot y_r^2}{\left| \frac{\partial \hat{Y}_{WNN}}{\partial w} \right|_{\max}^2} \quad (\text{C.22})$$

$$0 < \eta_{C_s} < \frac{2 \cdot P \cdot y_r^2}{\left| \frac{\partial \hat{Y}_{WNN}}{\partial C_s} \right|_{\max}^2} \quad (\text{C.23})$$

$$0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{\left| \frac{\partial \hat{Y}_{WNN}}{\partial C_w} \right|_{\max}^2} \quad (\text{C.24})$$

Proof: (C.12) for WNF models can be written as:

$$0 < \eta_v < \frac{2 \cdot P \cdot y_r^2}{\left| \frac{\partial \hat{Y}_{WNF}}{\partial v} \right|_{\max}^2} \quad (\text{C.25})$$

where  $\nu$  in consequent part of the rules is  $\mathcal{W}$ ,  $C_N$  or  $C_W$  and in premise part of the rules is  $\sigma$  or  $\bar{x}$ . For the parameters  $\mathcal{W}$ ,  $C_N$  and  $C_W$  by applying the partial derivatives to (5.5), we have:

$$\frac{\partial \hat{Y}_{WNF}}{\partial \mathcal{W}} = \beta_m \cdot \frac{\partial \hat{Y}_{WNN_m}}{\partial \mathcal{W}} \quad (\text{C.26})$$

$$\frac{\partial \hat{Y}_{WNF}}{\partial C_N} = \beta_m \cdot \frac{\partial \hat{Y}_{WNN_m}}{\partial C_N} \quad (\text{C.27})$$

$$\frac{\partial \hat{Y}_{WNF}}{\partial C_W} = \beta_m \cdot \frac{\partial \hat{Y}_{WNN_m}}{\partial C_W} \quad (\text{C.28})$$

Because  $\beta_m = \frac{\mu_{A^m}(\mathbf{X})}{\sum_{m=1}^M \mu_{A^m}(\mathbf{X})} \leq 1$  for all  $m$ , therefore (C.22 to C.24) easily is derived.

From (5.5, 4.4, 4.7) for parameters  $\sigma$  or  $\bar{x}$  there is:

$$\begin{aligned} \frac{\partial \hat{Y}_{WNF}}{\partial \sigma} &= \hat{Y}_{WNN_m} \cdot \frac{\beta_m}{\mu_{A^m}} \cdot (1 - \beta_m) \cdot \frac{2 \cdot (x_i - \bar{x}_{m_i})^2}{\sigma_{m_i}^3} \\ &= \hat{Y}_{WNN_m} \cdot \frac{(1 - \beta_m)}{\sum_{m=1}^M \mu_{A^m}} \cdot \frac{2 \cdot (x_i - \bar{x}_{m_i})^2}{\sigma_{m_i}^3} \end{aligned} \quad (\text{C.29})$$

$$\begin{aligned} \frac{\partial \hat{Y}_{WNF}}{\partial \bar{x}} &= \hat{Y}_{WNN_m} \cdot \frac{\beta_m}{\mu_{A^m}} \cdot (1 - \beta_m) \cdot \frac{2 \cdot (x_i - \bar{x}_{m_i})}{\sigma_{m_i}^2} \\ &= \hat{Y}_{WNN_m} \cdot \frac{(1 - \beta_m)}{\sum_{m=1}^M \mu_{A^m}} \cdot \frac{2 \cdot (x_i - \bar{x}_{m_i})}{\sigma_{m_i}^2} \end{aligned} \quad (\text{C.30})$$

and therefore (C.20 to C.21) is derived:

## C-4 (Chapter 6): Convergence theorems of Recurrent SS-W and MS-W neuron models

To prove convergence of the recurrent networks these facts are needed:

Fact 1: Let  $g(y) = ye^{(-y^2)}$ . Then  $|g(y)| < 1, \forall y \in \mathfrak{R}$

Fact 2: let  $f(y) = y^2e^{(-y^2)}$ . Then  $|f(y)| < 1, \forall y \in \mathfrak{R}$

Fact 3: let  $\theta(y) = \frac{1}{1+e^{-y}}$  a sigmoid function. Then  $|\theta(y)| < 1, \forall y \in \mathfrak{R}$

Fact 4: let  $\psi_{a,b}(y) = e^{-\left(\frac{y-b}{a}\right)^2} \cos\left(5\frac{y-b}{a}\right)$  a Morlet wavelet function. Then

$$|\psi_{a,b}(y)| < 1, \forall y, a, b \in \mathfrak{R}$$

### C-4.1 Stability analysis of the Recurrent SS-W neuron models

In this section, Theorems (6.7) to (6.11) for convergence analysis of the SS-RW, SRS-W, SFS-W, SFW-S, SRN, respectively, are presented.

#### a) Summation Sigmoid-Recurrent Wavelet (SS-RW)

Suppose  $Z = \sum_{i=1}^n C'_{s_i} \cdot x_i(k)$  and  $S = \sum_{i=1}^n C'_{w_i} \cdot x_i(k) + Q'_w \cdot y'_w(k-1)$

From the facts 3 and 4:

For parameter  $W$  in all models:

$$\frac{\partial \hat{y}}{\partial W_j} = y_j < |y'_w + y'_\theta| < 1+1=2 \quad (\text{C.31})$$

Therefore  $0 < \eta_w < \frac{2 \cdot P \cdot y_r^2}{2^2} = \frac{P \cdot y_r^2}{2}$

Differential of output of the model for another learning parameter is:

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial C_{w_i}'} &= x_i(k) \cdot W' \cdot \psi' \left( \sum_{i=1}^n C_{w_i}' \cdot x_i(k) + Q_w' \cdot y_w'(k-1) \right) < \\ 1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right| < \\ \left\{ \frac{2}{a_{\min}} \cdot 1.1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7 \end{aligned} \quad (C.32)$$

$$\text{Therefore } 0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{7^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial C_{s_i}'} &= x_i(k) \cdot W' \cdot \theta' \left( \sum_{i=1}^n C_{s_i}' \cdot x_i(k) \right) < \\ 1 \cdot 1 \cdot \theta(z) \cdot (1 - \theta(z)) < 1 \cdot 1 = 1 \end{aligned} \quad (C.33)$$

$$\text{Therefore } 0 < \eta_{C_s} < \frac{2 \cdot P \cdot y_r^2}{1^2} = 2 \cdot P \cdot y_r^2$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial Q_w'} &= W' \cdot y_w'(k-1) \cdot \psi' \left( \sum_{i=1}^n C_{w_i}' \cdot x_i(k) + Q_w' \cdot y_w'(k-1) \right) < \\ 1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right| < \\ \left\{ \frac{2}{a_{\min}} \cdot 1.1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7 \end{aligned} \quad (C.34)$$

$$\text{Therefore } 0 < \eta_{Q_w} < \frac{2 \cdot P \cdot y_r^2}{7^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

#### b) Summation Recurrent Sigmoid–Wavelet (SRS-W)

$$\text{Suppose } Z = \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_s' \cdot y_s'(k-1) \text{ and } S = \sum_{i=1}^n C_{w_i}' \cdot x_i(k)$$

By using the facts 1 to 4 and differential of network output to learning parameters:

$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial C'_{w_i}} &= x_i(k) \cdot W' \cdot \psi' \left( \sum_{i=1}^n C'_{w_i} \cdot x_i(k) \right) < \\
1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right| < \\
\left\{ \frac{2}{a_{mn}} \cdot 1.1 + \frac{5}{a_{mn}} \cdot 1 \right\} < 7
\end{aligned} \tag{C.35}$$

$$\text{Therefore } 0 < \eta_{C_u} < \frac{2 \cdot P \cdot y_r^2}{7^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial C'_s} &= x_i(k) \cdot W' \cdot \theta' \left( \sum_{i=1}^n C'_{s_i} \cdot x_i(k) + Q'_s \cdot y'_\theta(k-1) \right) < \\
1 \cdot 1 \cdot \theta(z) \cdot (1 - \theta(z)) < 1 \cdot 1 < 1
\end{aligned} \tag{C.36}$$

$$\text{Therefore } 0 < \eta_{C_s} < \frac{2 \cdot P \cdot y_r^2}{1^2} = 2 \cdot P \cdot y_r^2$$

$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial Q'_s} &= W' \cdot y'_\psi(k-1) \cdot \theta' \left( \sum_{i=1}^n C'_{s_i} \cdot x_i(k) + Q'_s \cdot y'_\psi(k-1) \right) < \\
1 \cdot 1 \cdot \theta(z) \cdot (1 - \theta(z)) < 1 \cdot 1 < 1
\end{aligned} \tag{C.37}$$

$$\text{Therefore } 0 < \eta_{Q_s} < \frac{2 \cdot P \cdot y_r^2}{1^2} = 2 \cdot P \cdot y_r^2$$

c) *Summation Feedback to Sigmoid from Wavelet (SFS-W)*

$$\text{Suppose } Z = \sum_{i=1}^n C'_{s_i} \cdot x_i(k) + Q'_{ws} \cdot y'_\theta(k-1) \text{ and } S = \sum_{i=1}^n C'_{w_i} \cdot x_i(k)$$

By calculating the maximum value of the difference of the network output to the learning parameter:



$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial C_{w_i}'} &= x_i(k) \cdot W' \cdot \psi' \left( \sum_{i=1}^n C_{w_i}' \cdot x_i(k) \right) < \\
1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right| < \\
\left\{ \frac{2}{a_{\min}} \cdot 1 \cdot 1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7
\end{aligned} \tag{C.38}$$

$$\text{Therefore } 0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{7^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial C_{s_i}'} &= x_i(k) \cdot W' \cdot \theta' \left( \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_{ws}' \cdot y_{\psi}'(k-1) \right) < \\
1 \cdot 1 \cdot \theta(z) \cdot (1 - \theta(z)) < 1 \cdot 1 < 1
\end{aligned} \tag{C.39}$$

$$\text{Therefore } 0 < \eta_{C_s} < \frac{2 \cdot P \cdot y_r^2}{1^2} = 2 \cdot P \cdot y_r^2$$

$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial Q_{ws}'} &= W' \cdot y_{\psi}'(k-1) \cdot \theta' \left( \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_{ws}' \cdot y_{\psi}'(k-1) \right) < \\
1 \cdot 1 \cdot \theta(z) \cdot (1 - \theta(z)) < 1 \cdot 1 < 1
\end{aligned} \tag{C.40}$$

$$\text{Therefore } 0 < \eta_{Q_{ws}} < \frac{2 \cdot P \cdot y_r^2}{1^2} = 2 \cdot P \cdot y_r^2$$

d) *Summation Feedback to Wavelet from Sigmoid (SFW -S)*

$$\text{Suppose } Z = \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_{sw}' \cdot y_{\theta}'(k-1) \text{ and } S = \sum_{i=1}^n C_{w_i}' \cdot x_i(k)$$

$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial C'_{w_i}} &= x_i(k) \cdot W' \cdot \psi' \left( \sum_{i=1}^n C'_{w_i} \cdot x_i(k) + Q'_{sw} \cdot y'_\theta(k-1) \right) < \\
1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right| < \\
\left\{ \frac{2}{a_{\min}} \cdot 1.1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7
\end{aligned} \tag{C.41}$$

$$\text{Therefore } 0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{7^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial C'_{s_i}} &= x_i(k) \cdot W' \cdot \theta' \left( \sum_{i=1}^n C'_{s_i} \cdot x_i(k) \right) < \\
1 \cdot 1 \cdot \theta(z) \cdot (1 - \theta(z)) < 1 \cdot 1 < 1
\end{aligned} \tag{C.42}$$

$$\text{Therefore } 0 < \eta_{C_s} < \frac{2 \cdot P \cdot y_r^2}{1^2} = 2 \cdot P \cdot y_r^2$$

$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial Q'_{sw}} &= W' \cdot y'_\theta(k-1) \cdot \psi' \left( \sum_{i=1}^n C'_{w_i} \cdot x_i(k) + Q'_{sw} \cdot y'_\theta(k-1) \right) < \\
1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right| < \\
\left\{ \frac{2}{a_{\min}} \cdot 1.1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7
\end{aligned} \tag{C.43}$$

$$\text{Therefore } 0 < \eta_{Q_{sw}} < \frac{2 \cdot P \cdot y_r^2}{7^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

e) *Summation Recurrent Neuron (SRN)*

$$\text{Suppose } Z = \sum_{i=1}^n C'_{s_i} \cdot x_i(k) + Q'_{s'} \cdot y'_\theta(k-1) \text{ and } S = \sum_{i=1}^n C'_{w_i} \cdot x_i(k)$$

$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial C'_{w_i}} &= x_i \cdot W'(k) \cdot \psi' \left( \sum_{i=1}^n C'_{w_i} \cdot x_i(k) + Q' \cdot y'(k-1) \right) < \\
1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right| < \\
\left\{ \frac{2}{a_{\min}} \cdot 1.1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7
\end{aligned} \tag{C.44}$$

$$\text{Therefore } 0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{7^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial C'_{s_i}} &= x_i(k) \cdot W^j \cdot \theta' \left( \sum_{i=1}^n C'_{s_i} \cdot x_i(k) + Q' \cdot y'(k-1) \right) < \\
1 \cdot 1 \cdot \theta(z) \cdot (1 - \theta(z)) < 1 \cdot 1 < 1
\end{aligned} \tag{C.45}$$

$$\text{Therefore } 0 < \eta_{C_s} < \frac{2 \cdot P \cdot y_r^2}{1^2} = 2 \cdot P \cdot y_r^2$$

$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial Q'} &= W' \cdot y'(k-1) \cdot \\
\left[ \psi' \left( \sum_{i=1}^n C'_{w_i} \cdot x_i(k) + Q' \cdot y'(k-1) \right) + \theta' \left( \sum_{i=1}^n C'_{s_i} \cdot x_i(k) + Q' \cdot y'(k-1) \right) \right] < \\
1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) + \theta(z) \cdot (1 - \theta(z)) \right| < \\
\left| \frac{-2}{a_{\min}} \cdot 1.1 - \frac{5}{a_{\min}} \cdot 1 + 1 \cdot 1 \right| < 6
\end{aligned} \tag{C.46}$$

$$\text{Therefore } 0 < \eta_Q < \frac{2 \cdot P \cdot y_r^2}{6^2} = \frac{P \cdot y_r^2}{18}$$

### C-4.2 Stability analysis of the Recurrent MS-W neuron models

In this section, Theorems (6.7) to (6.11) for convergence analysis of the MS-RW, MRS-W, MFS-W, MFW-S, MRN, respectively, are presented.

a) *Multiplication Sigmoid-Recurrent Wavelet (MS-RW)*

From fact 3 and 4:

$$\text{Suppose } Z = \sum_{i=1}^n C_{S_i}^j \cdot x_i(k) \text{ and } S = \sum_{i=1}^n C_{W_i}^j \cdot x_i(k) + Q_{W'}^j \cdot y_{\psi'}^j(k-1)$$

For parameter  $W$  in all networks:

$$\frac{\partial \hat{y}}{\partial W^j} = y_j = y_{\psi}^j \cdot y_{\theta}^j < 1.1 < 1 \quad (\text{C.47})$$

$$\text{Therefore } 0 < \eta_w < \frac{2 \cdot P \cdot y_r^2}{1} < 2 \cdot P \cdot y_r^2$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial C_{W_i}^j} &= x_i(k) \cdot W^j \cdot \theta \left( \sum_{i=1}^n C_{S_i}^j \cdot x_i(k) \right) \cdot \psi' \left( \sum_{i=1}^n C_{W_i}^j \cdot x_i(k) + Q_{W'}^j \cdot y_{\psi'}^j(k-1) \right) < \\ 1.1.1 \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos \left( 5 \frac{S-b}{a} \right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin \left( 5 \frac{S-b}{a} \right) \right| &< \left\{ \frac{2}{a_{\min}} \cdot 1.1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7 \end{aligned} \quad (\text{C.48})$$

$$\text{Therefore } 0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{(7)^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial C_{S_i}^j} &= x_i(k) \cdot W^j \cdot \theta' \left( \sum_{i=1}^n C_{S_i}^j \cdot x_i(k) \right) \cdot \psi \left( \sum_{i=1}^n C_{W_i}^j \cdot x_i(k) + Q_{W'}^j \cdot y_{\psi'}^j(k-1) \right) < \\ 1.1 \cdot \theta(Z) \cdot (1 - \theta(Z)) \cdot 1 &< 1.1 < 1 \end{aligned} \quad (\text{C.49})$$

$$\text{Therefore } 0 < \eta_{C_s} < \frac{2 \cdot P \cdot y_r^2}{(1)^2} = 2 \cdot P \cdot y_r^2$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial Q_{W'}^j} &= W^j \cdot y_{\psi'}^j(k-1) \cdot \theta \left( \sum_{i=1}^n C_{S_i}^j \cdot x_i(k) \right) \cdot \psi' \left( \sum_{i=1}^n C_{W_i}^j \cdot x_i(k) + Q_{W'}^j \cdot y_{\psi'}^j(k-1) \right) < \\ 1.1.1 \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos \left( 5 \frac{S-b}{a} \right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin \left( 5 \frac{S-b}{a} \right) \right| &< \left\{ \frac{2}{a_{\min}} \cdot 1.1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7 \end{aligned} \quad (\text{C.50})$$

$$\text{Therefore } 0 < \eta_{Q_w} < \frac{2 \cdot P \cdot y_r^2}{(7)^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

b) *Multiplication Recurrent Sigmoid-Wavelet (MRS-W)*

Suppose  $Z = \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_s' \cdot y_\theta'(k-1)$  and  $S = \sum_{i=1}^n C_{w_i}' \cdot x_i(k)$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial C_{w_i}'} &= x_i(k) \cdot W' \cdot \theta' \left( \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_s' \cdot y_\theta'(k-1) \right) \cdot \psi' \left( \sum_{i=1}^n C_{w_i}' \cdot x_i(k) \right) < \\ 1 \cdot 1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right| &< \left\{ \frac{2}{a_{\min}} \cdot 1.1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7 \end{aligned} \quad (C.51)$$

$$\text{Therefore } 0 < \eta_{c_w} < \frac{2 \cdot P \cdot y_r^2}{(7)^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial C_{s_i}'} &= x_i(k) \cdot W' \cdot \theta' \left( \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_s' \cdot y_\theta'(k-1) \right) \cdot \psi \left( \sum_{i=1}^n C_{w_i}' \cdot x_i(k) \right) < \\ 1 \cdot 1 \cdot \theta(Z) \cdot (1 - \theta(Z)) \cdot 1 &< 1 \cdot 1 < 1 \end{aligned} \quad (C.52)$$

$$\text{Therefore } 0 < \eta_{c_s} < \frac{2 \cdot P \cdot y_r^2}{(1)^2} = 2 \cdot P \cdot y_r^2$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial Q_s'} &= W' \cdot y_\nu'(k-1) \cdot \theta' \left( \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_s' \cdot y_\nu'(k-1) \right) \cdot \psi \left( \sum_{i=1}^n C_{w_i}' \cdot x_i(k) \right) < \\ 1 \cdot 1 \cdot \theta(Z) \cdot (1 - \theta(Z)) \cdot 1 &< 1 \cdot 1 < 1 \end{aligned} \quad (C.53)$$

$$\text{Therefore } 0 < \eta_{Q_s} < \frac{2 \cdot P \cdot y_r^2}{(1)^2} = 2 \cdot P \cdot y_r^2$$

c) *Multiplication Feedback to Sigmoid from Wavelet (MFS-W)*

Suppose  $Z = \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_{ws}' \cdot y_\theta'(k-1)$  and  $S = \sum_{i=1}^n C_{w_i}' \cdot x_i(k)$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial C_{w_i}'} &= x_i(k) \cdot W' \cdot \theta' \left( \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_{ws}' \cdot y_\theta'(k-1) \right) \cdot \psi' \left( \sum_{i=1}^n C_{w_i}' \cdot x_i(k) \right) < \\ 1 \cdot 1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right| &< \left\{ \frac{2}{a_{\min}} \cdot 1.1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7 \end{aligned} \quad (C.54)$$

$$\text{Therefore } 0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{(7)^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

$$\frac{\partial \hat{y}(k)}{\partial C_{s_i}'} = x_i(k) \cdot W_i \cdot \theta' \left( \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_{ws}' \cdot y_w'(k-1) \right) \cdot \psi \left( \sum_{i=1}^n C_{w_i}' \cdot x_i(k) \right) < \quad (C.55)$$

$$1 \cdot 1 \cdot \theta(Z) \cdot (1 - \theta(Z)) \cdot 1 < 1 \cdot 1 < 1$$

$$\text{Therefore } 0 < \eta_{C_s} < 2 \cdot P \cdot y_r^2$$

$$\frac{\partial \hat{y}(k)}{\partial Q_{ws}'} = W' \cdot y_w'(k-1) \cdot \theta' \left( \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_{ws}' \cdot y_w'(k-1) \right) \cdot \psi \left( \sum_{i=1}^n C_{w_i}' \cdot x_i(k) \right) < \quad (C.56)$$

$$1 \cdot 1 \cdot \theta(Z) \cdot (1 - \theta(Z)) \cdot 1 < 1 \cdot 1 < 1$$

$$\text{Therefore } 0 < \eta_{Q_{ws}} < 2 \cdot P \cdot y_r^2$$

d) *Multiplication Sigmoid Feedback to Wavelet (MS-FW)*

$$\text{Suppose } Z = \sum_{i=1}^n C_{s_i}' \cdot x_i(k) + Q_{sw}' \cdot y_\theta'(k-1) \text{ and } S = \sum_{i=1}^n C_{w_i}' \cdot x_i(k)$$

$$\frac{\partial \hat{y}(k)}{\partial C_{w_i}'} = x_i(k) \cdot W' \cdot \psi' \left( \sum_{i=1}^n C_{w_i}' \cdot x_i(k) + Q_{sw}' \cdot y_\theta'(k-1) \right) \cdot \theta \left( \sum_{i=1}^n C_{s_i}' \cdot x_i(k) \right) < \quad (C.57)$$

$$1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right| \cdot 1 < \left\{ \frac{2}{a_{\min}} \cdot 1.1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7$$

$$\text{Therefore } 0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{(7)^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

$$\frac{\partial \hat{y}(k)}{\partial C_{s_i}'} = x_i(k) \cdot W' \cdot \psi' \left( \sum_{i=1}^n C_{w_i}' \cdot x_i(k) + Q_{sw}' \cdot y_\theta'(k-1) \right) \cdot \theta' \left( \sum_{i=1}^n C_{s_i}' \cdot x_i(k) \right) < \quad (C.58)$$

$$1 \cdot 1 \cdot 1 \cdot \theta(Z) \cdot (1 - \theta(Z)) < 1 \cdot 1 < 1$$

$$\text{Therefore } 0 < \eta_{C_s} < 2 \cdot P \cdot y_r^2$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial Q'_{sw}} &= W' \cdot y'_\theta(k-1) \cdot \psi' \left( \sum_{i=1}^n C'_{w_i} \cdot x_i(k) + Q'_{sw} \cdot y'_\theta(k-1) \right) \cdot \theta' \left( \sum_{i=1}^n C'_{s_i} \cdot x_i(k) \right) < \\ 1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right| \cdot 1 < \left\{ \frac{2}{a_{\min}} \cdot 1 \cdot 1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7 \end{aligned} \quad (\text{C.59})$$

$$\text{Therefore } 0 < \eta_{Q_{sw}} < \frac{2 \cdot P \cdot y_r^2}{(7)^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

e) *Multiplication Recurrent to Sigmoid and Recurrent to Wavelet (MRN)*

$$\text{Suppose } Z = \sum_{i=1}^n C'_{s_i} \cdot x_i(k) + Q' \cdot y'_\theta(k-1) \text{ and } S = \sum_{i=1}^n C'_{w_i} \cdot x_i(k)$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial C'_{w_i}} &= x_i(k) \cdot W' \cdot \\ \psi' \left( \sum_{i=1}^n C'_{w_i} \cdot x_i(k) + Q' \cdot y'_\theta(k-1) \right) \cdot \theta' \left( \sum_{i=1}^n C'_{s_i} \cdot x_i(k) + Q' \cdot y'_\theta(k-1) \right) &< \\ 1 \cdot 1 \cdot \left| \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right| \cdot 1 &< \left\{ \frac{2}{a_{\min}} \cdot 1 \cdot 1 + \frac{5}{a_{\min}} \cdot 1 \right\} < 7 \end{aligned} \quad (\text{C.60})$$

$$\text{Therefore } 0 < \eta_{C_w} < \frac{2 \cdot P \cdot y_r^2}{(7)^2} = \frac{2 \cdot P \cdot y_r^2}{49}$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial C'_{s_i}} &= x_i(k) \cdot W' \cdot \\ \psi \left( \sum_{i=1}^n C'_{w_i} \cdot x_i(k) + Q' \cdot y'_\theta(k-1) \right) \cdot \theta' \left( \sum_{i=1}^n C'_{s_i} \cdot x_i(k) + Q' \cdot y'_\theta(k-1) \right) &< \\ 1 \cdot 1 \cdot 1 \cdot \theta(Z) \cdot (1 - \theta(Z)) &< 1 \cdot 1 < 1 \end{aligned} \quad (\text{C.61})$$

$$\text{Therefore } 0 < \eta_{C_s} < 2 \cdot P \cdot y_r^2$$

$$\begin{aligned}
\frac{\partial \hat{y}(k)}{\partial Q^j} &= W^j \cdot y^j(k-1). \\
\left[ \psi^j \left( \sum_{i=1}^n C_{w_i}^j \cdot x_i(k) + Q^j \cdot y^j(k-1) \right) \cdot \theta \left( \sum_{i=1}^n C_{s_i}^j \cdot x_i(k) + Q^j \cdot y^j(k-1) \right) + \right. \\
\left. \psi \left( \sum_{i=1}^n C_{w_i}^j \cdot x_i(k) + Q^j \cdot y^j(k-1) \right) \cdot \theta \left( \sum_{i=1}^n C_{s_i}^j \cdot x_i(k) + Q^j \cdot y^j(k-1) \right) \right] &< \\
1 \cdot 1 \cdot \left\{ \left[ \frac{-2}{a} \cdot \frac{S-b}{a} \cdot e^{-\left(\frac{S-b}{a}\right)^2} \cdot \cos\left(5 \frac{S-b}{a}\right) - e^{-\left(\frac{S-b}{a}\right)^2} \cdot \frac{5}{a} \cdot \sin\left(5 \frac{S-b}{a}\right) \right] \cdot 1 + 1 \cdot \theta(Z) \cdot (1 - \theta(Z)) \right\} &< \\
\left\{ \frac{-2}{a_{\min}} \cdot 1 \cdot 1 - \frac{5}{a_{\min}} \cdot 1 + 1 \cdot 1 \right\} &< |-2 - 5 + 1| < 6
\end{aligned} \tag{C.62}$$

Therefore  $0 < \eta_Q < \frac{2 \cdot P \cdot y_r^2}{6^2} = \frac{2 \cdot P \cdot y_r^2}{36}$



## References

- [Azeem'00a] M. F. Azeem, Soft Computing Based Modeling of Dynamic Systems, *Ph.D Thesis*, Electrical Engineering Department, Indian Institute of Technology, Delhi, 2000
- [Azeem'00b] M. F. Azeem, M. Hanmandlu, N. Ahmad, "Generalization of Adaptive Neuro-Fuzzy Inference Systems", *IEEE Trans. On Neural Networks*, vol. 11, no. 6, 2000
- [Azeem'03a] M. F. Azeem, M. Hanmandlu, N. Ahmad, "Structure Identification of Generalized Adaptive Neuro-Fuzzy Inference Systems", *IEEE Trans. On Fuzzy Systems*, vol. 11, no. 5, 2003
- [Azeem'03b] M. F. Azeem, M. Hanmandlu, N. Ahmad, "Evolutive Learning Algorithms for Fuzzy Modeling", *International Journal of Smart Engineering System Design*, vol. 5, no. 4, 2003
- [Azeem'06] M. F. Azeem, A. Banakar, V. Kumar, "Comparative Study of Different Types of Wavelet Functions in Neural Network", *International Joint Conference on Neural Networks (IJCNN2006)*, Vancouver, BC, Canada, 2006
- [Banakar'06a] A. Banakar, M. F. Azeem, "Generalized Wavelet Neural Network Model and its Application in Time Series Prediction", *International Joint Conference on Neural Networks (IJCNN2006)*, Vancouver, BC, Canada, 2006
- [Banakar'06b] A. Banakar, M. F. Azeem, "A New Artificial Wavelet Neural Network and its Application in Wavelet Neuro-Fuzzy", *IEEE IS'06; 3rd IEEE Conference On Intelligent Systems*, University of Westminster, London, UK, 2006
- [Barbounis'06] T. G. Barbounis et al., "Long-Term Wind Speed and Power Forecasting Using Local Recurrent Neural Network Model", *IEEE Trans. Energy Conversion*, vol. 21, no. 1, pp. 273-284, 2006
- [Benettin'80] G. Benettin, L. Galgani, A. Giorgilli, J. M. Strelcyn, "Lyapunov Characteristic Exponents for smooth dynamical systems and for Hamiltonian systems; A method for computing all of them", *Meccanica, Springer Netherlands*, vol. 15, pp. 21-30, 1980
- [Benveniste'94] A. Benveniste, A. Juditsky, B. Delyon, Q. Zhang, P. Y. Glorennec, "Wavelets in identification", in *Proc. SYSID'94, 10th IFAC Symp. Syst. Identification*, Copenhagen, Denmark, 1994
- [Bernieri'94] A. Bernieri, M. D. Apuzzo, L. Sansone, M. Savastano, "A Neural Network Approach for Identification and Fault Diagnosis on Dynamic Systems", *IEEE Trans. On Instrumentation and Measurement*, vol. 43, no. 6, pp. 867-873, 1994

- [Billings'05] S. A. Billings, H. L. Wei, "A New Class of Wavelet Networks for Nonlinear System Identification", *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 862 – 874, 2005
- [Blum'91] E. K. Blum, L. K. Li, "Approximation theory and feedforward networks", *Neural Networks*, vol. 4, pp. 511-515, 1991
- [Boubez'93] T. I. Boubez, R. L. Peskin, "Wavelet Neural Networks and Receptive Field Partitioning", *IEEE Int. Conf. Neural Networks*, pp. 1544-1549, 1993
- [Box'70] G. E. P. Box, G. M Jenkins, G. C. Reinsel, Time Series Analysis, Forecasting and Control, *San Francisco: Holden Day*, 1970
- [Brock'92] W. A. Brock, D. A. Hsieh, B. Lebaron "Nonlinear Dynamics Chaos and Instability Statistical and Economic Theory", MIT press, Massachusetts, 1992
- [Burrus'97] C. S. Burrus, R. A. Gopinath, H. Guo, Introduction to wavelets and wavelet transforms, Prentice Hall, 1997
- [Cao'03 a] J. Cao, J. Wang, X. Liao, "Novel stability criteria of delayed cellular neural networks", *Int. J. Neural Syst.*, vol. 13, no. 5, pp. 367-375, 2003
- [Cao'03 b] J. Cao, J. Wang, "Global asymptotic stability of a general class of recurrent neural networks with time-varying delays", *IEEE Trans. Circuits Syst. I*, vol. 50, no. 1, pp. 34-44, 2003
- [Cgui'95] C. K. Chui, K. I. Ramachandran, Wavelet Analysis and Its Applications, *Academic Press*, 1995
- [Chakraborty'04] D. Chakraborty, N. R. Pal, "A neurofuzzy scheme for simultaneous feature selection and fuzzy rule-based classification", *IEEE Trans. on Neural Network*, vol. 15, no. 1, pp. 110-123, 2004
- [Chen'94] D. S. Chen, R. C. Jain, "A robust back propagation learning algorithm for fuunction approximation", *IEEE Trans. Neural Networks*, vol. 5, pp. 467-479, 1994
- [Chiu'96] S. Chiu, "Method and Software for Extracting Fuzzy Classification rules by Subtractive Clustering", *1996 NAFIPS conf.*, pp. 461-465, 1996
- [Daubechies'92] I. Daubechies, Ten Lecture on Wavelets, *CBMS series. SIMA*, 1992
- [Davis'89] L. Davis, "Adapting operator probabilities in genetic algorithms", *Proc. of the third International Conference on Genetic Algorithms*, pp. 60-69, 1989
- [Farag'98] W. A. Farag, V. H. Quintana, G. L. Torres, "A genetic-based neuro-fuzzy approach for modeling and control of dynamical systems", *IEEE Trans. Neural Network*, vol. 9, no. 5, pp. 756-767, 1998

- [Forti'94] M. Forti, "On global asymptotic stability of a class of nonlinear systems arising in neural network theory", *J. Differential Equations*, vol. 113, pp. 246-264, 1994
- [Forti'95] M. Forti, A. Tesi, "New conditions for global stability of neural networks with application to linear and quadratic programming problems", *IEEE Trans. Circuits Syst. I*, vol. 42, no. 7, pp. 354-366, 1995
- [Funahashi'89] K. Funahashi, "On the approximate realization of continuous mappings by neural networks", *Neural Networks*, vol. 2, pp. 183-192, 1989
- [Frasconi'92] Frasconi, P., Gori, M., Soda. G., "Local feedback multilayered networks", *Neural Computation*, vol. 4, issue 1, pp.120-130, 1992
- [Frasconi'96] P. Frasconi, M. Gori, "Computational capabilities of local-feedback recurrent networks acting as finite-state machines", *IEEE Trans. on Neural Networks (Letters)*, vol. 7, no. 6, pp. 1521-1525, 1996
- [Friedman'91] J. H. Friedman, "Multivariate Adaptive Regression Splines", *The Annals of Statistics*, 19:1, pp. 1-141, 1991
- [Gadgil'02] S. Gadgil, et al., "On forecasting the Indian summer monsoon: The intriguing season of 2002", *Current Science*, vol. 83, pp. 394-403, 2002
- [Gadgil'05] S. Gadgil, M. Rajeevan, R. Nanjundiah, " Monsoon prediction – Why yet another failure?", *Current Science*, vol. 88, no. 9, 2005
- [Gebhardt'94] J. E. Gebhardt, R. Kruse, F. Klawonn, *Foundations of Fuzzy Systems*, Wiley, 1994
- [Goldberg'91] D. E. Goldberg, "Real-coded genetic algorithms, virtual alphabets, and blocking", *Complex Systems*, vol. 5, pp.139-167, 1991
- [Gowariker'89] V. Gowariker, V. Thapliyal, R. P. Sarkar, C. S. Mandal, D. R. Sikka, "Parametric and power regression models: New approach for long range forecasting of monsoon rainfall in India", *Mausam*, 40, pp. 115-122, 1989
- [Hartman'90] E. J. Hartman, J. D. Keeler, J. M. Kowalski, "Layered neural networks with Gaussian hidden units as universal approximations", *Neural Computation*, vol. 2, pp. 210-215, 1990
- [Hinton'87] G. E. Hinton, S. J. Nowlan, "How learning can guide evolution", *Complex Systems*, vol. 1, pp. 495-502, 1987
- [Ho'01] D. W. C. Ho, P.A. Zhang, J. Xu, "Fuzzy wavelet networks for function learning", *IEEE Transactions on Fuzzy Systems*, vol. 9, Issue 1, pp. 200 – 211, 2001
- [Homic'89] K. Homic, "Multilayer feedforward neural networks are universal approximator", *Neural Networks*, vol. 2, 1989

- [Hurst'51] H. E. Hurst, "Long-term storage of reservoirs", *Transaction of the American Society of Civil Engineering*, vol. 116, pp. 770-808, 1951
- [Ihara'80] J. Ihara, "Group method of data handling towards a modeling of complex systems-IV", *Systems and Control (in Japanese)*, vol. 24, pp. 158-168, 1980
- [IITM] <http://www.tropmet.res.in/>  
Indian Institute of Tropical Meteorology, Pune, India, 411008
- [Isham'93] V. Isham, "Statistical Aspects of Chaos, A review in : Networks and Chaos Statistical and Probabilistic Aspects", *Chamann & Hall*, London, 1993
- [Kalsi'04] S. R. Kalsi, et al., "Various aspects of unusual behaviour of monsoon 2002, 2004", *IMD Met. Monograph, Synoptic Meteorology 2/2004*, pp. 97, 2004
- [Kang'05] I. S. Kang, "Dynamical seasonal prediction and predictability of monsoon", *Review topic B1c: Numerical Modeling-Predictability*, Climate Environment System Research Center, Seoul National University, 2005
- [Khaloozadeh'04] H. Khaloozadeh, A. K. Sedigh., "On the Predictability of Tehran Price Index, TEPIX", *III International Conference on System Identification and Control Problems, Moscow*, pp. 1279-1287, 2004
- [Klir'03] G. J. Klir, B. Yuan, *Fuzzy Sets and Fuzzy Logic; Theory and Application*, *Prentice Hall*, 2003
- [Krishnamurti'98] T. N. Krishnamurti, C. M. Kishtawal, D. R. LaRow, Z. Zhang, C. E. Willford, S. Gadgil, S. Surendran, "Improved weather and seasonal climate prediction forecasts from multimodel superensemble", *Science*, 285, pp. 1548- 1550, 1998
- [Jacobs'91] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, "Adaptive mixtures of local experts", *Neural Computation*, vol. 3, pp. 79-87, 1991
- [Jang'93] J. S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system", *IEEE Tran. on Sys., Man and Cybernetics*, vol. 23, issue 3, pp. 665 – 685, 1993
- [Jin'99] L. Jin, M. M. Gupta, "Stable Dynamic Backpropagation Learning in Recurrent Neural Networks", *IEEE Trans. Neural Networks*, vol. 10, no. 6, pp. 1321-1334, 1999
- [Juang'02] C. F. Juang, "A TSK-Type Recurrent Fuzzy Network for Dynamic Systems Processing by Neural Network and Genetic Algorithms", *IEEE Trans. on Fuzzy Systems*, vol. 10, No. 2, pp. 155-170, 2002
- [Lee'00] C. H. Lee, C. C. Teng, "Identification and control of dynamic systems using Recurrent Fuzzy Neural Networks", *IEEE Trans. on Fuzzy Systems*, vol. 8, no. 4, pp. 349-366, 2000

- [Li'04] X. Li, L. Huang, J. Wu, "A new method of lyapunov functionals for delayed cellular neural networks", *IEEE Trans. Circuits Syst. I*, vol. 51, no. 11, pp. 2263-2270, 2004
- [Li'05] X. D. Li, J. K. L. Ho, T. W. S Chow, "Approximation of Dynamical Time-Variant Systems by Continues-Time Recurrent Neural Networks", *IEEE Trans. on Circuits and systems*, vol. 52, no. 10, pp. 656-660, 2005
- [Li'06] C. Li, X. Liao, "Robust Stability and Robust Periodicity of Delayed Recurrent Neural Networks With Noise Disturbance", *IEEE Trans. Circuits Syst. I*, vol. 53, no. 10, pp. 2265-2273, 2006
- [Liang'01] X. B. Liang , J. Si, "Global exponential stability of neural networks with globally Lipschitz continuous activations and its application to linear variational inequality problem", *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 349-359, 2001
- [Lin'03] C. J. Lin, C. C. Chin, C. L. Lee, "A wavelet based neuro-fuzzy system and its applications", *Proceedings of the Int. Joint Conference on Neural Networks*, vol. 3, pp. 1921-1926, 2003
- [Lin'04] C. J. Lin, C. C. Chin, "Prediction and Identification Using Wavelet-Based Recurrent Fuzzy Neural Network", *IEEE Trans. on Sys. Man and Cybern. Part B*, vol. 34, no.5, pp.2144-2154, 2004
- [Lin'06] C. K. Lin, "Nonsingular Terminal Sliding Mode Control of Robot Manipulators Using Fuzzy Wavelet Networks", *IEEE Trans. on Fuzzy Systems*, vol. 14, no. 6, pp. 849-859, 2006
- [Liu'04] D. Liu, S. Hu, J. Wang, "Global output convergence of a class of continuous-time recurrent neural networks with time-varying thresholds" , *IEEE Trans. Circuits Syst. II*, vol. 51, no. 4, pp. 161-167, 2004
- [Lobo'97] F. Lobo, D. E. Goldberg, "Decision making in a hybrid genetic algorithm", *Proc. of IEEE Intl Conf. on Evolutionary Computation*, pp. 121-125, 1997
- [Mastorocostas'02] P. A. Mastorocostas, J. B. Theocharis, "A Recurrent Fuzzy-Neural Model for Dynamic system Identification", *IEEE Transaction on System, Man and Cybernetics-Part B: Cybernetics*, vol. 32, no. 2, pp. 176-190, 2002
- [Moody'89] J. Moody, C. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units", *Neural Computation*, vol. 1, pp. 281-294, 1989
- [Narendra'90] K. S. Narendra, K. Parthasarathy, "Identification and control of dynamical systems using neural networks", *IEEE Trans. on Neural Networks*, vol. 1, no. 1, 1990
- [Nauck'97] D. Nauck, R. Kruse, F. Klawonn, Foundations of Neuro-Fuzzy Systems, *John Wiley & Sons Inc*, 1997

- [Oseledec'68] V. I. Oseledec, "A Multiplication Ergodic Theorem, Lyapunov Characteristic Numbers for Dynamical Systems", *Trudy Moskov. Mat. Obsc.*, 19, pp.197-221, 1968
- [Oussar'00] Y. Oussar, G. Dreyfus, "Initialization by Selection for Wavelet Network Training", *Neurocomputing*, vol.34, pp. 131-143, 2000
- [Pal'99] P. K. Pal, W. J. Prakash, P. K. Thapliyal, C. M. Kishtawal, "A Technique of Rainfall Assimilation for Dynamic Extended Range Monsoon Prediction", *Meteorology and Atmospheric Physics, Printed in Austria*, pp. 157-168, 1999
- [Palmer'94] T. N. Palmer, D. L. T. Anderson., "The prospects for seasonal forecasting: A review paper", *Quart. J. Roy. Meteor. Soc.*,120, p.p. 755-793, 1994
- [Patra'99] J. C. Patra, R. N. Pal, B. N. Chatterji, G. Panda, "Identification of Nonlinear Dynamic Systems Using Functional Link Artificial Neural Networks", *IEEE Trns. on System, Man and Cybernetics-Part B: Cybernetics*, vol. 29, no. 2, pp. 254-262, 1999
- [Patterson'96] D. W. Patterson, *Artificial Neural Network; Theory and Applications*, Prentice Hall, 1996
- [Pedrycz'04] W. Pedrycz, F. Gomide, *An Introduction to Fuzzy Sets; Analysis and Design*, Prentice Hall of India, 2004
- [Piche'00] S. Piche, "Steepest Descent Algorithm for neural network controllers and filters", *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 198-221, 1994
- [Poggio'90a] T. Poggio, F. Girosi, "Networks for approximation and learning", *Proceedings of the IEEE*, vol. 78, Issue 9, pp. 1481 – 1497, 1990
- [Poggio'90b] T. Poggio, F. Girosi, "Regularization Algorithm for Limiting That Are equivalent to multilayer Networks", *Science*, vol. 247, pp. 978-982, 1990
- [Qin'92] S. Z. Qin, H. T. Su, T. J. McAvoy, "Comparison of Four Neural Net Learning Methods for Dynamic System Identification", *IEEE Trans. On Neural Networks*, vol. 3, no. 1, pp. 122-130, 1992
- [Quang'05] C. S. Quang, W. J. Lee, S. J. Lee, "A TSK-Type Neuro-fuzzy Network Approach to System Modeling Problems," *IEEE Trans. on Sys. Man and cybernetics, part-B cybernetics*, vol. 35, no. 4, 2005
- [Rajeevan'04] M. Rajeevan, D. S. Pai, S. K Dikshit, R. R. Kelkar, "IMD's new operational models for long range forecast of southwest monsoon rainfall over India and their verification for 2003", *Current Science*, vol. 86, pp. 422–431, 2004
- [Rao'04] R. M. Rao, A. S. Bopardikar, *Wavelet Transform: Introduction to theory and applications*, Pearson education, 2004

- [Rumelhart'86] D. E. Rumelhart, et.al. , “Learning Internal Representations by Error Propagation”, In D. E. Rumelhart and J. L. McClelland (Eds.). *Parallel Distributed Processing I*, Cambridge: MIT Press, pp. 675-695, 1986
- [Sanger'90] T. D. Sanger, "Basis-Function Trees for Approximation in high Dimensional Spaces," *Proc. 1990, Connectionist Models Summer School*, Morgan Kaufmann, 1990
- [Sanger'91] T. D. Sanger, R. S. Sutton, C. J. Matheus, "Iterative Construction Sparse Polynomial Approximations," *1991 NIPS Conference*, 1991
- [Schalkoff'97] R. J. Schalkoff, *Artificial Neural Networks*, *Mc Graw-Hill*, 1997
- [Serinivasan'94] B. Serinivasan, U. R. Prasad, N. J. Rao, “Back Propagation Through Adjoint for the Identification of Nonlinear Dynamic Systems Using Recurrent Neural Models”, *IEEE Trans. on Neural Networks*, vol. 5, no. 2, pp. 213-227, 1994
- [Shaefer'87] C. G. Shaefer, “The ARGOT strategy: Adaptive representation genetic optimizer technique”, *Genetic Algorithms and Their Applications: Proc. of the Second Intl Conf. on Genetic Algorithms*, pp. 50-58, 1987
- [Sikka'03] D. R. Sikka, "Evaluation of monitoring and forecasting of summer monsoon rainfall over India and a review of monsoon drought of 2002", *Proc. Indian Natl. Sci. Acad.*, vol. 69, pp. 479–504, 2003
- [Singhrattna1'04] N. Singhrattna1, B. Rajagopalan, M. Clark, K. K. Kumar, "Seasonal Forecasting of Thailand Summer Monsoon Rainfall", *International Journal of Climatology*, 2004
- [Soman'05] K. P. Soman, K. I. Ramachandran, *Insight into Wavelets from Theory to Practice*, *Prentice Hall*, 2005
- [Stark'05] H. G. Stark, *Wavelets and Signal Processing; An Application-Based Introduction*, *Springer*, 2005
- [Sugeno'88] M. Sugeno, G. T. Kang, "Structure Identification of Fuzzy Model", *Fuzzy Sets and Systems*, vol. 28, pp. 15-33, 1988
- [Sugeno'93] M. Sugeno, T. Yasukawa, “A Fuzzy logic based approach to qualitative modeling”, *IEEE Trans. on Fuzzy Systems*, vol. 1, no.1, pp.7-31, 1993
- [Takagi'85] T. Takagi, M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control”, *IEEE Tran. on Sys., Man and Cybernetics*, vol. 15, pp.116-132, 1985
- [Ting'99] W. Ting, Y. Sugai, “A wavelet neural network for the approximation of nonlinear multivariable function”, *IEEE Int. Conf. on Systems, Man, and Cybernetics*, vol. 3, pp. 378-383, 1999

- [Xie'87] X. L. Xie, G. Beni, "A Validity Measure for Fuzzy Clustering", *IEEE Trans. on Pattern Anal. Machine Intell.*, vol. 13, no. 8, pp. 841-847, 1987
- [Xu'87] C. W. Xu, Y. Z. Lu, "Fuzzy model Identification and Self-Learning for Dynamic System", *IEEE Trans. On System, Man and Cybernetics*, vol.17, no. 4, pp. 683-689, 1987
- [Wang'97] W. Y. Wang, T. T. Lee, C. L. Liu, C. H. Wang, "Function approximation using fuzzy neural networks with robust learning algorithm", *IEEE Trans. Syst., Man, Cybern. B*, vol. 27, pp. 740-747, 1997
- [Wang'05] D. Y. Wang, H. C. Chuang, Y. J. Xu, C. J. Lin, "A Novel Evolution Learning for Recurrent Wavelet-Based Neuro-Fuzzy Networks", *IEEE Int. Conf. Fuzzy Systems*, pp. 1092-1096, 2005
- [Wang'06] Z. Wang, H. Peng, J. Wang, "Research for a Dynamic Recurrent Fuzzy Wavelet Network", *Sixth Int. Conf. on Intelligent Systems Design and Applications*, 2006
- [Werbos'88] P. Werbos, "Generalization of Backpropagation with Application to a Recurrent Gas Markov Model", *Neural networks*, vol. 1, pp. 339-356, 1988
- [Whitley'95] D. Whitley, "Modeling hybrid genetic algorithms", In G. Winter, J. P'eriaux, M. Gal'an, and P. Cuesta (Eds.), *Genetic algorithms in engineering and computer science*, pp. 191-201, Chichester John Wiley, 1995
- [Williams'89] R. J. Williams, D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Networks", *Neural Computation*, vol. 1, pp. 270-280, 1989
- [Wu'00] S. Wu, M. J. Er, "Dynamic fuzzy neural networks- a novel approach to function approximation", *IEEE Trans. Syst., Man, Cybern. B*, vol. 30, pp. 358-364, 2000
- [vinod'06] V. Kumar, Wavelet Based Adaptive Fuzzy Inference System and its Application, M.Tech. Thesis, Dept. of Elec. Eng., Z.H. College of Eng. & Tech., Aligarh Muslim University, Aligarh, India, 2006
- [Yabuta'91] T. Yabuta, T. Yamada, "Learning Control using neural Networks", *IEEE Int. Conf. Robot. Automat.*, Sacramento, CA, pp. 740-745, 1991
- [Yager'94] R. Yager, D. Filev, "Generation of fuzzy rules by mountain clustering", *J. Intelligent Fuzzy Systems*, vol. 2, pp. 209-219, 1994
- [Yamakawa'94] T. Yamakawa, E. Uchino, T. Samatsu, "Wavelet Neural Networks Employing Over-Complete Number of Compactly Supported Non-Orthogonal Wavelets and Their Applications", *Neural Networks, IEEE World Congress on Computational Intelligence*, vol. 3, pp. 1391-1396, 1994



- [Yi'01] Z. Yi, P. A. Heng, K. S. Leung, "Convergence analysis of cellular neural networks with unbounded delay", *IEEE Trans. Circuits Syst. I*, vol. 48, no. 6, pp. 680–687, 2001
- [Yi'06] Z. Yi, J. C. Lv, L. Zhang, "Output Convergence Analysis for a Class of Delayed Recurrent Neural Networks With Time-Varying Inputs", *IEEE Trans. Syst., Man, Cybern. B*, vol. 36, no. 1, pp. 87-95, 2006
- [Yoo'06] S. J. Yoo, Y. H. Choi, J. B. Park, "Generalized Predictive Control Based on Self-Recurrent Wavelet Neural Network for Stable Path Tracking of Mobile Robots: Adaptive Learning Rates Approach", *IEEE Trans. on Circuits and systems*, vol. 53, no. 6, pp. 1381-1394, 2006
- [Yu'95a] X. H. Yu et al., "Dynamic learning rate optimization of the backpropagation algorithm", *IEEE Trans. Neural Networks*, vol. 6, pp. 669-677, 1995
- [Yu'95b] W. Yu, X. Li, "Fuzzy Identification Using Fuzzy Neural Networks With Stable Learning Algorithms", *IEEE Trans. on Fuzzy Systems*, vol. 12, no. 3, pp. 411-420, 1995
- [Yu'01a] W. Yu, X. Li, "Some stability properties of dynamic neural networks", *IEEE Trans. Circuits Syst. I*, vol. 48, pp. 256-259, 2001
- [Yu'01b] W. Yu, X. Li, "Some new results on system identification with dynamic neural networks", *IEEE Trans. Neural Networks*, vol. 12, pp. 412-417, 2001
- [Zhang'92] Q. Zhang, A. Benveniste, "Wavelet networks", *IEEE Transactions on Neural Networks*, vol. 3, issue 6, pp. 889 – 898, 1992
- [Zhang'95] J. Zhang, G. G. Walter, Y. Miao, W. Lee, "Wavelet neural networks for function learning", *IEEE Transactions on Signal Processing*, vol. 43, issue 6, pp. 1485-1497, 1995
- [Zhang'97] Q. Zhang, "Using Wavelet Network in Nonparametric Estimation", *IEEE Trans. on Neural Networks*, vol. 8, no. 2, pp. 227–236, 1997
- [Zeng'94] X. J. Zeng, M. S. Singh, "Approximation Theory of Fuzzy Systems-SISO Case", *IEEE Trans. on Fuzzy Systems*, vol. 2, no. 2, pp. 162-176, 1994
- [Zeng'95] X. J. Zeng, M. S. Singh, "Approximation Theory of Fuzzy Systems-MIMO Case", *IEEE Trans. on Fuzzy Systems*, vol. 3, no. 2, pp. 162-176, 1995

## Bio-Data

Name : Ahmad Banakar  
Date of Birth : 14<sup>th</sup>, April, 1977

### Academic Qualifications:

1. B. Tech. degree in Electronic, Department of Electrical Engineering, Shiraz University, Iran, 1999
2. M. Tech. degree in Control, Department of Electrical Engineering, Ferdusi Mashad University, Iran, 2002

### Research Publications during the Ph.D. Program

#### (a) Conference Papers

1. M. F. Azeem, Ahmad Banakar, "Recurrent Sigmoid-Wavelet Neurons for Forecasting of Dynamic Systems", *IEEE International Conference. on Information Reuse and Integration*, Hilton Hotel, Las Vegas, USA, August 2007, pp. 556-562.
2. Ahmad Banakar, M. F. Azeem, "A Comparison Study of Different Types of Non-Orthogonal Wavelet in Neuro-Fuzzy Model based on Time Series Forecasting", *International Conference on Intelligent Systems & Networks IISN-2007*, COMPUTATIONAL INTELLIGENCE LAB (CI-LAB), SOCIETY FOR EDUCATION AND RESEARCH (SER), Jagadhri-135003, Distt. Yamuna Nagar, Haryana (INDIA), February 2007
3. Ahmad Banakar, M. F. Azeem, "Parameter Estimation of Neuro-Fuzzy Model by Parallel and Series-Parallel Identification Configurations", *FUZZ-IEEE 2007*, London, UK, July 2007
4. M.F. Azeem, Ahmad Banakar, Vinod Kumar, "Comparative Study of Different Types of Wavelet Functions in Neural Network", *International Joint Conference Neural Networks (IJCNN2006)*, Vancouver, BC, Canada, July 2006, pp. 1061-1066.
5. Ahmad Banakar, M. F. Azeem, "Generalized Wavelet Neural Network Model and its Application in Time Series Prediction", *International Joint Conference Neural Networks (IJCNN2006)*, Vancouver, BC, Canada, July 2006, pp. 882-886.
6. Ahmad Banakar, M. F. Azeem, "Generalized Wavelet Neuro-Fuzzy Model and its Application in Time Series Forecasting", *2nd Symposium on Evolving Fuzzy Systems*, Ambelside, Lake Discript, UK , September 2006, pp. 253-258.
7. Ahmad Banakar, M.F. Azeem, " Input Selection Based on TSK Fuzzy Model and Modified Mountain Clustering", *3rd IEEE Conference on Intelligent Systems*, University of Westminster, London, UK, September 2006, pp. 295-299.

8. Ahmad Banakar, M.F. Azeem, "Identification and prediction of nonlinear dynamical plants Based on TSK fuzzy model and Wavelet Neuro-Fuzzy model", *3rd IEEE Conference on Intelligent Systems*, University of Westminster, London, UK, September 2006, pp. 617-620.
9. Ahmad Banakar, M.F. Azeem, "A New Artificial Neural Network and its Application in Wavelet Neural Network and Wavelet Neuro-Fuzzy Case study: Time Series Prediction", *3rd IEEE Conference on Intelligent Systems*, University of Westminster, London, UK, September 2006, pp. 621-625.
10. Ahmad Banakar, M.F. Azeem, "Comparative Study of Different Type of Wavelet in Artificial Wavelet Neuro-Fuzzy Model", *2006 IEEE Mountain Workshop on Adaptive and Learning Systems (SMCals/06)*, USA, July 2006, pp.165-170.

**(b) Papers Communicated**

- 1- Ahmad Banakar, M.F. Azeem, Vinod Kumar, "Comparative Study of Wavelet Based Neural Network and Neuro-Fuzzy Systems", *International Journal of Wavelets, Multiresolution and Information Processing, World Scientific* (Accepted, November 2007).
- 2- Ahmad Banakar, M. F. Azeem, "Artificial Wavelet Neuro-Fuzzy Model based on parallel Wavelet Network and Neural Network", *Journal of Soft Computing - Springer*, (Accepted)
- 3- Ahmad Banakar, M. F. Azeem, "Artificial Wavelet Neural Network and Its Application in Neuro-Fuzzy Models", *Applied Soft Computing - Elsevier*, (Revised Accepted)
- 4- Ahmad Banakar, M. F. Azeem, "Recurrent Sigmoid-Wavelet Neurons in Feedforward Network for Identification of Dynamic Systems", *IEEE Transaction on Neural Network*, (Revised Accepted)
- 5- Ahmad Banakar, M. F. Azeem, "A Comparative Study of Parameter Identification Configurations for TSK Neuro-Fuzzy Model of Dynamic Systems", *IEEE Transactions on Fuzzy Systems* (Submitted)
- 6- Ahmad Banakar, M. F. Azeem, "Local Recurrent Sigmoidal-Wavelet Neurons in Feed-Forward Neural Network for Forecasting of Dynamic Systems", *IEEE Transaction on System, Cybernetics, and Man Part-B* (Submitted)
- 7- Ahmad Banakar, M. F. Azeem, "Wavelet Neuro-Fuzzy Model Based on Sigmoid-Wavelet Neuron", *Applied Soft Computing - Elsevier* (Ready for submission)