

# **ARTIFICIAL IMMUNE SYSTEMS: PRINCIPLE, ALGORITHMS AND APPLICATIONS**

A THESIS SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF  
**Master of Technology (Research)**  
In  
**Electronics and Communication Engineering**

BY

**SATYASAI JAGANNATH NANDA**

Under the guidance of

**Prof. GANAPATI PANDA, FNAE, FNASc.**



*ELECTRONICS AND COMMUNICATION ENGINEERING  
NATIONAL INSTITUTE OF TECHNOLOGY  
ROURKELA  
INDIA*

# CERTIFICATE

*This is to certify that the thesis entitled “ Artificial Immune Systems : Principle, Algorithms and Applications” by Mr. Satyasai Jagannath Nanda, submitted to the National Institute of Technology, Rourkela for the award of Master of Technology (Research) in Electronics and Communication Engineering , is a record of bona fide research work carried out by him in the department of Electronics and Communication Engineering, National Institute of Technology, Rourkela under my supervision. I believe that this thesis fulfills part of the requirements for the award of degree of Master of Technology (Research). The results embodied in the thesis have not been submitted for the award of any other degree.*

*Prof. G. Panda, FNAE, FNASc.*

*Department of ECE  
National Institute of Technology  
Rourkela- 769008*

# ACKNOWLEDGEMENT

I take the opportunity to express my reverence to my supervisor Prof. G. Panda for his guidance, inspiration and innovative technical discussions during the course of this work. He is not only a great teacher with deep vision but also a very kind person. His trust and support inspired me for taking right decisions and I am glad to work with him.

I thank all my teachers Prof. S.K. Patra, Prof. G.S. Rath, Prof. K. K. Mahapatra and Prof. S.K. Meher for their contribution in my studies and research work. They have been great sources of inspiration to me and I thank them from the bottom of my heart.

I am grateful to my seniors Prof. Ajit Ku. Sahoo , Debidutta Mohanty, Prof. D. P. Acharya and Babita Majhi for their help and support in my research. It's my pleasure to show my indebtedness to my friends like Upendra, Premlatha, Jiju, Ayeskant, Jitendra sir, Pyari Mohan, Nithin, Vikas, Pavan and Sasmita for their help during the course of this work. Special thanks to Sitanshu Sekhar Sahu and Mamata Panigrahy for infallible motivation and moral support.

I can't represent my words to speak of the inspirations from my father and mother. Love and support from Kaka, Khudi, Mamu, Mai, Dada and Mama has been instrumental in completing this task. Wishes from Lisu, Tikili, Kunu, Sifan and Suvam are the key to my steps towards success.

I am also thankful to my classmates and all those who made my stay in Rourkela an unforgettable and rewarding experience.

Satyasai Jagannath Nanda

# CONTENTS

ABSTRACT	v
LIST OF FIGURES	vi
LIST OF TABLES	ix
ACRONYMS	x

## Chapter 1. INTRODUCTION

1.1	Background	1
1.2	Motivation	3
1.3	Present Work	3
1.4	Chapter wise contribution	4
1.5	Publications from this thesis	7

## Chapter 2. BASIC ARCHITETURES AND EVOLUTIONARY COMPUTING

### TECHNIQUES USED IN THE THESIS

2.1	Introduction	9
2.2	Model Structures	9
2.2.1	Adaptive Linear Combiner	9
2.2.2	Artificial Neural Network (ANN)	11
2.2.2.1	Single Neuron Structure	12
2.2.2.2	Multi Layer Perceptron (MLP)	14
2.2.2.3	Functional-link Artificial Neural Network (FLANN)	16
2.3	Learning Algorithms of Various Models	18
2.3.1	Gradient Based Adaptive Algorithms	18
2.3.1.1	Least Mean Square (LMS) Algorithm	18
2.3.1.2	Recursive Least Square (RLS) Algorithm	20
2.3.1.3	Back propagation (BP) Algorithm	22
2.3.1.4	The FLANN Algorithm	24
2.3.2	Evolutionary Computing based Algorithms	25
2.3.2.1	Genetic Algorithm (GA)	25
2.3.2.1.1	Basic Principles of GA	25
2.3.2.1.2	Operators of GA	27
2.3.2.1.3	Parameters of GA	31

2.3.2.2	Particle Swarm Optimization (PSO)	32
2.3.2.2.1	Basic concept of PSO	32
2.3.2.2.2	Particle swarm optimization algorithm	32
2.4	SUMMARY	34

## **Chapter 3. PRINCIPLES AND THEORY OF ARTIFICIAL IMMUNE SYSTEMS**

3.1	Artificial Immune Systems	35
3.2	Biological Immune System	35
3.2.1	Anatomic Barrier	36
3.2.2	Innate Immunity	36
3.2.3	Adaptive Immunity	37
3.2.4	BIS Response	39
3.3	Computational aspects of Immune System	40
3.4	Theories of AIS	41
3.4.1	Immune Network Model	42
3.4.2	Negative Selection Algorithm	43
3.4.3	Clonal Selection Algorithm	44
3.4.4	Danger Theory	45
3.5	Recent uses of AIS based Modeling	48
3.6	SUMMERY	49

## **Chapter 4. APPLICATION OF AIS TO NONLINEAR SYSTEM IDENTIFICATION**

4.1	Introduction	50
4.2	Principle of Adaptive System Identification	50
4.3	Classification of Nonlinear Systems	51
4.4	Problems associated with existing gradient based models	53
4.5	Proposed hybrid FLANN-AIS Model	53
4.5.1	Nonlinear SISO Plant Identification	54
4.5.2	Nonlinear MIMO Plant Identification	57
4.6	System Simulation	59
4.6.1	Identification of Nonlinear Static Plants	60

4.6.2	Identification of Nonlinear Dynamic Plants	67
4.6.3	Identification of Nonlinear MIMO Plants	79
4.7	Conclusion	84
4.8	Summary	84

## **Chapter 5. APPLICATION OF AIS TO NONLINEAR CHANNEL EQUALIZATION**

5.1	Introduction	86
5.2	Need of Adaptive Channel Equalization	86
5.3	Background and problems associated with existing digital channel equalizers	86
5.4	Baseband Communication System	87
5.5	Channel Interference	88
5.6	Intersymbol Interference	90
5.7	Proposed AIS based Channel Equalizer	91
5.8	System Simulation	95
5.9	Case study on AIS equalizer	100
5.10	Conclusion	101
5.11	Summary	101

## **Chapter 6. DEVELOPMENT NEW EVOLUTIONARY HYBRID ALGORITHMS AND IT'S APPLICATION TO HAMMERSTEIN MODEL IDENTIFICATION**

6.1	Introduction	107
6.2	Proposed New Evolutionary Hybrid Algorithms	107
6.2.1	Clonal PSO (CPSO)	108
6.2.2	Immunized PSO (IPSO)	109
6.3	Identification of Hammerstein Model	112
6.3.2	FLANN Structure of the Model	113
6.3.3	Weight update of the Model	117
6.3.3.1	Identification algorithm using FLANN structure and PSO based training	117
6.3.3.2	Identification algorithm using FLANN structure and CPSO based training	119
6.3.3.3	Identification algorithm using FLANN structure and IPSO based training	120

<b>6.4</b>	System Simulation	<b>121</b>
<b>6.5</b>	Conclusion	<b>136</b>
<b>6.6</b>	Summary	<b>136</b>

## **Chapter 7. CONCLUSIONS**

<b>7.1</b>	Conclusions	<b>137</b>
<b>7.2</b>	Scope of future work	<b>138</b>

## **REFERENCES**

Chapter 1	<b>139</b>
Chapter 2	<b>140</b>
Chapter 3	<b>142</b>
Chapter 4	<b>147</b>
Chapter 5	<b>149</b>
Chapter 6	<b>152</b>

## **Abstract**

---

The present thesis aims to make an in-depth study of adaptive identification, digital channel equalization, functional link artificial neural network (FLANN) and Artificial Immune Systems (AIS). Two learning algorithms CPSO and IPSO are also developed in this thesis. These new algorithms are employed to train the weights of a low complexity FLANN structure by way of minimizing the squared error cost function of the hybrid model. These new models are applied for adaptive identification of complex nonlinear dynamic plants and equalization of nonlinear digital channel. Investigation has been made for identification of complex Hammerstein models.

To validate the performance of these new models simulation study is carried out using benchmark complex plants and nonlinear channels. The results of simulation are compared with those obtained with FLANN-GA, FLANN-PSO and MLP-BP based hybrid approaches. Improved identification and equalization performance of the proposed method have been observed in all cases.



# LIST OF FIGURES

## Chapter- 2

<b>Fig 2.1</b>	Adaptive linear Combiner	<b>10</b>
<b>Fig 2.2</b>	Structure of a single neuron	<b>12</b>
<b>Fig 2.3</b>	MLP Structure	<b>15</b>
<b>Fig 2.4</b>	Structure of the FLANN model	<b>17</b>
<b>Fig 2.5</b>	Adaptive filter using LMS algorithm	<b>19</b>
<b>Fig 2.6</b>	Neural network using BP algorithm	<b>22</b>
<b>Fig 2.7</b>	A GA iteration cycle	<b>26</b>
<b>Fig 2.8</b>	Chromosome	<b>27</b>
<b>Fig 2.9</b>	Biased roulette-wheel that is used in the selection of the mating pool	<b>28</b>
<b>Fig 2.10</b>	Single point Crossover	<b>29</b>
<b>Fig 2.11</b>	Double point Crossover	<b>30</b>
<b>Fig 2.12</b>	Mutation	<b>30</b>
<b>Fig 2.13</b>	Representation of PSO algorithm	<b>32</b>

## Chapter- 3

<b>Fig 3.1</b>	Response of BIS to antigens	<b>40</b>
<b>Fig 3.2</b>	Presence of paratope and idiotope on antibody	<b>42</b>
<b>Fig 3.3</b>	Basic of Negative Selection Algorithm	<b>44</b>
<b>Fig 3.4</b>	Basic of Clonal Selection Algorithm	<b>45</b>
<b>Fig 3.5</b>	Principle of Danger Theory	<b>47</b>

## Chapter- 4

<b>Fig 4.1</b>	Block diagram of Nonlinear SISO plant identification	<b>54</b>
<b>Fig 4.2</b>	Block diagram of Nonlinear MIMO plant identification	<b>57</b>
<b>Fig 4.3</b>	Identification of $f_1$	<b>63</b>
<b>Fig 4.4</b>	Identification of $f_2$	<b>64</b>

<b>Fig 4.5</b>	Identification of $f_3$	<b>65</b>
<b>Fig 4.6</b>	Identification of $f_4$	<b>66</b>
<b>Fig 4.7</b>	Proposed FLANN-AIS based model structure for identification of dynamic system of Example-1	<b>68</b>
<b>Fig 4.8</b>	Identification of dynamic system Example-1	<b>69</b>
<b>Fig 4.9</b>	Proposed FLANN-AIS based model structure for identification of dynamic system of Example-2	<b>71</b>
<b>Fig 4.10</b>	Identification of dynamic system Example-2	<b>72</b>
<b>Fig 4.11</b>	Proposed FLANN-AIS based model structure for identification of dynamic system of Example-3	<b>74</b>
<b>Fig 4.12</b>	Identification of dynamic system Example-3	<b>75</b>
<b>Fig 4.13</b>	Proposed FLANN-AIS based model structure for identification of dynamic system of Example-4	<b>77</b>
<b>Fig 4.14</b>	Identification of dynamic system Example-4	<b>78</b>
<b>Fig 4.15</b>	Proposed FLANN-AIS based structure for identification of MIMO Plant	<b>81</b>
<b>Fig 4.16</b>	Identification of MIMO plant Output1	<b>82</b>
<b>Fig 4.17</b>	Identification of MIMO plant Output2	<b>83</b>

## Chapter- 5

<b>Fig 5.1</b>	A baseband Communication System	<b>88</b>
<b>Fig 5.2</b>	Impulse Response of a transmitted signal in a channel which has 3 modes of Propagation	<b>89</b>
<b>Fig 5.3</b>	Interaction between two neighboring symbols	<b>90</b>
<b>Fig 5.4</b>	An AIS based adaptive digital channel equalizer	<b>92</b>
<b>Fig 5.5</b>	BER at 5dB SNR: (a) using $f_1$ (b) using $f_2$ (c) using $f_3$ nonlinearity based channel CH1	<b>96</b>
<b>Fig 5.6</b>	BER at 10dB SNR: (a) using $f_1$ (b) using $f_2$ (c) using $f_3$ nonlinearity based channel CH1	<b>97</b>
<b>Fig 5.7</b>	BER at 5dB SNR: (a) using $f_1$ (b) using $f_2$ (c) using $f_3$ nonlinearity based channel CH2	<b>98</b>

<b>Fig 5.8</b> BER at 10dB SNR: (a) using $f_1$ (b) using $f_2$ (c) using $f_3$ nonlinearity based channel CH2	<b>99</b>
<b>Fig 5.9</b> BER plot of all four case studies	<b>100</b>
<b>Chapter- 6</b>	
<b>Fig 6.1</b> Representation of conventional PSO algorithm	<b>111</b>
<b>Fig 6.2</b> Representation of CPSO algorithm	<b>111</b>
<b>Fig 6.3</b> Representation of IPSO algorithm	<b>112</b>
<b>Fig 6.4</b> The Hammerstein Model	<b>113</b>
<b>Fig 6.5</b> Structure of FLANN Model	<b>114</b>
<b>Fig 6.6</b> Identification structure of Hammerstein model	<b>115</b>
<b>Fig 6.7</b> Identification structure of Hammerstein model for system simulation	<b>122</b>
<b>Fig 6.8</b> True and estimated nonlinear response of static part of the model of Example 1	<b>123</b>
<b>Fig 6.9</b> Response matching of Model and estimated output during testing of Example 1	<b>125</b>
<b>Fig 6.10</b> True and estimated nonlinear response of static part of the model of Example 2	<b>127</b>
<b>Fig 6.11</b> Response matching of Model and estimated output during testing of Example 2	<b>128</b>
<b>Fig 6.12</b> True and estimated nonlinear response of static part of the model of Example 3	<b>130</b>
<b>Fig 6.13</b> Response matching of Model and estimated output during testing of Example 3	<b>132</b>
<b>Fig 6.14</b> True and estimated nonlinear response of static part of the model of Example 4	<b>133</b>
<b>Fig 6.15</b> Response matching of Model and estimated output during testing of Example 4	<b>135</b>

# LIST OF TABLES

## Chapter- 4

<b>TABLE 4.1</b>	Comparative results of static SISO systems obtained through simulation study	<b>62</b>
<b>TABLE 4.2</b>	Comparative results of dynamic systems obtained through simulation study	<b>79</b>
<b>TABLE 4.3</b>	Comparative results of MIMO systems obtained through simulation study	<b>84</b>

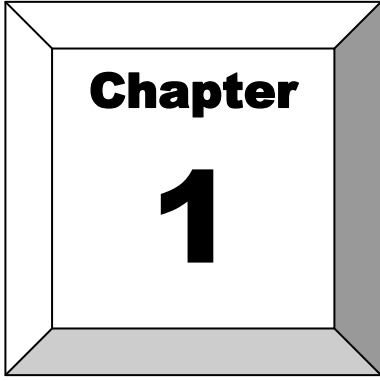
## Chapter- 6

<b>TABLE 6.1</b>	Comparative results of estimates of system parameters for dynamic part of model of Example1	<b>124</b>
<b>TABLE 6.2</b>	Comparative results of CPU time and SSE for model of Example1 obtained through simulation study	<b>124</b>
<b>TABLE 6.3</b>	Comparative results of estimates of system parameters for dynamic part of model of Example2	<b>129</b>
<b>TABLE 6.4</b>	Comparative results of CPU time and SSE for model of Example2 obtained through simulation study	<b>129</b>
<b>TABLE 6.5</b>	Comparative results of estimates of system parameters for dynamic part of model of Example3	<b>131</b>
<b>TABLE 6.6</b>	Comparative results of CPU time and SSE for model of Example3 obtained through simulation study	<b>131</b>
<b>TABLE 6.7</b>	Comparative results of estimates of system parameters for dynamic part of model of Example4	<b>134</b>
<b>TABLE 6.8</b>	Comparative results of CPU time and SSE for model of Example4 obtained through simulation study	<b>134</b>

# ACRONYMS

AIS	Artificial Immune Systems
AMSE	Average MSE
ASDM	Adaptive SDM
AWGN	Additive White Gaussian Noise
ANN	Artificial Neural Network
BER	Bit Error Rate
BFO	Bacterial Foraging Optimization
BGA	Binary Coded Genetic Algorithm
BIS	Biological Immune Systems
BP	Back Propagation
CDF	Cumulative Distribution Function
CPSO	Clonal PSO
CPU	Central Processing Unit
DSP	Digital Signal Processing
DV	Decoded Value
FIR	Finite Impulse Response
FLANN	Functional Link Artificial Neural Network
GA	Genetic Algorithm
GD	Gradient Descent
IIR	Infinite Impulse Response
IPSO	Immunized PSO
ISI	Inter Symbol Interference
LAN	Local Area Network
LMS	Least Mean Square
MLP	Multilayer Perceptron
MSE	Mean square Error
MMSE	Minimum MSE
MIMO	Multiple Input Multiple Output

NN	Neural Network
PDF	Probability Density Function
pH	Power of Hydrogen
PPN	Polynomial Perceptron Network
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
RLS	Recursive Least Square
RV	Real Value
SDM	Single Dimensional Mutation
SI	System Identification
SISO	Single Input Single Output
SSE	Sum of Squared Errors



**Chapter**  
**1**



**INTRODUCTION**

---

## 1.1 Background

**N**ATURE is the main source of inspiration for the development of various computational algorithms and tools. The design prospective of development of computational tools is termed as biologically inspired computing or computing with biological metaphors. Other aspect of using the computer to design various algorithms and to learn more about natural world is named as computationally motivated biology. The understanding of biological immune system (BIS) has increased dramatically over the recent few years by several researchers. It has provided a miraculous insight into how the body resists itself from infectious diseases. Through improved understanding and investigation, new algorithms inspired by BIS are developed under a new branch of computational intelligence known as artificial immune system (AIS) [1.1]-[1.4]. Like human nervous systems and Darwin evolutionary theory the biological immune systems offer a number of attractive features such as ability to remember, classify and neutralize the effect of foreign particles. The focus of this thesis is to develop and introduce new supervised learning paradigms inspired by mechanism found in natural immune systems and to develop methodology to apply these algorithms to effectively solve problems of communication and control such as channel equalization and identification.

System identification is one of the most important areas in control, communication and instrumentation because of its applicability to a wide range of problems. When a practical plant is considered its behavior is completely unknown. It may be nonlinear, dynamic, time varying or chaotic. In general it is difficult to model such complex plants by conventional numerical analysis method where as adaptive system identification is an alternate way of efficient modeling of these plants. In the adaptive method of identification of complex plants and equalization of nonlinear channels a suitable nonlinear architecture such as multilayer perceptron[1.5], functional link artificial neural networks[1.6] or radial basis function neural network[1.7] is selected and its associated weights are trained using one of the evolutionary computing tools like genetic algorithm[1.8], particle swarm



---

optimization[1.9], ant colony optimization[1.10] or bacterial foraging optimization[1.11]. The evolutionary tools as learning algorithm are preferred because they are based on the process of natural selection and do not require error gradient statistics to update the parameters. As a consequence, they are able to find global minimum parameters. Such type of approach offers flexibility, adaptability and versatility so that the identification model can able to meet specific goal and accuracy.

The Hammerstein plant is widely used because its structure reflects the nonlinearity of practical dynamic systems [1.12]. It plays a significant role in stability analysis and design of control systems. The structure of the plant is composed of a nonlinear static block in series with a linear dynamic system block. The identification of such practical plants is useful to determine the system parameters of the model from the known input output data of the plant.

The field of digital data communications has experienced an explosive growth in recent years and its demand reaches at the peak as additional services are being added to existing infrastructure. The telephone networks are originally designed for voice communication but, in recent times, the advances in digital communications using Integrated Service Digital Network (ISDN), data communications with computers, fax, video conferencing etc. have pushed the use of these facilities far beyond the scope of their original intended use. Similarly, introduction of digital cellular radio (DCR) and wireless local area networks (LAN's) have stretched the limited available radio spectrum capacity to the limits it can offer. These advances in digital communications have been made possible by the effective use of the existing communication channels with the aid of signal processing techniques [1.13]. Therefore transmission bandwidth is one of the most precious resources in digital communication systems. Communication channels are usually modeled as band-limited linear finite impulse response (FIR) filters with low pass frequency response. When the amplitude and the envelope delay response are not constant within the bandwidth of the filter, the channel distorts the transmitted signal causing inter-symbol interference (ISI). The addition of noise during propagation also degrades the quality of the received signal.

---

## **1.2 Motivation**

Over the last two decades a lot of research has gone into the development of identification models based on a range of soft computing techniques. Early models employed the MLP architecture using back propagation (BP) algorithm [1.5], while a lot of recent work is based on evolutionary optimization techniques such as GA and PSO. Thus the review of the existing literature reveals that varieties of ANN structures have been employed to develop new identification models. In most cases it is noticed that the development of these models and the testing involve large computational complexity as well as more time to learn the structure. Second issue is the accuracy of identification of existing techniques. Thus there is a need to develop simple but efficient nonlinear adaptive structure which involves less computation and better identification capability particularly for complex, nonlinear, dynamic and Hammerstein plants.

To recover the transmitted data faithfully at the receiver end, the effect ISI needs to be reduced and the additive noise to be suppressed. This is achieved by adaptive channel equalizer. But when the channels are nonlinear and complex, the equalization task becomes difficult [1.14]. Thus there is a need to develop simple but efficient adaptive equalizers which will provide better timing and accurate equalization particularly for complex and nonlinear channels.

## **1.3 Present Work**

The present work describes in brief the theory and principles of artificial immune system. This new concept is suitably applied for identification of complex nonlinear dynamic and Hammerstein plants. Studies on effective equalization of nonlinear channels has also been investigated in this study. The present research work comprises of a novel hybrid approach employing AIS learning tool to a low complexity FLANN for achieving better direct and inverse modeling of complex plants.

---

## **1.4 Chapter wise contribution**

The work carried out in the present thesis is organized in seven chapters.

### **Chapter-1 INTRODUCTION**

### **Chapter-2 BASIC ARCHITECTURES AND EVOLUTIONARY COMPUTING TECHNIQUES USED IN THE THESIS**

Chapter 2 deals with the basic architecture used in the thesis to develop adaptive models for identification of plants and equalization of channels. The structures used are broadly classified into two categories linear (FIR and IIR etc.) and nonlinear (Artificial neural structures like multilayer perceptron, functional link artificial neural network etc.). The basics of these structures are discussed in this chapter. To design adaptive identification and channel equalizer models their parameters need training which are to be carried out by various derivative based and derivative free learning algorithms. The derivative or gradient based algorithms include LMS, RLS, Back Propagation etc. Algorithms based on natural selection or derivative free are genetic algorithm (GA), particle swarm optimization (PSO). The stepwise procedure of each of these learning algorithms is also briefly outlined in this chapter.

### **Chapter-3 PRINCIPLE AND THEORY OF ARTIFICIAL IMMUNE SYSTEMS (AIS)**

The Biological immune system (BIS) is a multilayer protection system where each layer provides different types of defense mechanisms for detection, recognition and responses. Investigation is done on the functionality of BIS that is how the body restricts itself from the invasion of external microorganisms. The artificial immune system (AIS) is developed by following the principle of BIS. The four forms of AIS algorithm reported in the literature are immune network model, negative selection,

---

clonal selection and danger theory. An overview of each of these algorithms is discussed in this chapter.

#### **Chapter-4 APPLICATION OF AIS TO NONLINEAR SYSTEM IDENTIFICATION**

The first contribution of the thesis lies in this chapter. It contains novel contribution of system identification using AIS technique. In this chapter the nonlinear plants are classified into three categories such as static, dynamic and MIMO, depending upon its input-output relationship. The detail architecture of these plants is described. The identification model proposed here consists of a FLANN structure whose weights are trained with CLONAL selection principle of AIS. Simulation study is carried out on some benchmark identification problems. The proposed model is compared with other standard models like MLP with back propagation and FLANN structure with different evolutionary algorithms like GA, PSO etc. The efficiency of identification is determined by comparison of overall output response of the plant and the estimated model, sum of square errors (SSE) and overall CPU time required to train the model. The results are embodied in three different papers [2, 3, 5].

#### **Chapter-5 APPLICATION OF AIS TO NONLINEAR NOISY CHANNEL EQUALIZATION**

The contribution in this chapter is the development of AIS based adaptive channel equalization. Transmission and storing of high density digital information plays an important role in the present age of communication and information technology. These data are distorted while reading out of the recording medium or arriving at the receiver end due to inter symbol interference in the channel. The adaptive channel equalizer alleviates this distortion and reconstructs the transmitted data faithfully. In this chapter, we propose a novel digital channel equalizer using CLONAL selection algorithm of AIS. Simulation study has been carried out to show superior

---

performance of the proposed equalizer particularly for nonlinear noisy channels compared to that offered by LMS and GA based training. The comparative bit error rate plots obtained by simulation of proposed and standard equalizers are embodied in the paper [4].

## **Chapter-6 DEVELOPMENT OF NEW EVOLUTIONARY HYBRID ALGORITHMS AND THEIR APPLICATION TO HAMMERSTEIN MODEL IDENTIFICATION**

In this chapter we propose two new hybrid evolutionary algorithms known as Clonal PSO (CPSO) and Immunized PSO (IPSO) by suitably combining the good features of PSO and AIS algorithms. The details of these two algorithms are outlined. The performance of these new algorithms has been assessed by employing them in identification of various standard Hammerstein models. The Hammerstein model is widely used because its structure effectually reflects the nonlinearity of practical dynamic systems. It finds extensive applications in stability analysis and control design. The nonlinear static part of the model to be estimated is represented by a single layer low complexity nonlinear functional link artificial neural network architecture. The weights of this structure and the dynamic part of the model are estimated by the proposed algorithms. The results obtained in this chapter are embodied in three different papers [1, 6,7].

## **Chapter-7 CONCLUSION AND SCOPE OF FUTURE WORK**

The overall conclusion of the thesis is presented in this chapter. It also contains some future research topics which need attention and further investigation.

---

## **Publications form this thesis**

### **Journals**

#### **(Communicated)**

1. **Satyasai Jagannath Nanda, G. Panda and Babita Majhi** , “Improved Identification of Hammerstein Models Using New Clonal PSO and Immunized PSO Algorithms”, submitted to **IEEE Trans. On System Man and Cybernetics, Part-B**, October, 2008.
2. **Satyasai Jagannath Nanda, G. Panda and Babita Majhi**, “Improved Identification of Nonlinear Plants using Artificial Immune System based FLANN model”, submitted to **Engg. Application of Artificial Intelligence, Elsevier, UK**, November. 2008.

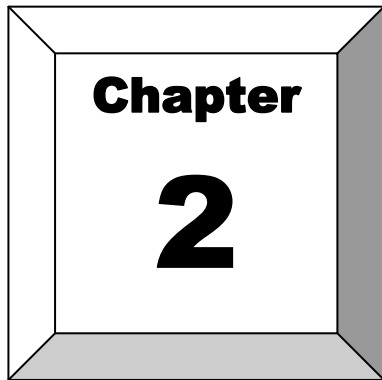
### **International Conferences**

3. **Satyasai Jagannath Nanda, G. Panda and Babita Majhi**, “Improved Identification of Nonlinear Dynamic systems using Artificial Immune system”, **IEEE international conference on Control, Communication and Automation (INDICON-08)**, IIT Kanpur, pp. 268-273, December 2008.
4. **Satyasai Jagannath Nanda, G. Panda and Babita Majhi**, “Development of Novel Digital Equalizers for Noisy Nonlinear Channel Using Artificial Immune System”, **3<sup>rd</sup> IEEE international conference on Industrial and information Systems (ICIIS-08)**, IIT Kharagpur, December 2008
5. **Satyasai Jagannath Nanda, G. Panda, Babita Majhi and Prakash Tha**, “Development of a New Optimization Algorithm based on Artificial Immune System and Its Application”, **IEEE international conference on Information Technology (ICIT-08)**, XIMB Bhubaneswar, pp.45-48, December 2008.
6. **Satyasai Jagannath Nanda, G. Panda and Babita Majhi**, “Improved Identification of Hammerstein Model based on Artificial Immune System”, **IEEE international conference on Emerging Trends in Computing (ICETiC-09)**, Kamaraj College of Engg. & Tech., Tamil Nadu, pp.193-198, January 2009.

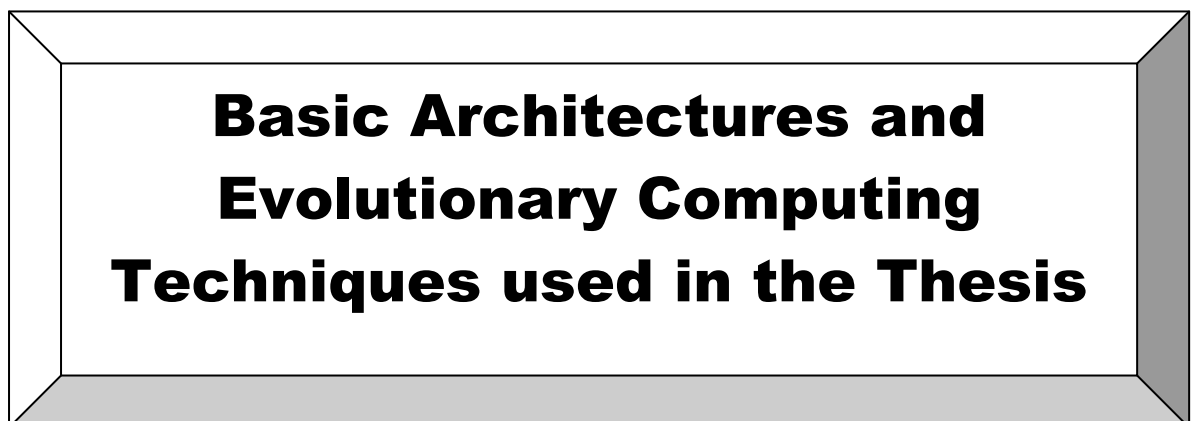
#### **(Communicated)**

7. **Satyasai Jagannath Nanda, G. Panda and Babita Majhi**, “Development of Immunized PSO Algorithm and Its Application to Hammerstein Model Identification,” **IEEE Congress on Evolutionary Computation (CEC- 09)**, Trondheim, Norway, 18<sup>th</sup> - 21<sup>th</sup> May 2009.

- 
8. **Satyasai Jagannath Nanda, G. Panda and Babita Majhi, “Improved Identification of Nonlinear MIMO Plants using New Hybrid FLANN-AIS Model,” IEEE International Advance Computing Conference (IACC- 09), Thapar University, Patiala, Punjab, India, 6<sup>th</sup> - 7<sup>th</sup> March 2009.**



**Chapter**  
**2**



**Basic Architectures and  
Evolutionary Computing  
Techniques used in the Thesis**



---

## 2.1 Introduction

**A**DAPTIVE modeling essentially consists of two important components. The most important one is the structure of the model which is basically selected as a linear combiner, a multilayer perceptron (MLP) structure or a Functional Link Artificial Neural Network (FLANN). These structures contain connecting weights which are trained by various derivative based and derivative free learning algorithms. The derivative or gradient based algorithms include least means square (LMS), recursive least square (RLS), back propagation (BP) etc. Algorithms based on natural selection or derivative free are Genetic algorithm (GA), Particle swarm optimization (PSO) etc.. In this chapter the basics of various structures used and their corresponding learning algorithm adopted to design the adaptive identification and channel equalizer models are discussed in brief.

## 2.2 Model Structures

In this subsection a brief description of each of these structures is outlined. The key equations which govern the function of the structures are also presented.

### 2.2.1 Adaptive Linear Combiner

An adaptive linear combiner is a computational device that attempts to model the relationship between two signals in real time in an iterative manner. The general form of an adaptive linear combiner [2.1, 2.2] is shown in Fig 2.1. There is an input signal vector with elements  $x_0, x_1 \dots x_L$ , a corresponding set of adjustable weights  $w_0, w_1 \dots w_L$ , a summing unit, and a single output signal,  $y$ . A procedure for adjusting or adopting the weights is called weight adjustment or adaptation procedure. The combiner is called linear because for fix setting of weights its output is a linear combination of the input components. The main aspects of the model development of a adaptive liner combiner is

- (i) It is used as a multiple input and single output device.

(ii) The output of the structure is computed from its input signal.

(iii) The parameters within the structure are changed iteratively to alter the input output relationship of the device.

At any  $k^{\text{th}}$  instant the multiple input is represented as  $X_k$ . The output of the combiner can be represented as

$$y_k = \sum_{i=0}^{L-1} w_{ik} X_{k-i} \quad (2.1)$$

where  $w_{ik}$  denotes  $i^{\text{th}}$  weight at  $k^{\text{th}}$  instant.

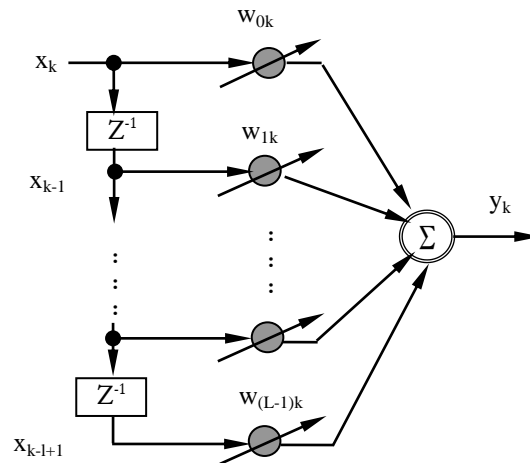


Fig. 2.1 Adaptive linear combiner

If the weight and input vectors are expressed as

$$X_k = [x_{0k} \quad x_{1k} \cdots x_{Lk}]^T \quad (2.2)$$

$$W_k = [w_{0k} \quad w_{1k} \cdots w_{Lk}]^T \quad (2.3)$$

then the output is given by

$$y_k = X_k^T W_k \quad (2.4)$$

---

The weights of the combiner are to be updated using various learning algorithms such as the LMS or the RLS.

## **2.2.2 Artificial Neural Network (ANN)**

In past few decades the Artificial neural network (ANN) has emerged as a powerful learning tool to perform complex tasks in highly nonlinear dynamic environments. There are extensive applications of various types of ANN in the field of communication, control, instrumentation and forecasting. The tool takes its name from the network of nerve cells in the brain. ANN has been found to be an important technique for classification and optimization problem [2.3-2.6]. McCulloch and Pitts have developed the neural networks for different computing machines. The ANN is capable of performing nonlinear mapping between the input and output space due to its large parallel interconnection between different layers and the nonlinear processing characteristics. An artificial neuron basically consists of a computing element that performs the weighted sum of the input signal and the connecting weight. The sum is added with the bias or threshold and the resultant signal is then passed through a nonlinear function of sigmoid or hyperbolic tangent type. Each neuron is associated with three parameters whose learning can be adjusted; these are the connecting weights, the bias and the slope of the nonlinear function. For the structural point of view a NN may be single layer or it may be multilayer. In multilayer structure, there is one or many artificial neurons in each layer and for a practical case there may be a number of layers. Each neuron of the one layer is connected to each and every neuron of the next layer. The functional-link ANN is another type of single layer NN. In this type of network the input data is allowed to pass through a functional expansion block where the input data are nonlinearly mapped to more number of points. This is achieved by using trigonometric functions, tensor products or power terms of the input. The output of the functional expansion is then passed through a single neuron.

The learning of the NN may be supervised in the presence of the desired signal or it may be unsupervised when the desired signal is not accessible. Rumelhart developed the Back-propagation (BP) algorithm, which is central to much work on supervised learning in MLP [2.3]. A feed-forward structure with input, output, hidden layers and nonlinear sigmoid functions are used in this type of network. In recent years many different types of learning algorithm using the incremental back-propagation algorithm [2.11], evolutionary learning using the nearest neighbor MLP [2.12] and a fast learning algorithm based on the layer-by-layer optimization procedure [2.13] are suggested in literature. In case of unsupervised learning the input vectors are classified into different clusters such that elements of a cluster are similar to each other in some sense. The method is called competitive learning [2.14], because during learning sets of hidden units compete with each other to become active and perform the weight change. The winning unit increases its weights on those links with high input values and decreases them on those with low input values. This process allows the winning unit to be selective to some input values. Different types of NNs and their learning algorithms are discussed in sequel.

### 2.2.2.1 Single Neuron Structure

In 1958 Rosenblatt demonstrated the use of perceptron [2.5]. The perceptron is a single level connection of neurons sometimes known as single layer feed forward network. The basic structure of an artificial neuron is presented in Fig. 2.2. The operation in a neuron involves the computation of the weighted sum of inputs and threshold [2.3-2.6]. The resultant signal is then passed through a nonlinear activation function.

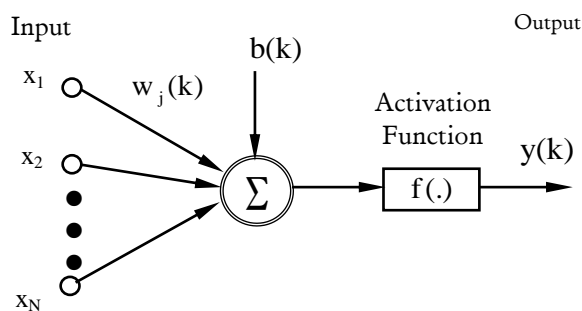


Fig. 2.2. Structure of a single neuron

---

The output of the neuron may be represented as,

$$y(k) = f \left[ \sum_{j=1}^N w_j(k)x_j(k) + b(k) \right] \quad (2.5)$$

where  $b(k)$  is the threshold to the neurons at the first layer,  $w_j(k)$  is the weight associated with the  $j^{\text{th}}$  input,  $N$  is the no. of inputs to the neuron and  $f(\cdot)$  is the nonlinear activation function. Different types of nonlinear function are described as follows

**Signum Function:** For this type of activation function, we have

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} \quad (2.6)$$

**Threshold Function:** This function is represented as,

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (2.7)$$

**Sigmoid Function:** This function is S-shaped, is the most common form of the activation function used in artificial neural network. It is a function that exhibits a graceful balance between linear and nonlinear behaviour.

$$f(x) = \frac{1}{1 + e^{-ax}} \quad , a > 0 \quad (2.8)$$

where  $x$  is the input to the sigmoid function and  $a$  is the slope of the sigmoid function. For the steady convergence a proper choice of  $a$  is required.

---

**Piecewise-Linear Function:** This function is

$$f(x) = \begin{cases} 1, & \text{if } x \geq +0.5 \\ x, & \text{if } +0.5 > x > -0.5 \\ 0, & \text{if } x \leq -0.5 \end{cases} \quad (2.9)$$

where the amplification factor inside the linear region of operation is assumed to be unity. This can be viewed as an approximation to a nonlinear amplifier.

**Hyperbolic Tangent Function:** This function is represented as

$$f(x) = \tanh(\gamma x) = \frac{1 - e^{-\gamma x}}{1 + e^{-\gamma x}}, \quad \gamma > 0 \quad (2.10)$$

where  $x$  is the input to the hyperbolic function.

### 2.2.2.2 Multi Layer Perceptron (MLP)

In the multilayer neural network or multilayer perceptron (MLP), the input signal propagates through the network in a forward direction, on a layer-by-layer basis. This network has been applied successfully to solve some difficult and diverse problems by training in a supervised manner with a highly popular algorithm known as the error back-propagation algorithm [2.3,2.4]. The scheme of MLP using four layers is shown in Fig. 2.3.  $x_p(k)$  represent the input to the network,  $\phi_q$  and  $\phi_r$  represent the output of the two hidden layers and  $y_s(k)$  represents the output of the final layer of the neural network. The connecting weights between the input to the first hidden layer, first to second hidden layer and the second hidden layer to the output layers are represented by  $w_{pq}$ ,  $w_{qr}$  and  $w_{rs}$  respectively.

If  $P_1$  is the number of neurons in the first hidden layer, each element of the output vector of first hidden layer may be calculated as,

$$\phi_q = f_q \left[ \sum_{p=1}^N w_{pq} x_p + b_q \right], \quad p = 1, 2, 3, \dots, N, \quad q = 1, 2, 3, \dots, P_1 \quad (2.11)$$

where  $b_q$  is the threshold to the neurons of the first hidden layer,  $N$  is the no. of inputs and  $f(\cdot)$  is the nonlinear activation function in the first hidden layer chosen from (2.6)-(2.10). The time index 'k' has been dropped to make the equations simpler.

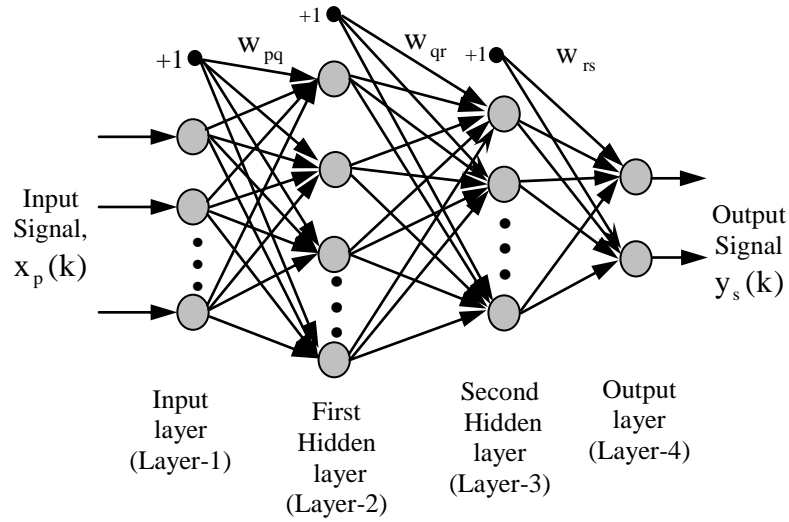


Fig. 2.3 MLP Structure

Let  $P_2$  be the number of neurons in the second hidden layer. The output of this layer is represented as,  $\phi_r$  and may be written as

$$\phi_r = f_r \left[ \sum_{q=1}^{P_1} w_{qr} \phi_j + b_r \right], \quad r = 1, 2, 3, \dots, P_2 \quad (2.12)$$

where,  $b_r$  is the threshold to the neurons of the second hidden layer. The output of the final output layer can be calculated as

$$y_s = f_s \left[ \sum_{r=1}^{P_2} w_{rs} \phi_r + b_s \right], \quad s = 1, 2, 3, \dots, P_3 \quad (2.13)$$

where,  $b_s$  is the threshold to the neuron of the final layer and  $P_3$  is the no. of neurons in the output layer. The output of the MLP may be expressed as

$$y_s = f_s \left[ \sum_{r=1}^{P_2} w_{rs} f_r \left( \sum_{q=1}^{P_1} w_{qr} f_q \left\{ \sum_{p=1}^N w_{pq} x_p + b_q \right\} + b_r \right) + b_s \right] \quad (2.14)$$

### 2.2.2.3 Functional-link Artificial Neural Network (FLANN)

Pao originally proposed FLANN and it is a novel single layer ANN structure capable of forming arbitrarily complex decision regions by generating nonlinear decision boundaries [2.8]-[2.10]. Here, the initial representation of a pattern is enhanced by using nonlinear function and thus the pattern dimension space is increased. The functional link acts on an element of a pattern or entire pattern itself by generating a set of linearly independent function and then evaluates these functions with the pattern as the argument. Hence separation of the patterns becomes possible in the enhanced space. The use of FLANN not only increases the learning rate but also has less computational complexity [2.15]. Pao *et al* [2.10] have investigated the learning and generalization characteristics of a random vector FLANN and compared with those attainable with MLP structure trained with back propagation algorithm by taking few functional approximation problems. A FLANN structure is shown in Fig. 2.4.

Let  $\mathbf{X}$  is the input vector of size  $N \times 1$  which represents  $N$  number of elements; the  $k^{\text{th}}$  element is given by

$$X(k) = x(k), 1 \leq k \leq N \quad (2.15)$$

Each element undergoes nonlinear expansion to form  $M$  elements such that the resultant matrix has the dimension of  $N \times M$ .

The functional expansion of the element  $x_n$  by power series expansion is carried out using the equation given in (2.16)



$$s_i(k) = \begin{cases} 1 & \text{for } i = 0 \\ x(k) & \text{for } i = 1 \\ x^i(k) & \text{for } i = 2,3,4\dots M+1 \end{cases} \quad (2.16)$$

For trigonometric expansion, the expanded elements are

$$s_i(k) = \begin{cases} 1 & \text{for } i = 0 \\ x(k) & \text{for } i = 1 \\ \sin(i\pi x(k)) & \text{for } i = 2,4\dots M \\ \cos(i\pi x(k)) & \text{for } i = 3,5\dots M+1 \end{cases} \quad (2.17)$$

where  $i = 1,2,\dots, M/2$ . The bias input is unity. So total expanded values including the bias becomes  $Q = M+2$ .

Let the weight vector is represented as  $\mathbf{W}$  having  $Q$  elements. The output  $y$  is given as

$$y(k) = \sum_{i=1}^Q s_i(k)w_i(k) \quad (2.18)$$

In matrix notation the output can be,

$$\mathbf{Y} = \mathbf{S} \cdot \mathbf{W}^T \quad (2.19)$$

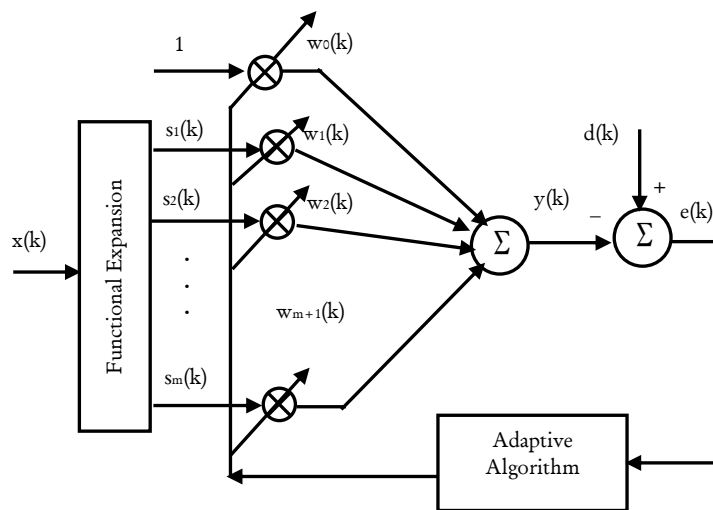


Fig. 2.4 Structure of the FLANN model

---

## 2.3 Learning Algorithms of Various Models

There are varieties of learning algorithms which are employed to train various adaptive models. The performance of these models depends on the rate of convergence, the training time, computational complexity involved and minimum mean square error achieved after training. The learning algorithms may be broadly classified into two categories a) Gradient based b) Evolutionary computing based algorithms. The gradient based adaptive algorithms include least means square (LMS), recursive least square (RLS), back propagation (BP), FLANN. Under evolutionary computing based algorithms we have employed genetic algorithm (GA) and particle swarm optimization (PSO). In this section the details of these algorithms are outlined.

### 2.3.1 Gradient Based Adaptive Algorithms

These types of algorithms are gradient search in nature and have been derived by taking the derivative of the squared error. During the process of training these algorithms tend to optimize the weights of the model. They are expressed in close form equations and are simple to implement. A brief description of each of them is presented below.

#### 2.3.1.1 Least Mean Square (LMS) Algorithm

The general architecture of the LMS based adaptive filter is depicted in Fig. 2.7. Let  $X$  is  $N^{\text{th}}$  input pattern having one unit delay in each instant. This process is also called as adaptive linear combiner [2.1, 2.2]. Let  $X_k = [x_k \ x_{k-1} \ \dots \ x_{k-L+1}]^T$  form of the  $L$ -by-1 tap input vector where  $L-1$  is the number of delay elements. Correspondingly, the tap weights  $W_k = [w_{0k} \ w_{1k} \ \dots \ w_{(L-1)k}]^T$  form the elements of the  $L$ -by-1 tap weight vector. The output is represented as,

$$y_k = \sum_{i=0}^{L-1} w_{ik} X_{k-i} \quad (2.20)$$

The output can be represented in vector notation as

$$y_k = \mathbf{X}_k^T \mathbf{W}_k = \mathbf{W}_k^T \mathbf{X}_k \quad (2.21)$$

Generally for the adaptive linear combiner the other data include a “desired response” or “training signal”,  $d_k$ . This is accomplished by comparing the output with the desired response to obtain an “error signal”  $e_k$  and then adjusting or optimizing the weight vector to minimize this signal. The error signal is,

$$e_k = d_k - y_k \quad (2.22)$$

The weights associated with the network are then updated using the LMS algorithm [2.1]. The weight vector can be updated by taking the derivative of the cost function which is the square of error defined in (2.22).

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \Delta \mathbf{W}_k \quad (2.23)$$

The change in weight at  $k$ th instant is  $\Delta \mathbf{W}_k$  where

$$\Delta \mathbf{W}_k = 2 \cdot \mu \cdot e_k \cdot \mathbf{X}_k \quad (2.24)$$

where  $\mu$  is the learning rate parameter ( $0 \leq \mu \leq 1$ ).

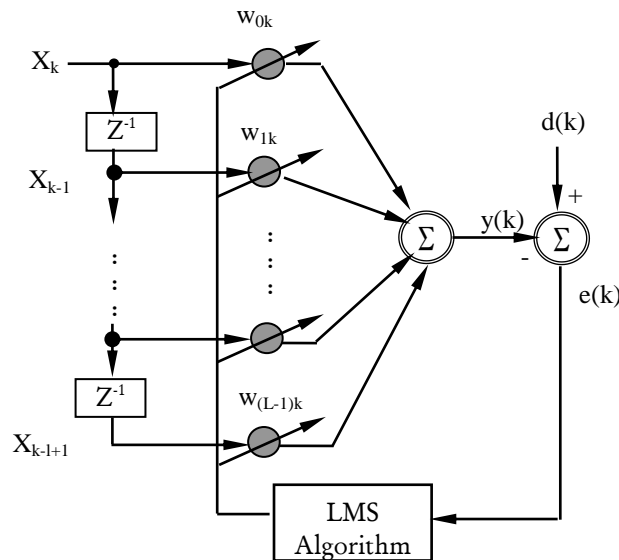


Fig. 2.5 Adaptive filter using LMS algorithm

---

This procedure is repeated till the Mean Square Error (MSE) of the network approaches a minimum value. The MSE at the time index 'k' may be defined as,  $\xi = E[e_k^2]$ , where  $E[.]$  is the expectation value or average of the signal.

### 2.3.1.2 Recursive Least Square (RLS) Algorithm

The algorithms that result from the gradient descent methods have the disadvantages that they tend to be slow to approach the optimal weight vector and, once close to it, usually “rattles around” the optimal; vector rather than actually converge to it. This is due to the effects of approximations made in the estimate of the performance function gradient. To overcome these difficulties, another efficient approach known as RLS algorithm [2.1, 2.2] has been discussed in this section. In this algorithm the input data  $\{x, d\}$  is used in such a way that optimality at each step is ensured.

#### Steps involved in RLS Algorithm

The step-by-step procedures for updating the optimal weight vector are given in this section. It is assumed that the inverse of the auto-correlation matrix,  $R_k^{-1}$  of the input exists. The steps then proceeds as follows

- (i) Accept new samples  $x(k), d(k)$
- (ii) Form  $X(k)$  by shifting  $x(k)$  into the information vector.
- (iii) Compute the a priori output  $y_0(k)$

$$y_0(k) = W_k^{ot} x(k) \quad (2.25)$$

- (iv) Compute a priori error  $e_0(k)$

$$e_0(k) = d(k) - y_0(k) \quad (2.26)$$

- (v) Compute the filtered information vector  $Z_k$

---


$$Z_k = R_k^{-1}X(k) \quad (2.27)$$

(vi) Compute the normalized error power  $q$

$$q = X^t(k)Z_k \quad (2.28)$$

(vii) Compute the gain constant  $v$

$$v = \frac{1}{1+q} \quad (2.29)$$

(viii) Compute the normalized filtered information vector  $\tilde{Z}_k$

$$\tilde{Z}_k = v.Z_k \quad (2.30)$$

(ix) Update the optimal weight vector  $W_k^o$  to  $W_{k+1}^o$

$$W_{k+1}^o = W_k^o + e_o(k)\tilde{Z}_k \quad (2.31)$$

(x) Update the inversion correlation matrix  $R_k^{-1}$  to  $R_{k+1}^{-1}$  in preparation for the next iteration

$$R_{k+1}^{-1} = R_k^{-1} - \tilde{Z}_k \tilde{Z}_k^t \quad (2.32)$$

Initially  $R_k^{-1}$  is taken as

$$R_k^{-1} = \eta I_N \quad (2.33)$$

Where  $I_N$  is an identity matrix of size  $N \times N$ . The value  $\eta$  is initially taken as a large number of about 10000 [2.2].

### 2.3.1.3 Back propagation (BP) Algorithm

An MLP network with 2-3-2-1 neurons (2, 3, 2 and 1 denote the number of neurons in the input layer, the first hidden layer, the second hidden layer and the output layer respectively) with the back-propagation (BP) learning algorithm, is depicted in Fig. 2.6. The parameters of the neural network can be updated in both sequential and batch mode of operation. In BP algorithm, the weights and the thresholds are initialized as very small random values. The intermediate and the final outputs of the MLP are calculated by using (2.11), (2.12), and (2.13) respectively.

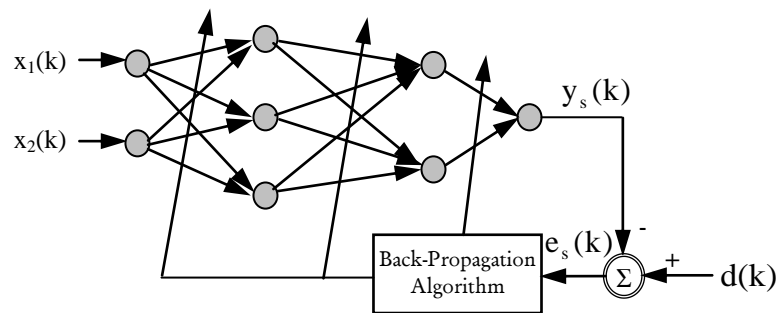


Fig. 2.6 Neural network using BP algorithm

The final output  $y_s(k)$  at the output of neuron 's', is compared with the desired output  $d(k)$  and the resulting error signal  $e_s(k)$  is obtained as

$$e_s(k) = d(k) - y_s(k) \quad (2.34)$$

The instantaneous value of the total error energy is obtained by summing all error signals over all neurons in the output layer, that is

$$\xi(k) = \frac{1}{2} \sum_{s=1}^{P_3} e_s^2(k) \quad (2.35)$$

where  $P_3$  is the no. of neurons in the output layer.

This error signal is used to update the weights and thresholds of the hidden layers as well as the output layer. The reflected error components at each of the hidden layers

is computed using the errors of the last layer and the connecting weights between the hidden and the last layer and error obtained at this stage is used to update the weights between the input and the hidden layer. The thresholds are also updated in a similar manner as that of the corresponding connecting weights. The weights and the thresholds are updated in an iterative method until the error signal becomes minimum.

The weights are updated according to,

$$w_{pq}(k+1) = w_{pq}(k) + \Delta w_{pq}(k) \quad (2.36)$$

$$w_{qr}(k+1) = w_{qr}(k) + \Delta w_{qr}(k) \quad (2.37)$$

$$w_{rs}(k+1) = w_{rs}(k) + \Delta w_{rs}(k) \quad (2.38)$$

where,  $\Delta w_{rs}(k)$ ,  $\Delta w_{qr}(k)$  and  $\Delta w_{pq}(k)$  are the change in weights of the second hidden layer-to-output layer, first hidden layer-to-second hidden layer and input layer-to-first hidden layer respectively. That is,

$$\begin{aligned} \Delta w_{rs}(k) &= -2\mu \frac{d\xi_s(k)}{dw_{rs}(k)} = 2\mu e(k) \frac{dy_s(k)}{dw_{rs}(k)} \\ &= 2\mu e(k) f'_s \left[ \sum_{r=1}^{P_2} w_{rs} \phi_r + b_s \right] \phi_r \end{aligned} \quad (2.39)$$

Where,  $\mu$  is the convergence coefficient ( $0 \leq \mu \leq 1$ ). Similarly the  $\Delta w_{qr}(k)$  and  $\Delta w_{pq}(k)$  can be computed [ 2.4].

The thresholds of each layer can be updated in a similar manner, i.e.

$$b_s(k+1) = b_s(k) + \Delta b_s(k) \quad (2.40)$$

$$b_r(k+1) = b_r(k) + \Delta b_r(k) \quad (2.41)$$

$$b_q(k+1) = b_q(k) + \Delta b_q(k) \quad (2.42)$$

where,  $\Delta b_s(k)$ ,  $\Delta b_r(k)$  and  $\Delta b_q(k)$  are the change in thresholds of the output, hidden and input layer respectively. The change in threshold is represented as,

$$\begin{aligned}\Delta b_s(k) &= -2\mu \frac{d\xi(k)}{db_s(k)} = 2\mu e(k) \frac{dy_s(k)}{db_s(k)} \\ &= 2\mu e(k) f'_s \left[ \sum_{r=1}^{P_2} w_{rs} \phi_r + b_s \right]\end{aligned}\quad (2.43)$$

### 2.3.1.4 The FLANN Algorithm

Referring to Fig. 2.4 the error signal  $e(k)$  at  $k^{\text{th}}$  iteration can be computed as

$$e(k) = d(k) - y(k) \quad (2.44)$$

Let  $\xi(k)$  denotes the cost function at iteration  $k$  and is given by

$$\xi(k) = \frac{1}{2} \sum_{j=1}^P e_j^2(k) \quad (2.45)$$

where  $P$  is the number of nodes at the output layer.

The weight vector can be updated by least mean square (LMS) algorithm, as

$$w(k+1) = w(k) - \frac{\mu}{2} \overset{\wedge}{\nabla}(k) \quad (2.46)$$

where  $\overset{\wedge}{\nabla}(k)$  is an instantaneous estimate of the gradient of  $\xi$  with respect to the weight vector  $w(k)$ . It is derived as

$$\begin{aligned}\overset{\wedge}{\nabla}(k) &= \frac{\partial \xi}{\partial w} = -2e(k) \frac{\partial y(k)}{\partial w} = -2e(k) \frac{\partial [w(k)s(k)]}{\partial w} \\ &= -2e(k)s(k)\end{aligned}\quad (2.47)$$



---

Substituting the values of  $\hat{V}(k)$  in (2.35) we get

$$w(k+1) = w(k) + \mu e(k)s(k) \quad (2.48)$$

where  $\mu$  denotes the step-size ( $0 \leq \mu \leq 1$ ), which controls the convergence speed of the LMS algorithm.

## 2.3.2 Evolutionary Computing based Algorithms

Though the gradient based algorithms are simple to implement, they employ derivative based learning rules to update the weights which at times lead to local optimal solution. In some cases the algorithms rather than converging to the optimum solution normally rattles around it and thus leads to the incorrect estimate of parameters of the model. To overcome these limitations evolutionary algorithms are used which are based on the principle of natural selection and provide global optimal solution.

### 2.3.2.1 Genetic Algorithm (GA)

**Genetic algorithm** is a part of **evolutionary computing**, which is a rapidly growing area of artificial intelligence. Genetic algorithm is inspired by Darwin's theory of evolution. In this case the problems are solved by an evolutionary process resulting in a best (fittest) solution (survivor).

Evolutionary computing was introduced in the 1960s by I. **Rechenberg** in his work "Evolution strategies". His idea was then developed by other researchers. Genetic Algorithm (GAs) was first imposed by **John Holland** and developed by his students and colleagues [2.16] and are available in the book "Adaption in Natural and Artificial Systems" published in 1975.

#### 2.3.2.1.1 Basic Principles of GA

The Algorithm begins with a set of possible solutions called chromosomes which are used to assess the cost surface of the problem. The process can be thought of as

solution breeding in that it creates a new generation of solutions by crossing two chromosomes. The solution variables or genes that provide a positive contribution to the population multiply and be passed through each subsequent generation until an optimal combination is obtained.

The population is updated after each learning cycle through three evolutionary processes: selection, crossover and mutation. These create the new generation of solution variables.

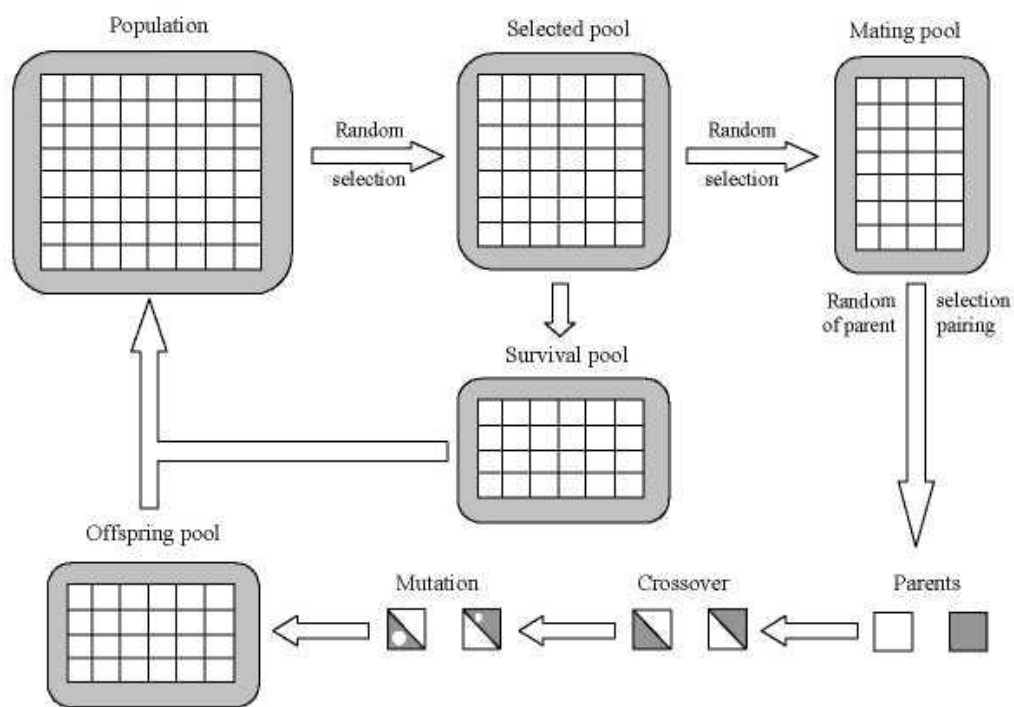


Fig. 2.7 A GA iteration cycle

The selection function creates a mating pool of parent solution strings based upon the “survival of the fittest” criterion. From the mating pool the crossover operator exchanges gene information. This essentially crosses the more productive genes within the solution population to create an improved, more productive generation. Mutation randomly alters selected genes, which helps prevent premature convergence by pulling the population into unexplored areas of the solution surface and adds new gene information into the population.

---

### **2.3.2.1.2 Operators of GA**

As one can see from the iteration cycle of genetic algorithm selection, crossover and mutation are the most important parts of the genetic algorithm. The performance is influenced mainly by these three operators.

#### **Encoding of a Chromosome**

A chromosome should in some way contain information about solution that it represents. The most used way of encoding is a binary string.

A chromosome then could look like this:

Chromosome 1 :- 1101100100110110

Chromosome 2 :- 1101111000011110

Fig. 2.8 Chromosome

Each chromosome is represented by a binary string. Each bit in the string can represent some characteristics of the solution. There are many other ways of encoding. The encoding depends mainly on the solved problem. For example, one can encode directly integer or real numbers; sometimes it is useful to encode some permutations and so on.

#### **Selection**

The selection process is used to weed out the weaker chromosomes from the population so that the more productive chromosomes may be used in the production of the next generation. There are many methods in selecting the best chromosomes. Examples are roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others. In this thesis we have used the roulette wheel based selection as it performs better than the others.

---

## Roulette wheel selection

In this selection process the fitness of each chromosome in the population is evaluated. Then the population is divided into two pools a survival pool and a mating pool. The chromosomes from the mating pool will be used to create a new set of chromosomes through the evolutionary processes of natural selection and the survival pool allows a number of chromosomes to pass onto the next generation. The chromosomes are selected randomly for the two pools but biased towards the fittest. Each chromosome may be chosen more than once and the fitter chromosomes are more likely to be chosen so that they will have a greater influence in the new generation of solutions.

The selection procedure can be described using a biased roulette wheel with the buckets of the wheel sized according to the individual fitness relative to the population's total fitness [2.16]. Consider an example population of ten chromosomes that have the fitness assessment of  $f = \{0.16, 0.16, 0.48, 0.08, 0.16, 0.24, 0.32, 0.08, 0.24, 0.16\}$  and the sum of the fitnesses are used to normalize these values,  $f_{\text{mm}} = 2.08$ . Figure 2.9 shows a roulette wheel that has been split into ten segments and each segment is in proportion to the population chromosomes relative fitness. The third segment is in proportion to the population chromosomes relative fitness. The third segment therefore fills nearly a quarter of the roulette wheel's area. The random selector points to a chosen chromosome, which is then copied into the mating pool because the third individual controls a greater proportion of the wheel, it has a greater probability of being selected.

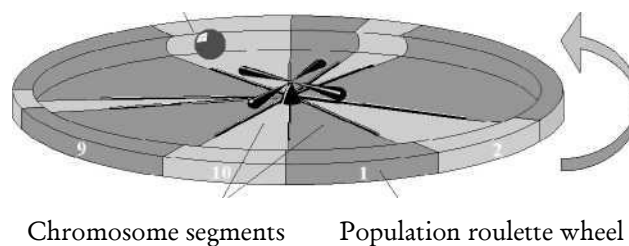


Fig.2.9. Biased roulette-wheel that is used in the selection of the mating pool

---

An individual is selected once the partial sum of fitness becomes greater than the random selector, which will be a value between zero and the sum of fitness. After the GA crossover and mutation operators update the selected mating pool chromosomes, these supersede the old population and consequently the genes from the unselected chromosomes are lost.

## Crossover

The crossover operator exchanges gene information between two selected chromosomes, where this operation aims to improve the diversity of the solution vectors. The pair of chromosomes, taken from the mating pool, becomes the parents of two offspring chromosomes for the new generation.

A binary crossover operation can be either single point or two point crossover. The simplest way to do single point crossover is to choose randomly some crossover point and copy everything before this point from the first parent and then copy everything after the crossover point from the other parent. In Fig.2.10 the fifth crossover position is randomly chosen, where the first position corresponds to the left side. The bits from the right of the fourth bit are exchanged to produce two offspring chromosomes.

Single point crossover can be illustrated as follows:

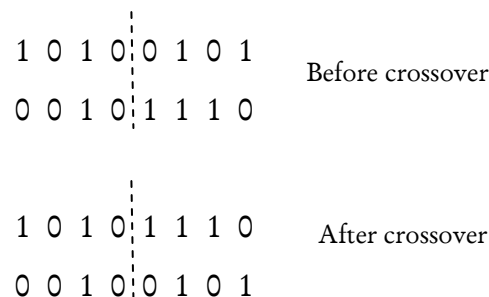


Fig. 2.10 Single point Crossover

In two point crossover two points are randomly chosen and the bits in between them are exchanged.

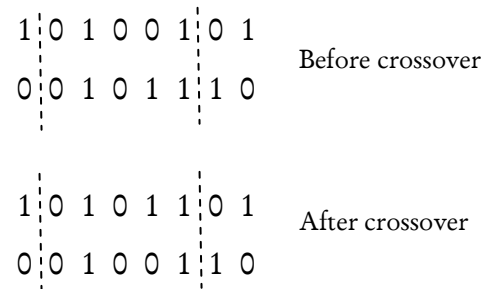


Fig. 2.11 Double point Crossover

### Mutation

After a crossover is performed, mutation takes place. Mutation is intended to prevent falling of all solutions in the population into a local optimum of the solved problem. Mutation operation randomly changes the offspring resulted from crossover. In case of binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. Mutation can be then illustrated as follows:

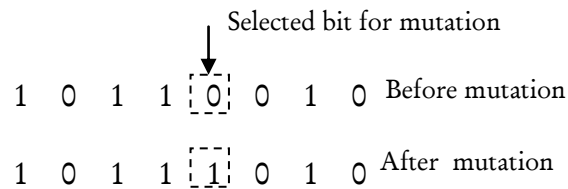


Fig. 2.12 Mutation

---

### **2.3.2.1.3 Parameters of GA**

#### **Crossover and Mutation Probability**

There are two basic parameters of GA - crossover probability and mutation probability.

#### **Crossover probability**

This probability controls the frequency at which the crossover occurs for every chromosome in the search process. This is a number between (0,1) which is determined according to the sensitivity of the variables of the search process. The crossover probability is chosen small for systems with sensitive variables. If there is crossover, offspring are made from parts of both parent's chromosome. Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. However, it is good to leave some part of old population survives to next generation.

#### **Mutation probability**

This parameter decides how often parts of chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. If mutation is performed, one or more parts of a chromosome are changed. If mutation probability is 100%, whole chromosome is changed, if it is 0%, nothing is changed. Mutation generally prevents the GA from falling into local extremes. Mutation should not occur very often, because then GA will in fact change to random search.

#### **Population size**

There are also some other parameters of GA. One another particularly important parameter is population size. It represents the number of many chromosomes in a population (in one generation). If there are too few chromosomes, GA has few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, the solution using GA slows down.

## 2.3.2.2 Particle Swarm Optimization (PSO)

### 2.3.2.2.1 Basic concept of PSO

The Particle Swarm Optimization (PSO) was developed by Eberhart and Kennedy in 1995 [2.23]-[2.25] inspired by swarm intelligence theory such as birds flocking, fish schooling etc. It refers to a relatively new family of algorithms where the individuals evolved through generation by cooperation and competition among each other. In other evolutionary algorithms the evolutionary operators are used to manipulate individuals. In [2.26] it has shown that PSO is comparable in performance with other evolutionary algorithms.

### 2.3.2.2.2 Particle swarm optimization algorithm

In PSO a swarm consists of a set of volume-less particles (a point) moving in a D-dimensional search space, each representing a potential solution. Each particle flies in the search space with position and velocity which are dynamically adjusted according to its own as well as its companions flying experiences.

The concept of movement of particles of PSO described in Fig. 2.13 .

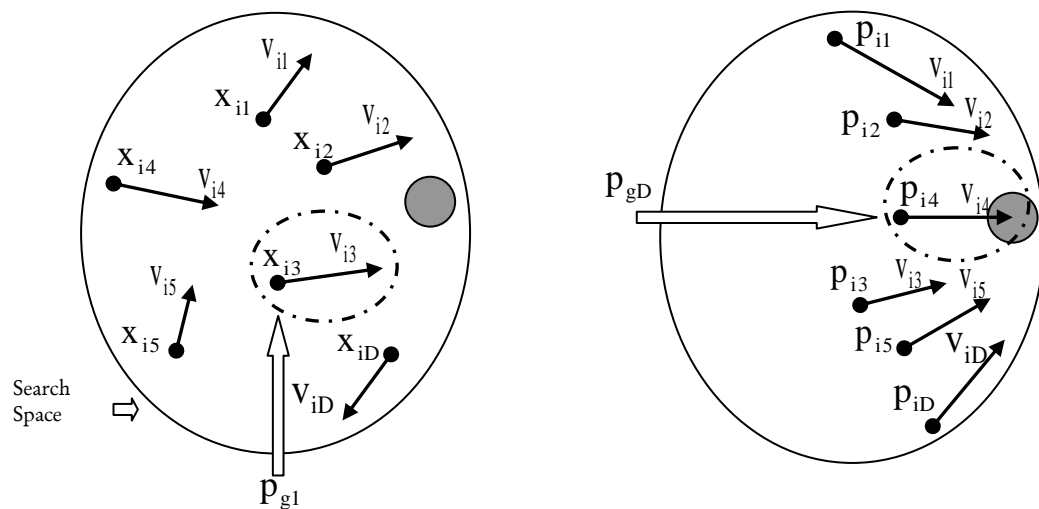


Fig.2.13 (a)

Fig.2.13 (b)

Fig.2.13 Representation of PSO algorithm: (a) Initialization (b) Particles movement towards solution



The  $i^{\text{th}}$  particles is represented by a vector:  $X_i = [x_{i1}, x_{i2} \dots x_{id} \dots x_{iD}]$ . The best previous position (the position giving the best fitness value) of the  $i^{\text{th}}$  particle is recorded and represented as  $P_i = [p_{i1}, p_{i2} \dots p_{id} \dots p_{iD}]$ . At each iteration, the global best particle in the swarm is represented by  $P_g = [p_{g1}, p_{g2} \dots p_{gd} \dots p_{gD}]$ . The rate of change of position of the  $i^{\text{th}}$  particle is represented as  $V_i = [v_{i1}, v_{i2} \dots v_{id} \dots v_{iD}]$ . The maximum velocity and the range of particles are given by  $V_{\max} = [v_{\max 1}, v_{\max 2} \dots v_{\max d} \dots v_{\max D}]$  and  $X_{\max} = [x_{\max 1}, x_{\max 2} \dots x_{\max d} \dots x_{\max D}]$ . The velocity and position of the  $d^{\text{th}}$  element of the  $i^{\text{th}}$  particle at  $(k+1)^{\text{th}}$  search from the knowledge of previous search are modified as per the following

$$V_{id}(k+1) = w(k) * v_{id}(k) + c_1 * r_1 * (p_{id}(k) - x_{id}(k)) + c_2 * r_2 * (p_{gd}(k) - x_{id}(k)) \quad (2.56)$$

$$V_{id}(k+1) = \begin{cases} v_{\max d}, & v_{id}(k+1) > v_{\max d} \\ -v_{\max d}, & v_{id}(k+1) < -v_{\max d} \end{cases} \quad (2.57)$$

$$X_{id}(k+1) = X_{id}(k) + V_{id}(k+1) \quad (2.58)$$

$$X_{id}(k+1) = \begin{cases} x_{\max d}, & x_{id}(k+1) > x_{\max d} \\ -x_{\max d}, & x_{id}(k+1) < -x_{\max d} \end{cases} \quad (2.59)$$

where  $i = 1, 2 \dots N_1$ ,  $d = 1, 2 \dots D$  and  $N_1$  is the number of particles. The symbols  $r_1$  and  $r_2$  represent random numbers between 0 and 1. Similarly  $c_1$  and  $c_2$  denote acceleration constants that pull each particle towards its best and global best positions. The acceleration constants are usually taken as 2.05 for most applications. The inertia weight,  $W$  is employed to control the impact of pervious history of velocities on the current one in order for tradeoff between the global and local exploitations. At early stage of optimization it is desirable that the individual particles wonder though the entire search space, without clustering around the local optima. On the other a hand, during later stages it is very important to enhance convergence toward the global optima so as to find optimum solution efficiently. Large inertia weight enables the PSO to explore locally. So a self adaptive strategy is introduced

---

such that the value of  $w$  is decreased linearly as the generation goes on increasing [2.27]. The time-varying inertia weight is given by

$$w(k) = (w_i - w_f) \left( \frac{I - k}{I} \right) + w_f \quad (2.60)$$

where  $k$  is the search space number.  $w_i$  and  $w_f$  are the initial and final value of inertia weights taken values 0.4 and 0.9 respectively.  $I$  is the maximum number of search or generation.

## **2.4 SUMMARY**

The basic architectures used to develop adaptive models are discussed in this chapter. The structures used are broadly classified into two categories linear (FIR, IIR etc.) and nonlinear (Artificial neural structures like MLP, FLANN etc.). The models parameters are trained by various derivative based and derivative free learning algorithms. The derivative or gradient based algorithms include LMS, RLS, Back Propagation etc. Algorithms based on natural selection or derivative free are genetic algorithm (GA), particle swarm optimization (PSO). The stepwise procedure of each of these learning algorithms is also outlined in this chapter.

**Chapter**  
**3**

**Principle and Theory  
Of  
Artificial Immune Systems (AIS)**

---

## 3.1 Artificial Immune Systems

**T**HE design prospective of development of computational tools inspired by nature is termed as biologically inspired computing. An immune system is a naturally occurring event-response system that can quickly adapt to the changing situations. The efficient mechanisms of a biological immune system (BIS) are ability to remember, classify and neutralize the effect of foreign particles. The understanding and investigation on BIS has increased dramatically over the recent few years by several researchers. These leads to development of new algorithms inspired by BIS, under a new branch of computational intelligence known as artificial immune system (AIS). The AIS is emerging as an active and attractive field involving models, techniques and applications of great diversity [3.1]. It offers powerful and robust information processing capabilities for solving complex problems. Here the objective is to introduce new algorithms inspired by mechanism found in natural immune systems and to develop methodology to apply these algorithms to effectively solve problems of communication and control such as channel equalization and system identification.

## 3.2 Biological Immune System

The Biological Immune System is a complex network of specialized tissues, organs and cells. Its main function is to recognize the presence of strange elements in the body and to respond in order to eliminate or to neutralize the foreign invaders [3.2].

All living organisms are exposed to many different microorganisms and viruses that are of causing illness. These microorganisms are called pathogens. In general organisms try to protect against pathogens using different mechanisms including high temperature, low pH and chemicals that repel or kill the invaders. More advanced organisms (vertebrates) have developed an efficient defense mechanism called the immune system [3.4]. Substances that can stimulate specific responses of the immune system are commonly referred to as antigens (pathogens usually act as antigens). Once

---

the immune system gets stimulated it generates a number of antibodies which respond to the foreign antigens. To be effective the immune system should be able to distinguish between the self (cells, proteins in general any molecule that belongs to or is produced by the body) and non-self (antigens). The self/ non-self discrimination is an essential characteristic of the immune system, since the outcome of an inappropriate response to self molecules is fatal.

The immune system can be envisioned as a multilayer system with defense mechanisms in several layers [3.5]. The three main layers include the anatomic barrier, the innate immunity and the adaptive immunity. They are described as follows :

### **3.2.1 Anatomic Barrier**

The first layer is the anatomic barrier, composed of the skin and the surface of mucous membranes. In fact skin prevents the penetration of most pathogens and also inhibits most bacterial growth because of its low pH. On the other hand, many pathogens enter the body by binding or penetrating through the mucous membranes; these membranes provide a number of nonspecific mechanisms that help to prevent such entry. Saliva, tears and some mucous secretions act to wash away potential invaders and also contain antibacterial and antiviral substances [3.2].

### **3.2.2 Innate Immunity**

Innate immunity is the amount of immunity that gets transferred from the mother to the baby when the individuals are born. It has nonspecific response towards the foreign entities. It is mainly composed of the following mechanisms

**Physiologic barriers** : This includes mechanisms like temperature, pH, oxygen tension and various soluble chemicals. The purpose of these mechanisms is to provide detrimental living conditions for foreign pathogens. For instance, the low acidity of the gastric system acts as a barrier to infection by ingested microorganisms, since they cannot survive the low pH of the stomach.

---

**Phagocytic barriers** : Some specialized cells (like macrophages, neutrophils and natural killer cells) are able to ingest specific material, including whole pathogenic microorganisms. This ingestion has two purposes: to kill the antigen and to present fragments of the invader's proteins to other immune cells and molecules.

**Inflammatory response** : Activated macrophages produce proteins called cytokines. They work as hormone-like messengers that include the inflammatory response, which is characterized by vasodilation and rise in capillary permeability. These changes allow a large number of circulating immune cells to be recruited to the site of the infection. The cytokines are also produced by other immune cells and non-immune cells, for example those that secrete cytokines when damaged [3.6].

### **3.2.3 Adaptive Immunity**

Adaptive immunity [3.7], also called acquired or specific immunity, represents the part of the immune system that is able to specifically recognize and selectively eliminate foreign microorganism and molecules. The main characteristics of the adaptive immunity [3.8] are the following:

**Antigenetic specificity** : It allows the immune system to distinguish subtle differences among antigens.

**Diversity** : The adaptive immune system can generate billions of different recognition molecules that are able to uniquely recognize different structures of foreign antigens.

**Immunologic memory** : The adaptive immune system can remember a previous encounter with an antigen. This helps to deliver a quick response in subsequent encounters.

**Self/non-self recognition** : As the immune cells can distinguish its own cells from foreign antigens and so responds only to the non-self molecules.

It is important to note that the acquired immunity does not act independently of the innate immunity; on the contrary, they work together to eliminate foreign invaders. For instance, the phagocytic cells (innate immunity) are involved in the activation of the

---

adaptive immune response. Also, some soluble factors, produced during a specific immune response, have been found to augment the activity of these of these phagocytic cells [3.2].

An important part of the adaptive immune system is managed by white blood cells, called lymphocytes. These cells are produced in the bone marrow, circulate in the blood and lymph system, and reside in various lymphoid organs to perform immunological functions.

**B-cells and T-cells :** They represent the major population of lymphocytes. The cells are produced by the bone marrow and are inert initially, i.e. they are not capable of executing their functions. In order to become immune-component, they have to go through a maturation process. In the case of B-cells, the maturation process occurs in the bone marrow itself. For T-cells, they have to migrate first to the thymus where they mature. In general, a mature lymphocyte can be considered as a detector that can detect specific antigens. There are billions of these detectors which circulate in the body, constituting an effective, distributed anomaly detection and response system [3.9].

**Humoral immunity :** Mature B-cells express unique antigen-binding receptors (ABR) on their surface. The interaction of the ABR with specific antigen induces proliferation and differentiation of B-cells into antibody-secreting plasma cells. An antibody is a molecule that binds to antigens and neutralize them or facilitate their elimination. Antigen coated with antibodies can be eliminated in multiple ways: by phagocytic cells, by the complement system or by preventing them from performing any damaging function (e.g. binding of viral particles to host cells) [3.8].

**Cellular immunity :** During their maturation, T-cells express an unique ABR on their surface called the T-cell receptor. Unlike B-cell ABR that can recognize antigens alone, T-cell receptors can only recognize antigenic peptides that are presented by cell-membrane proteins known as major histocompatibility complex (MHC) molecules. When a T-cell encounters antigens associated with an MHC molecule on a cell, the T-cell proliferates and differentiates into memory T-cells and various effector T-cells. The cellular immunity is accomplished by these generated effector T-cells.

---

There are different types of T cells that interact in a complex way to kill altered self-cells (for instance, virus infected cells) or to activate phagocytic cells.

**Self/non-self discrimination :** During the maturation process in the thymus the T-cells undergo a process of selection that ensures that they are able to recognize non-self peptides presented by MHC. This process has two main phases: positive selection and negative selection [3.10].

**Positive selection :** During the positive selection phase, T-cells are tested for recognition of MHC molecules expressed on the cortical epithelial cells. If a T-cell fails to recognize any of the MHC molecules, it is discarded; otherwise, it is kept.

**Negative selection :** The purpose of negative selection is to test for tolerance of self cells. T-cells that recognize the combination of MHC and self peptides fail this test. This process can be seen as a filtering of a big diversity of T-cells; only those T-cells that do not recognize self peptides are kept [3.11].

**Immune Memory :** Immune lymphocytes are able to recognize specific antigens through their ABR. Most of the lymphocytes die when the antigen is eliminated; but some of them are kept as memory cells. The memory cells have longer life span and on the next appearance of the same antigen they respond quickly.

### **3.2.4 BIS Response**

The first encounter of naïve immune lymphocytes with an antigen generates the primary response shown in Fig3.1. As the body has never been exposed to that antigen before the time lag for primary response is more. During this time the antigen interacts with the mature lymphocytes, resulting in the proliferation of lymphocytes with a unique antigenic specificity. The specificity of each T-cell and each B-cell is determined prior to its contact with the antigen through matching of a portion of structure. The process of population expansion of particular T-cells and B-cells which recognize the specific antigen is called clonal selection [3.12], [3.9]. Among these proliferated lymphocytes most die when the antigen is eliminated; however some are kept as memory cells as discussed above. The next occurrence of the same



antigen activates a secondary response. In this case the time lag is less and the antigen is detected easily because of the presence of the memory cells. The overall cross-reactive response of the immune system to all the antigens is also presented in the Fig 3.1.

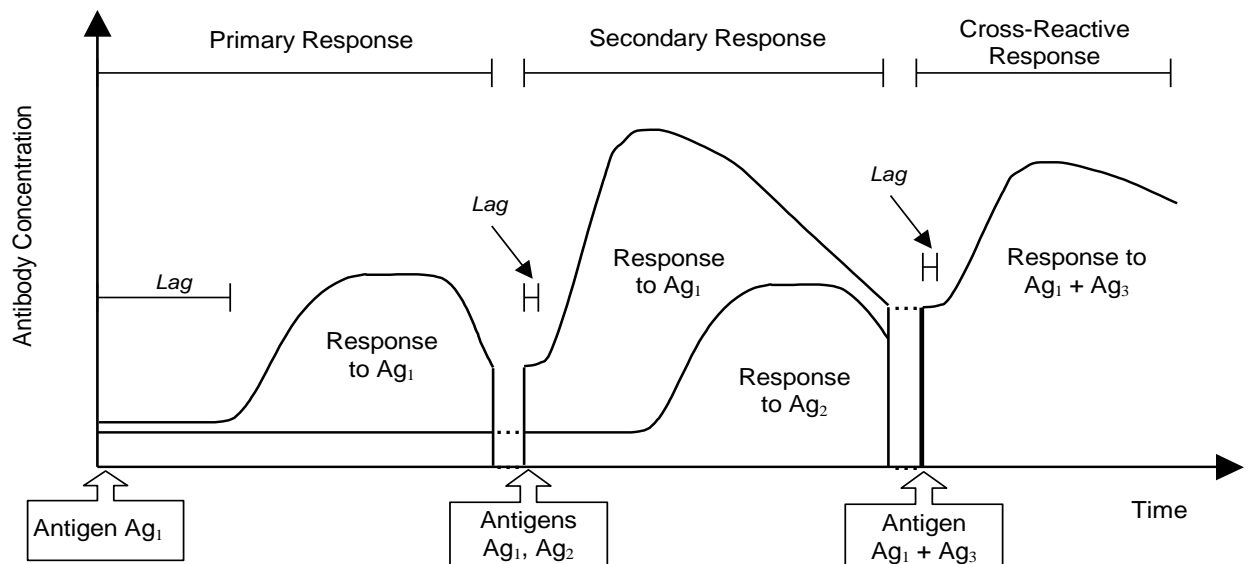


Fig. 3.1 Response of BIS to antigens

### 3.3 Computational aspects of Immune System

From the point of view of information processing, the natural biological immune system exhibits many interesting characteristics. The following is a list of these characteristics [3.13].

**Pattern matching :** The immune system is able to recognize specific antigens and generate appropriate responses. This is accomplished by a recognition mechanism based on chemical binding of receptors and antigens. This binding depends on the molecular shape.

**Feature extraction :** In general antibodies do not bind to the complete antigen, rather portion of it. In this way, the immune system can recognize an antigen just by matching segments of it.

---

**Learning and Memory :** The main characteristic of the adaptive immune system is that it is able to learn through the interaction with the previously encountered antigens. So next time when the same antigen is detected, the memory cells generate a faster and more intense response (secondary response). Memory cells work as an associative distributed memory.

**Diversity :** Clonal selection and hypermutation mechanisms are constantly testing different detector configuration for known and unknown antigens. This is process explores the space of possible configurations looking for close-to-optimum receptors that can cope with the different types of antigens. Exploration is balanced with exploitation by favoring the reproduction of promising individuals.

**Distributed Processing :** Unlike nervous system, the immune system does not possess a central controller. Detection and response can be executed locally and immediately without communicating with any central organ. This distributed behavior is accomplished by billions of immune molecules and cells that circulate around the blood and lymph systems and are capable of making decisions in a local collaborative environment.

**Self-regulation :** Depending on the severity of the attack, response of the immune system can range from very light almost imperceptible to very strong. A stronger response uses a lot of resources to help repel the attacker. Once the invader is eliminated, the immune system regulates itself in order to stop the delivery of new resources and to release the used ones.

**Self-protection :** By protecting the whole body the immune system is protecting itself. It means that there is no other additional system to protect and maintain the immune system.

### **3.4 Theories of AIS**

The study and design of the artificial immune systems (AIS) is a relatively new area of research that tries to build computational systems that are inspired by the natural biological immune system. As we mentioned in subsection 3.3 there are many

---

desirable computational feature in BIS that can be used to solve computational problems. A typical AIS model/algorithm implements one or more of these features. The books [3.8] and [3.9] provides the detail of the modeling and applications of AIS. The four forms of AIS algorithm reported in the literature are immune network model, negative selection, clonal selection and danger theory.

### 3.4.1 Immune Network Model

The immune network model was proposed by Jerne [3.14]. This theory proposed that the immune system maintains a idiotypic network of interconnected cells for antigen recognition. These cells both stimulate and suppress each other in a certain way that leads to stabilization of network. The formation of such a network is possible by the presence of paratope and idiotope on the each antibody cell. The paratope present on one B-cell is recognized by other B-cells idiotopes so each cell recognize as well as recognized. In this network two cells are connected if the affinities they share exceed a certain threshold and the strength of the connection is directly proportional to the affinity they share.

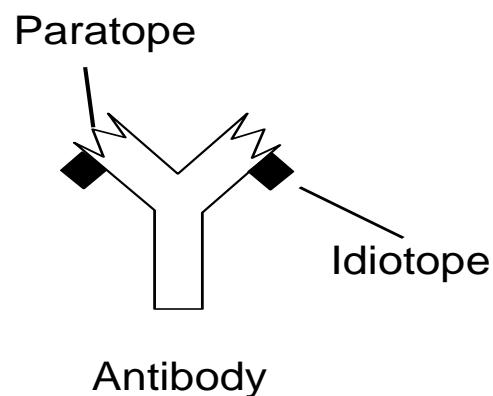


Fig. 3.2 Presence of paratope and idiotope on antibody

In network formation point of view two things are very important : antigen-antibody binding and antibody-antibody binding. This idiotypic network can also be thought of as having cognitive capabilities that makes it similar to a neural network [3.15].

---

### 3.4.2 Negative Selection Algorithm

The purpose of negative selection is to provide tolerance for self-cells. It deals with the immune system's ability to detect unknown antigens while not reacting to the self-cells [3.16]-[3.23]. During the generation of T-cells, receptors are made through a pseudo-random genetic rearrangement process. Then, they undergo a censoring process in the thymus, called the negative selection. There, T-cells that react against self-proteins are destroyed; thus, only those that do not bind to self-proteins are allowed to leave the thymus. These matured T-cells then circulate throughout the body to perform immunological functions and protect the body against foreign antigens.

This algorithm is given by Forest et al. [3.16], [3.20] whose main steps are

**Step 1.** In generation stage, the detectors are generated by some random process and censored by trying to match self samples as shown in Fig 2.

**Step 2.** Those candidates that match are eliminated and the rest are kept as detectors.

**Step 3.** In the detection stage, the collection of detectors (or detector set) is used to check whether an incoming data instance is self or non-self as shown in Fig 3.

**Step 4.** If it matches any detector, then it is claimed as non-self or anomaly.

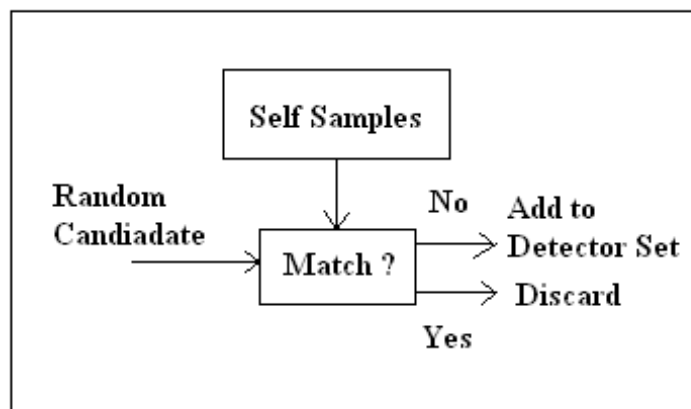


Fig. 3.3(a) Censoring Stage

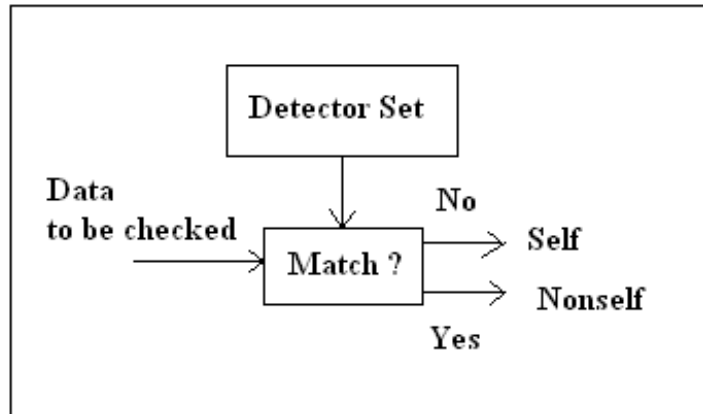


Fig. 3.3(b) Monitoring Stage

Fig. 3.3 Basic of Negative Selection Algorithm (a) Censoring Stage (b) Monitoring Stage

### 3.4.3 Clonal Selection Algorithm

The clonal selection principle of AIS describes how the immune cells eliminate a foreign antigen and is simple but efficient approximation algorithm for achieving optimum solution. The basic algorithm is first applied by Charsto et al. for solving optimization problems [3.25]-[3.26]. The steps involved in the clonal selection algorithm are

**Step 1:** Initialize a number of antibodies (immune cells) which represent initial population size.

**Step 2:** When an antigen or pathogen invades the organism; a number of antibodies that recognize these antigens survives. In Fig.3.4 only the antibody C is able to recognize the antigen<sup>3</sup> as its structure fits to a portion of the pathogen. So fitness of antibody C is higher than others.

**Step 3:** The immune cells recognize antigens under go cellular reproduction. During reproduction the somatic cells reproduce in an asexual form, i.e. there is no crossover of genetic material during cell mitosis. The new cells are copies (clones) of their parents as shown for antibody C in Fig.3.4.

**Step 4:** A portion of cloned cells undergo a mutation mechanism which is known as somatic hypermutation as described in [3.25].

**Step 5:** The affinity of every cell with each other is a measure of similarity between them. It is calculated by the distance between the two cells. The antibodies present in a memory response have on average a higher affinity than those of early primary response. This phenomenon is referred to as maturation of immune response. During the mutation process the fitness as well as the affinity of the antibodies gets changed. In each iteration after cloning and mutation those antibodies which have higher fitness and higher affinity are allowed to enter the pool of efficient cells. Those cells with low affinity or self-reactive receptors must be efficiently eliminated.

**Step 6:** At each iteration among the efficient immune cells some become effector cells (Plasma Cell), while others are maintained as memory cells. The effector cells secrete antibodies and memory cells having longer span of life so as to act faster or more effectively in future when the organism is exposed to same or similar pathogen.

**Step 7:** The process continues till the termination condition is satisfied else steps 2 to 7 are repeated.

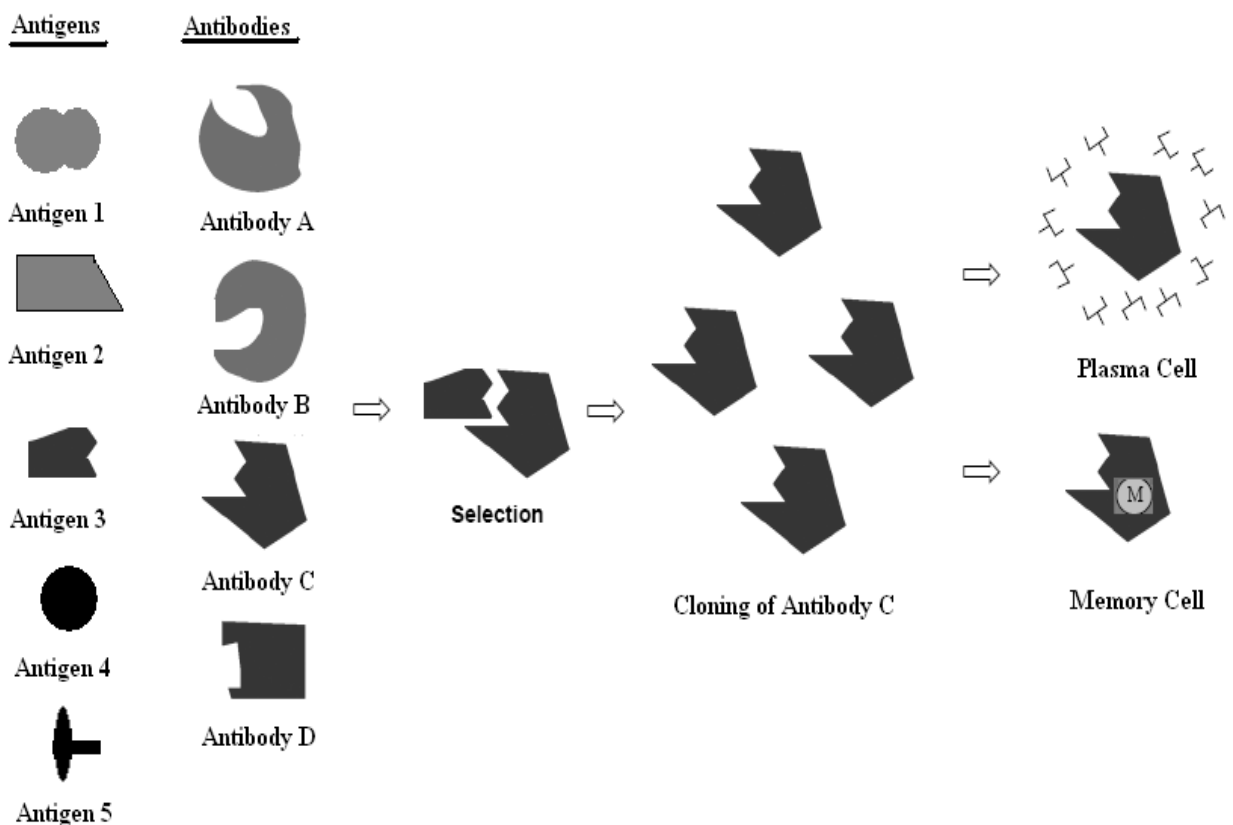


Fig. 3.4 Basic of Clonal Selection Algorithm

---

The clonal selection algorithm has several interesting features such as population size is dynamically adjustable, exploration of the search space, location of multiple optima, capability of maintaining local optima solutions and defined stopping criteria.

### **3.4.4 Danger Theory**

This theory is given by Matzinger in 1994 [3.29]. The immune system in order to function properly, it's very important that only the "correct" cells are matched as otherwise this could lead to a self-destructive autoimmune reaction. Classical immunology [3.31] stipulates that an immune response is triggered when the body encounters something non-self or foreign. It is not yet fully understood how this self-nonsel discrimination is achieved, but many immunologists believe that the difference between them is learnt early in life. In particular, it is thought that the maturation process plays an important role to achieve self-tolerance by eliminating those T- and B-cells that react to self. In addition, a "confirmation" signal is required: that is, for either B-cell or T- (killer) cell activation, a T- (helper) lymphocyte must also be activated. This dual activation is further protection against the chance of accidentally reacting to self.

In accordance to danger theory there must be discrimination happening that goes beyond the self-nonsel distinction described above. For instance:

1. There is no immune reaction to foreign bacteria in the gut or to the food we eat although both are foreign entities.
2. Conversely, some auto-reactive processes are useful, for example against self molecules expressed by stressed cells.
3. The definition of self is problematic—realistically, self is confined to the subset actually seen by the lymphocytes during maturation.
4. The human body changes over its lifetime and thus self changes as well. Therefore, the question arises whether defenses against non-self learned early in life might be auto-reactive later.

Other aspects that seem to be at odds with the traditional viewpoint are autoimmune diseases and certain types of tumors that are fought by the immune

system (both attacks against self) and successful transplants (no attack against non-self).

The Danger Theory takes care of “non-self but harmless” and of “self but harmful” invaders into our system. The central idea is that the immune system does not respond to non-self but to danger. Practically there is no need to attack everything that is foreign, something that seems to be supported by the counter-examples above. In this theory, danger is measured by damage to cells indicated by distress signals that are sent out when cells die an unnatural death. As shown in Fig.3.5 a cell that is in distress sends out an alarm signal, whereupon antigens in the neighborhood are captured by antigen-presenting cells such as macrophages, which then travel to the local lymph node and present the antigens to lymphocytes. Essentially, the danger signal establishes a danger zone around itself. Thus B-cells producing antibodies that match antigens within the danger zone get stimulated and undergo the clonal expansion process. Those that do not match or are too far away do not get stimulated.

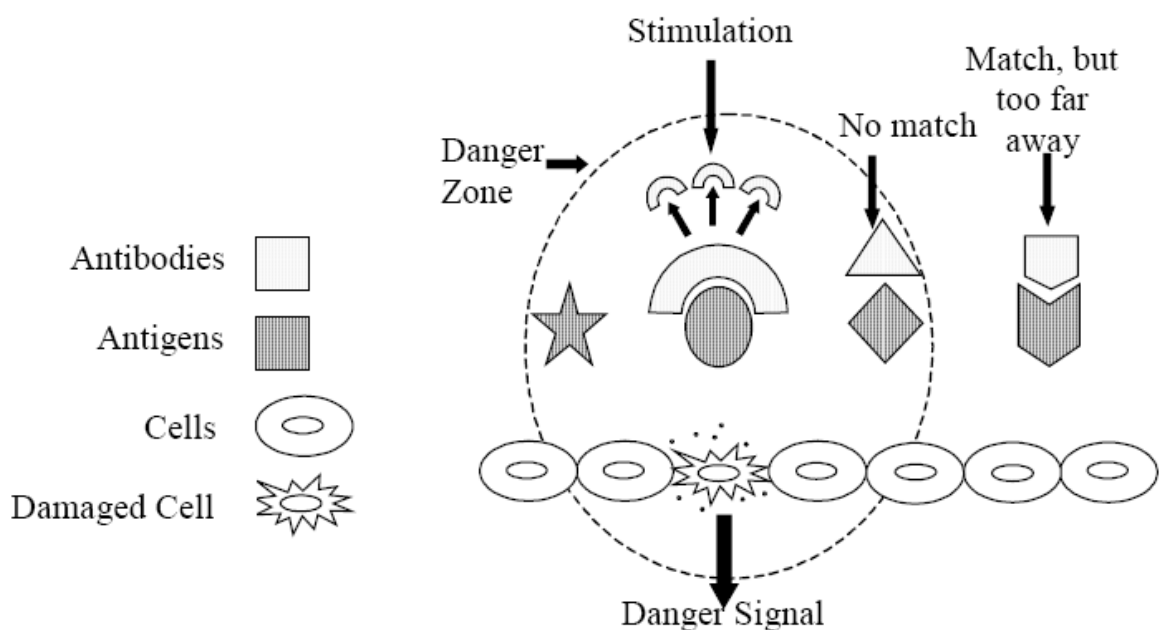


Fig. 3.5 Principle of Danger Theory



---

In accordance with Danger theory Bretscher and Chon proposed a two signal model. According to this

Signal1 : this is used for antigen recognition. Basically to determine the cell is a foreign cell.

Signal2 : this is used for co-stimulation. This refers that the cell is really dangerous. So in accordance to the two signal model the danger theory operates by 3 steps

**Step1** : Become activated if you receive signals one and two together. Die if you receive signal one in the absence of signal two. Ignore signal two without signal one.

**Step2** : Accept signal two from antigen-presenting cells only. Signal one may come from any cell.

**Step3** : After activation revert to resting state after a short time .

The challenge is clearly to define a suitable danger signal. The danger signal helps to identify which subset of feature vectors is of interest. A suitably defined danger signal overcomes many of the limitations of self–nonself selection. It restricts the domain of nonself to a manageable size, removes the need to screen against all self, and deals adaptively with scenarios where self (or nonself) changes over time.

### **3.5 Recent uses of AIS based Modeling**

In recent years the area of Artificial Immune System (AIS) based modeling has drawn attention of many researchers due to its broad applicability to different fields. Some of the significant application areas include optimization problem [3.25]-[3.26], [3.42], computer security [3.17]-[3.18], [3.37], design of intrusion detection [3.19]-[3.21], fault detection [3.22], fault tolerance [3.35]-[3.36], pattern recognition[3.27], distributed learning[3.33], sensor network[3.43], Job-shop scheduling[3.44], design of recommendation system[3.46]-[3.47] etc. The AIS is relatively young and emerging as an active and attractive field involving models, techniques and applications of great diversity.

---

## **3.6 SUMMERY**

This chapter presents the functionality of BIS that is how the body restricts itself from the invasion of external microorganisms. It also outlines how the artificial immune system (AIS) is developed by following the principle of BIS. The four forms of AIS algorithm are discussed and the application areas are highlighted. In the present the clonal selection principle is chosen to be used for various application as it is simple and easy to implement.

**Chapter**  
**4**

**Application of AIS to Nonlinear  
System Identification**

---

## 4.1 Introduction

**I**DENTIFICATION of Nonlinear plants finds extensive applications in stability analysis, controller design, modeling of intelligent instrumentation, analysis of power systems, modeling of multipath communication channels etc. When these practical plants are considered for identification their behaviors are completely unknown. The plant behavior may be nonlinear, dynamic, time varying, chaotic etc. So in general it is difficult to model practical nonlinear plants by conventional numerical analysis methods. Therefore adaptive methods of system identification have been proposed and are being used in practice.

## 4.2 Principle of Adaptive System Identification

The system identification concerns with the determination of a system on the basis of input output data samples. The identification task is to determine a suitable estimate of finite dimensional parameters which completely characterize the plant. The selection of the estimate is based on comparison between the actual output sample and predicted value on the basis of input data up to that instant. An adaptive automation is a system whose structure is alterable or adjustable in such a way that its behavior or performance improves through contact with its environment.

The essential and principal characteristics of an adaptive system is its time-varying, self-adjusting performance. System identification [4.1, 4.2] is the experimental approach to process modeling. System identification includes the following steps

**Experiment design :** Its purpose is to obtain good experimental data and it includes the choice of the measured variables and of the character of the input signals.

**Selection of model structure :** A suitable model structure is chosen using prior knowledge and trial and error.

---

**Choice of the criterion to fit :** A suitable cost function is chosen, which reflects how well the model fits the experimental data.

**Parameter estimation :** An optimization problem is solved to obtain the numerical values of the model parameters.

**Model validation:** The model is tested in order to reveal any inadequacies.

The adaptive systems have following characteristics [4.2]

- 1) They can automatically adapt (self-optimize) in the face of changing (non-stationary) environments and changing system requirements.
- 2) They can be trained to perform specific filtering and decision making tasks.
- 3) They can extrapolate a model of behavior to deal with new situations after trained on a finite and often small number of training signals and patterns.
- 4) They can repair themselves to a limited extent.
- 5) They can be described as nonlinear systems with time varying parameters.

## 4.3 Classification of Nonlinear Systems

According to number of input output nodes the nonlinear plants are broadly classified into either single input single output (SISO) or multiple input multi output (MIMO) systems. Depending upon the input output relation of the plants they are divided into two groups

### (i) Static systems

In static systems the output at any instant of time depends upon the input at that instant. The system is essentially a memory-less one and is mathematically represented as

$$y(k) = f[u(k)] \quad (4.1)$$

where  $f[.]$  represents the nonlinearity associated with the system.

---

## (ii) Dynamic systems

In dynamic systems the output at any instant depends upon the input at that instant as well as the past inputs and output values. These systems have memory to store past values. The nonlinear dynamic systems [4.5] is assumed to have one of the following form

**Model 1:**

$$y(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-i) + g[u(k), u(k-1), \dots, u(k-m+1)] + d(k) \quad (4.2)$$

**Model 2:**

$$y(k+1) = f[y(k), y(k-1), \dots, y(k-n+1)] + \sum_{i=0}^{m-1} \beta_i u(k-i) + d(k) \quad (4.3)$$

**Model 3:**

$$y(k+1) = f[y(k), y(k-1), \dots, y(k-n+1)] + g[u(k), u(k-1), \dots, u(k-m+1)] + d(k) \quad (4.4)$$

**Model 4:**

$$y(k+1) = f[y(k), y(k-1), \dots, y(k-n+1), u(k), u(k-1), \dots, u(k-m+1)] + d(k) \quad (4.5)$$

where  $d(k)$  is the noise associated with the nonlinear plant. The symbols  $u(k)$  and  $y(k)$  represents the input and output of the nonlinear plant at time instant  $k$  and  $m \leq n$ . The terms  $f[.]$  and  $g[.]$  denote nonlinear functions and  $\alpha_i$  and  $\beta_i$  represent constant values.

---

## 4.4 Problems associated with existing gradient based models

Many research work on nonlinear system identification have been reported in the literature but an important one is using multilayer perceptron (MLP) by Narendra and Parthasarathy (1990) [4.3]. Later on in [4.5] a new approach to identify such systems had proposed which provides identical or even better performance but employing a low complexity FLANN structure. However, the major disadvantage of these methods is that they employ derivative based learning rule to update their weights which at times leads incorrect estimate of the parameters because of the following reasons

- (i) **Converge to local optima** : In case of gradient based learning there is maximum probability of convergence of the weights of the model to local minima rather than global minima.
- (ii) **Rattling around the optimal Solution** : Due to the fixed step size of the search in gradient based method the estimated weights normally rattles around the optimal solution rather than converge to it.

## 4.5 Proposed hybrid FLANN-AIS Model

In this section we propose a low complexity FLANN structure whose weights are trained with clonal selection algorithm of AIS rather than conventional LMS based approach discussed in previous section (2.2.2.3) and (2.3.1.4). The clonal selection algorithm discussed in section (3.4.3) is simple but an efficient approximation algorithm for achieving optimum solution. The detail about the application of the proposed hybrid technique for identification of nonlinear SISO and MIMO plant is discussed below in the following subsections.

## 4.5.1 Nonlinear SISO Plant Identification

The block diagram of adaptive SISO system identification based on proposed method is shown in Fig.4.1. The symbols  $u(k)$ ,  $y(k)$ ,  $\hat{y}(k)$  and  $e(k)$  represent the input, output of plant, the estimated output of the model and the error signal respectively. The objective of the identification task is to minimize the error  $e(k)$  recursively such that  $\hat{y}(k)$  approaches  $y(k)$  when the same input  $u(k)$  is applied to the plant and the model.

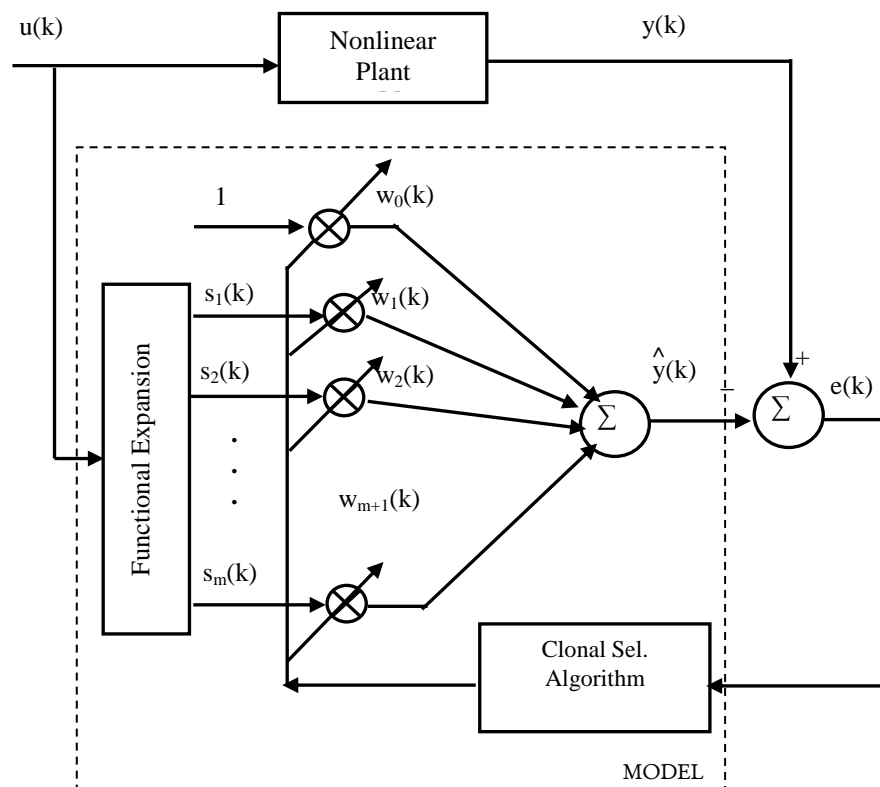


Fig.4.1. Block diagram of Nonlinear SISO plant identification

The stepwise procedure of the proposed identification algorithm is described as follows

**Step 1. Determination of output of plant :** The input is a random signal drawn from a uniform distribution in the interval  $[-1, 1]$ . Let 'k' be the numbers of



input samples taken. The input sample is then passed through the plant to produce plant output  $y(k)$ .

**Step 2. Functional expansion of input :** Same input samples are passed through the model consisting FLANN structure. Each input sample under goes trigonometric expansion given by

$$s_i(k) = \begin{cases} 1 & \text{for } i = 0 \\ u(k) & \text{for } i = 1 \\ \sin(i\pi u(k)) & \text{for } i = 2, 4 \dots M \\ \cos(i\pi u(k)) & \text{for } i = 3, 5 \dots M+1 \end{cases} \quad (4.6)$$

where  $i = 1, 2, \dots, M/2$ . The bias input is unity. So total expanded values including the bias become  $Q = M+2$ .

**Step 3. Initialization of a group of cells :** As it is an evolutionary algorithm we begin with a group of solutions .Here a group of weight vector of FLANN is taken. A weight vector consist of  $(M+2)$  no of elements. Each element of weight vector is represented by an immune cell which is basically a binary string of definite length. So a set of binary strings is initialized to represent a weight vector and n number of such weight vectors is taken each of which represent probable solution.

**Step 4. Decoding :** As each cell constitutes random binary bits they need to be converted to decimal values lying between some ranges to compute the fitness function. The equation that converts the binary to real number is given by

$$RV = R_{\min} + \{(R_{\max} - R_{\min}) / (2^L - 1)\} \times DV \quad (4.7)$$

where  $R_{\min}$  ,  $R_{\max}$  ,  $RV$  and  $DV$  represent the minimum range, maximum range, decimal and decoded value of an L bit coding scheme respectively.

**Step 5. Calculation of output of model :** Initially weight vector is taken as a random vector. The output of model is computed using expanded values of  $u(k)$  and weight vector . This is repeated for n times.

$$\hat{\mathbf{y}}(k) = \sum_{i=1}^Q \mathbf{s}_i(k) \mathbf{w}_i(k) \quad (4.8)$$

**Step 6- Fitness Evaluation :** The output of the model  $\hat{y}(k,n)$  due to  $k^{\text{th}}$  sample and  $n^{\text{th}}$  vector, is compared with the plant output to produce error signal given by

$$e(k,n) = y(k,n) - \hat{y}(k,n) \quad (4.9)$$

For each  $n^{\text{th}}$  weight vector the mean square error (MSE) is determined and is used as fitness function given by

$$\text{MSE}(n) = \frac{\sum_{k=1}^K e^2(k,n)}{K} \quad (4.10)$$

The objective is to minimize the fitness function of (4.10) by clonal selection principle.

**Step 7- Selection :** To select the weight vector (corresponding cells) for which MSE is minimum.

**Step 8- Clone:** The weight vector (corresponding cells) which yields best fitness value (minimum MSE) is duplicated.

**Step 9- Mutation:** Mutation operation introduces variations into the immune cells. Probability of mutation  $P_m$  is a smaller value which indicates that the operation occurs occasionally. Total number of bits to mutate is the product of total number of cells, number of bits in each cell and probability of mutation of each cell. Among the cloned cells the cell to be mutated is chosen randomly. A random position of the cell is chosen first and then its bit value is altered.

**Step 10- Stopping Criteria:** Steps 4-9 are repeated until a predefined MSE is obtained. The desired weight vector is achieved at this stage. This is known as memory cell.

## 4.5.2 Nonlinear MIMO Plant Identification

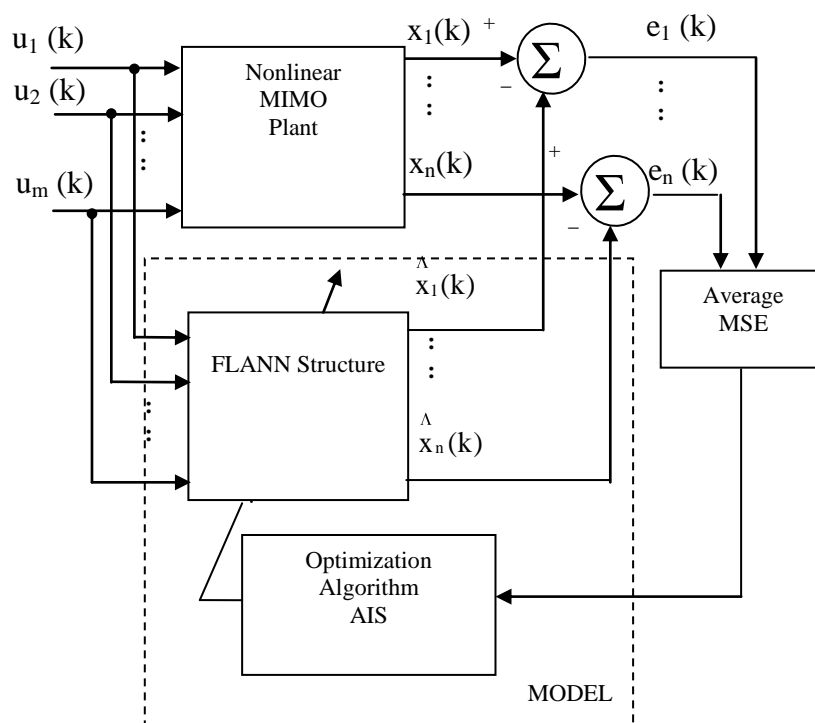


Fig.4.2. Block diagram of Nonlinear MIMO plant identification

The block diagram of proposed model for adaptive MIMO plant identification is shown in Fig.4.2 in which the plant is considered as nonlinear and dynamic in nature. The proposed model is a nonlinear single layer functional link neural network (FLANN) with no hidden layer with its connecting weights trained by AIS based algorithm. Let

$$u(k) = [u_1(k) \ u_2(k) \ u_3(k) \dots \ u_m(k)] \quad (4.11)$$

$$x(k) = [x_1(k) \ x_2(k) \ x_3(k) \dots \ x_n(k)] \quad (4.12)$$

$$\hat{x}(k) = [\hat{x}_1(k) \ \hat{x}_2(k) \ \hat{x}_3(k) \dots \ \hat{x}_n(k)] \quad (4.13)$$

$$e(k) = [e_1(k) \ e_2(k) \ e_3(k) \dots \ e_n(k)] \quad (4.14)$$

where symbols  $u(k)$ ,  $x(k)$ ,  $\hat{x}(k)$  and  $e(k)$  represent the input, output of plant, the estimated output of the model and the error signal respectively.

The MSE of the outputs of the MIMO plant are given by

$$mse_1 = \frac{\sum_{k=1}^K e^2(k,1)}{K} \quad (4.15)$$

⋮

$$mse_n = \frac{\sum_{k=1}^K e^2(k,n)}{K} \quad (4.16)$$

where  $K$  is the total number of input samples applied. The Average MSE is defined by

$$AMSE = \frac{\sum_{i=1}^n mse_i}{n} \quad (4.17)$$

The objective of the identification task is to minimize the AMSE recursively such that  $\hat{x}(k)$  approaches  $x(k)$  when the same input  $u(k)$  is applied to the plant and the model. The stepwise procedure of the proposed identification algorithm is described as follows

**Step 1. Determination of outputs of MIMO plant :** Let ‘ $K$ ’ be the total numbers of input samples taken. The input signals  $u(k)$  are fed to the plant to produce plant outputs  $x(k)$ .

**Step 2. Functional expansion of inputs in the model :** Same input samples are passed through the FLANN structure of the model. Each input sample undergo trigonometric expansion.

$$s_i(k) = \begin{cases} 1 & \text{for } i = 0 \\ u_1(k) & \text{for } i = 1 \\ \sin(i\pi u_1(k)) & \text{for } i = 2, 4 \dots M \\ \cos(i\pi u_1(k)) & \text{for } i = 3, 5 \dots M + 1 \end{cases} \quad (4.18)$$

---

**Step 3. Decoding :** The cells which represents binary values are converted to decimal values in accordance with (4.7).

**Step 4. Initialization of a group of cells:**

Here a group of weight vector of FLANN is taken. Each element of weight vector is represented by a cell which is basically a binary string of definite length. So a set of binary strings is initialized to represent a weight vector and n number of such weight vectors is taken each of which represent probable solution.

**Step 5. Calculation of outputs of model:**

Initially weight vector is taken random value. The outputs of model is computed using expanded values of  $u(k)$  and and weight vector as follows

$$\hat{y}(k) = \sum_{i=1}^Q s_i(k) w_i(k) \quad (4.19)$$

**Step 6. Fitness Evaluation:**

The output of the model  $\hat{x}(k,n)$  due to  $k^{\text{th}}$  sample and  $n^{\text{th}}$  vector, is compared with the plant output to produce error signal. For each  $n^{\text{th}}$  weight vector the AMSE is determined as described in accordance with (4.17).

Steps 7-10 are same as that for nonlinear SISO plant identification.

## 4.6 System Simulation

To demonstrate the performance of the proposed identification model discussed in section 4.5 simulation study using MATLAB is carried out. Some benchmark nonlinear static, dynamic and MIMO plants are identified and the performance is compared with other standard models on the following basis

- I. Comparison of overall output responses of the plant and the estimated model.
- II. Comparison of sum of squared errors (SSE) between true (plant) and estimated (model) response. The sum of squared error is defined as

$$SSE(k) = \sum_{k=1}^K (y(k) - \hat{y}(k))^2 \quad (4.19)$$

where  $y(k)$  is true output and  $\hat{y}(k)$  is estimated output during testing.

III. Comparison of overall CPU time required to train the model.

In order to have better idea about the improved performance of the proposed model the simulation study is broadly performed on 2 things

A. Comparison of different structural models like FLANN, MLP.

B. Comparison of same structure with different evolutionary algorithms like GA, PSO.

## 4.6.1 Identification of Nonlinear Static Plants

The example of the nonlinear static plant taken here is described by (4.1) to (4.5). The nonlinearity  $f[.]$  is given by

**Example 1:**

$$f_1(u(k)) = u^3(k) + 0.3u^2(k) - 0.4x(k) \quad (4.20)$$

**Example 2:**

$$f_2(u(k)) = 0.6 \sin(\pi u(k)) + 0.3 \sin(3\pi u(k)) + 0.1 \sin(5\pi u(k)) \quad (4.21)$$

**Example 3:**

$$f_3(u(k)) = \frac{4u^3(k) - 1.2u^2(k) - 3u(k) + 1.2}{0.4u^5(k) + 0.8u^4(k) - 1.2u^3(k) + 0.2u^2(k) - 3} \quad (4.22)$$

**Example 4:**

$$f_4(u(k)) = 0.5 \sin^3(\pi u(k)) - \frac{2}{u^3(k) + 2} - 0.1 \cos(4\pi u(k)) + 1.125 \quad (4.23)$$

where the input to the system  $u(k)$  is a uniformly distributed random signal over the

---

interval  $[-1, 1]$ . For identification the proposed algorithm discussed in section 4.5.1 is used. The model used for identification using FLANN structure is shown in Fig. 4.2. Simulation study is also carried out for the above functions using MLP structure trained by back propagation algorithms as presented in [4.5]. Investigation is also done by taking the same FLANN structure whose weights are trained with other evolutionary algorithms like GA and PSO on similar environment.

For simulation of all above functions the detail comparative study during training and testing of model is represented in Table I. In the proposed model the initial population of cells is taken as 78. The number of trigonometric expansion of input used in the FLANN depends upon the nonlinearity associated with the function. For identification of plants  $f_1, f_2, f_3$  and  $f_4$  the numbers of expansions are 5,11,7 and 11 respectively. The weights of the FLANN structure are trained for 150, 300, 200 and 3900 iterations respectively for  $f_1, f_2, f_3$  and  $f_4$  plants. Here for clonal selection based training each element of weight vector is represented by a cell which is basically a binary string of 10 bits. The values of  $R_{\max}$  and  $R_{\min}$  are chosen judiciously to attain satisfactory results. The probability of mutation is taken as 0.1.

For simulation using MLP, the structure is taken as  $\{1\_20\_10\_1\}$ . The nonlinearity used at the node is hyperbolic tan function. In back propagation based training both the convergence parameter and momentum term is set to 0.1. The weights of MLP are trained for 50000 iterations.

For simulation using FLANN-GA based model, the structure of FLANN is same as that of the proposed model. The basic genetic algorithm described in section 2.3.2.1 is used. Here each element of weight vector is represented by a chromosome. Initial population of chromosome is same as that of the immune cells taken. Binary coded GA is used where each chromosome is of length 10 bit. Roulette wheel scheme is used for selection. The fitness function is MSE as described in (4.10). The probabilities of crossover and mutations are taken values 0.8 and 0.1 respectively. The number of generations is taken same as that of the proposed model.

In FLANN-PSO based model same structure of FLANN is taken. The basic genetic algorithm is described in section 2.3.2.2 is used. The weights are represented by position of the particles. Each position is associated with a corresponding velocity. The symbols  $r_1$  and  $r_2$  represent random numbers between 0 and 1. Similarly  $c_1$  and  $c_2$  denote acceleration constants normally taken value 2.05. The initial and final time varying inertia weights  $w_i$  and  $w_f$  are taken as 0.9 and 0.4 respectively. The number of search iteration taken is same as that of the number of generation in the proposed model.

The performance of all the models are compared in terms of estimated output and the error plots for  $f_1, f_2, f_3$  and  $f_4$  plants are shown in Figures 4.3 - 4.6. Table 4.1 reveals that the proposed models offer lesser CPU time, lesser input samples and minimum sum of squared errors compared to other models.

TABLE 4.1  
COMPARATIVE RESULTS OF STATIC SISO SYSTEMS OBTAINED THROUGH SIMULATION STUDY

Static systems	Training						Testing				
	No. of input samples used by structures		CPU time required during Training (in Sec.)				No. of input samples used by both structures	Sum of square errors (SSE)			
	MLP	FLANN	MLP BP	FLANN AIS	FLANN GA	FLANN PSO		MLP BP	FLANN AIS	FLANN GA	FLANN PSO
Ex-1	50000	90	105.5	10.5	21.7	18.5	90	0.017	0.010	0.042	0.303
Ex-2	50000	60	100.0	18.0	51.6	36.4	60	0.916	0.029	2.748	5.670
Ex-3	50000	75	110.3	56.3	104.1	74.2	75	0.023	0.019	0.761	0.893
Ex-4	50000	60	120.8	704.7	2048.2	750.8	60	1.511	0.132	0.977	1.275



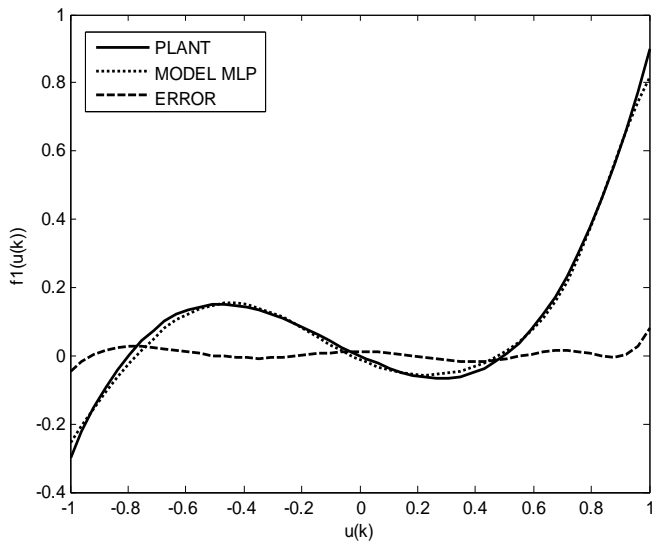


Fig.4.3 (a)

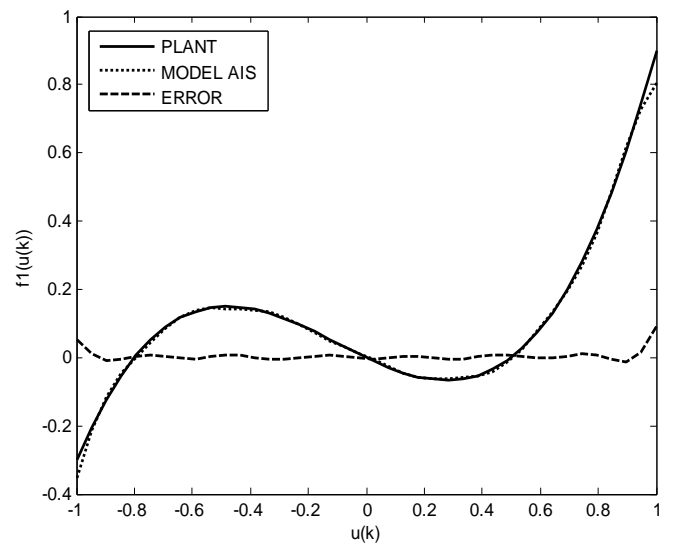


Fig.4.3 (b)

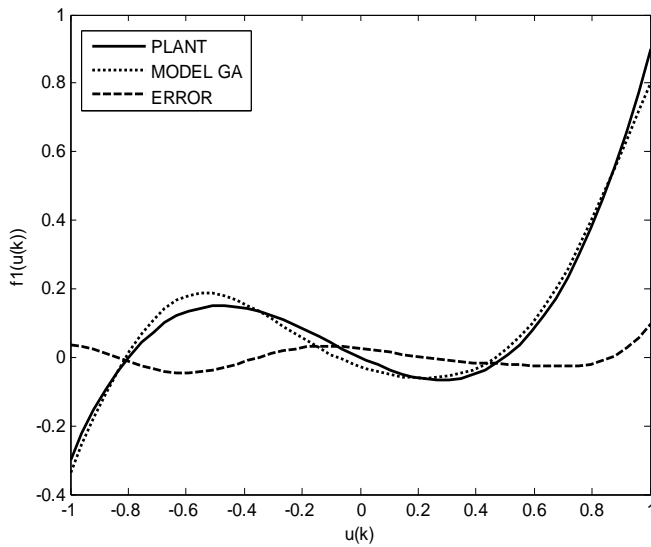


Fig.4.3 (c)

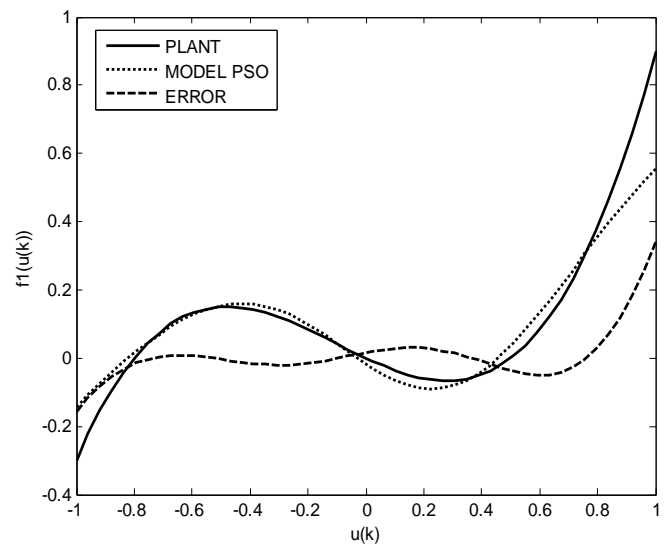


Fig.4.3 (d)

Fig.4.3. Identification of  $f_1$ : (a) using MLP-BP (b) using proposed FLANN-AIS (c) using FLANN-GA (d) using FLANN-PSO

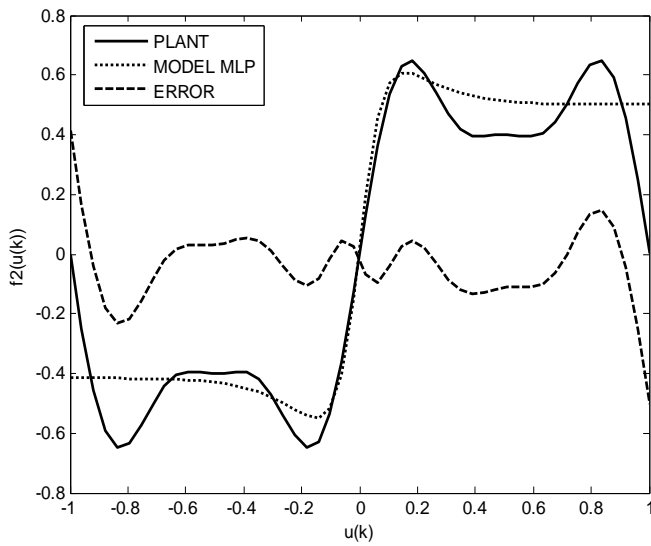


Fig.4.4 (a)

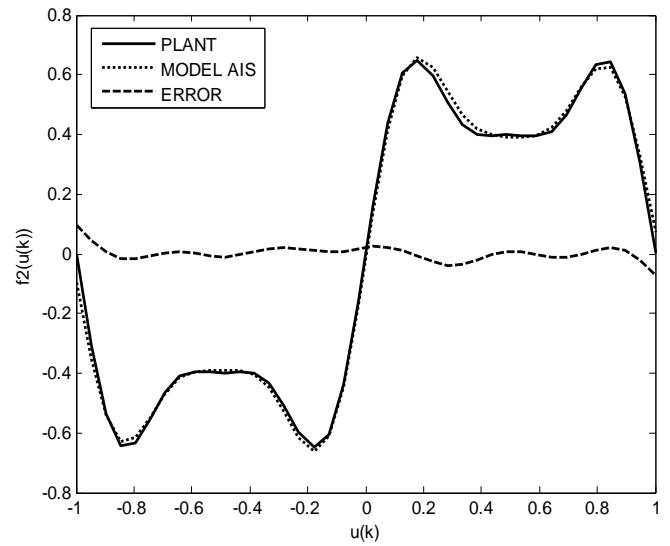


Fig.4.4 (b)

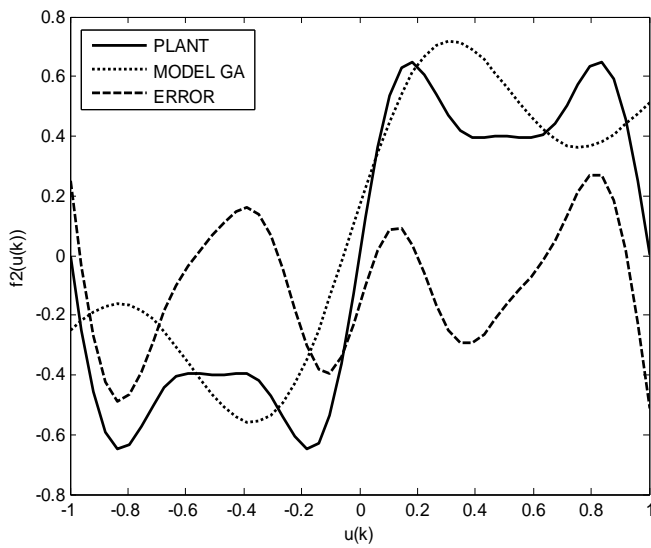


Fig.4.4 (c)

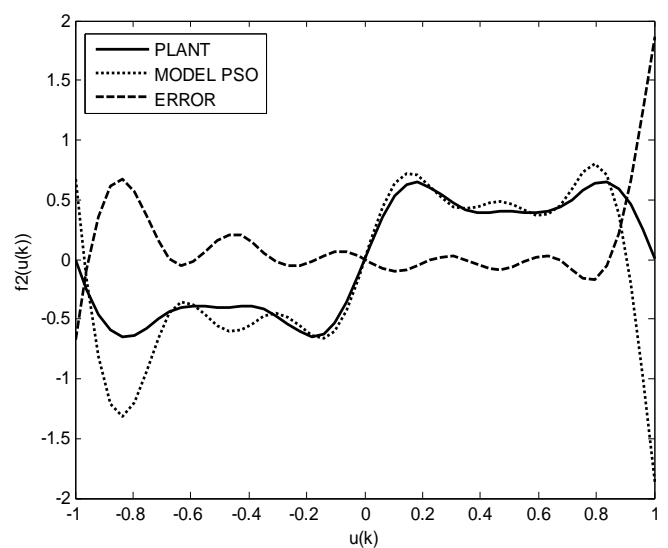


Fig.4.4 (d)

Fig.4.4. Identification of  $f_2$ : (a) using MLP-BP (b) using proposed FLANN-AIS (c) using FLANN-GA (d) using FLANN-PSO

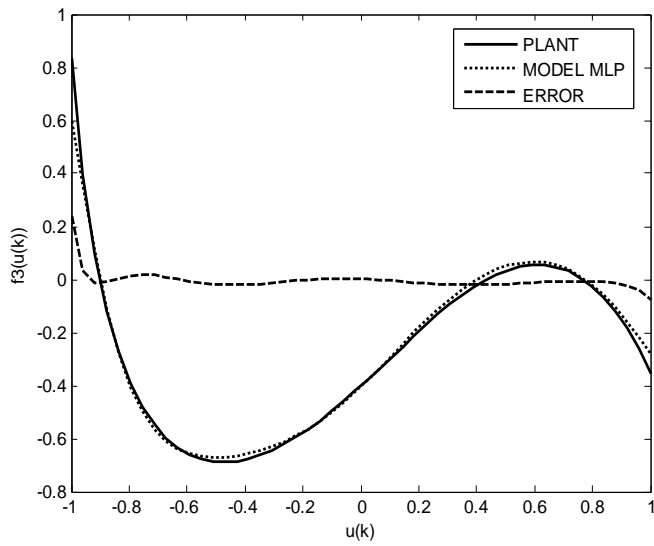


Fig.4.5 (a)

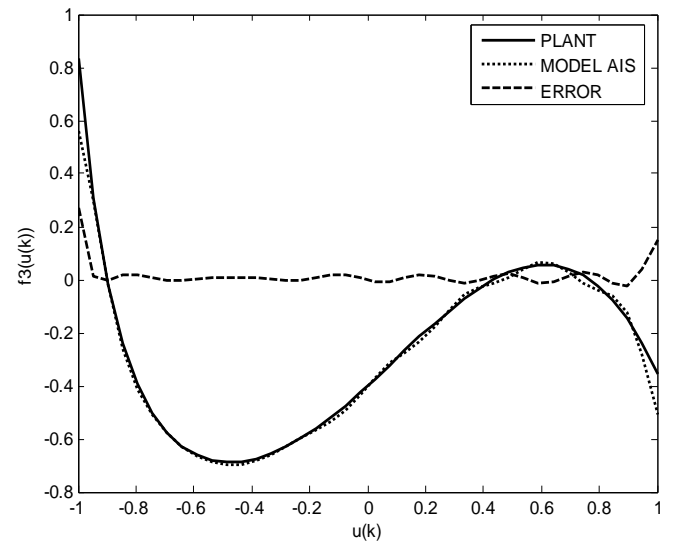


Fig.4.5 (b)

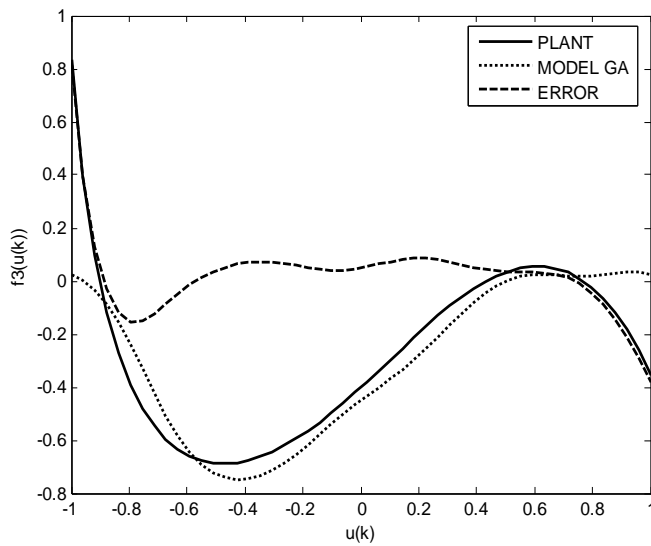


Fig.4.5 (c)

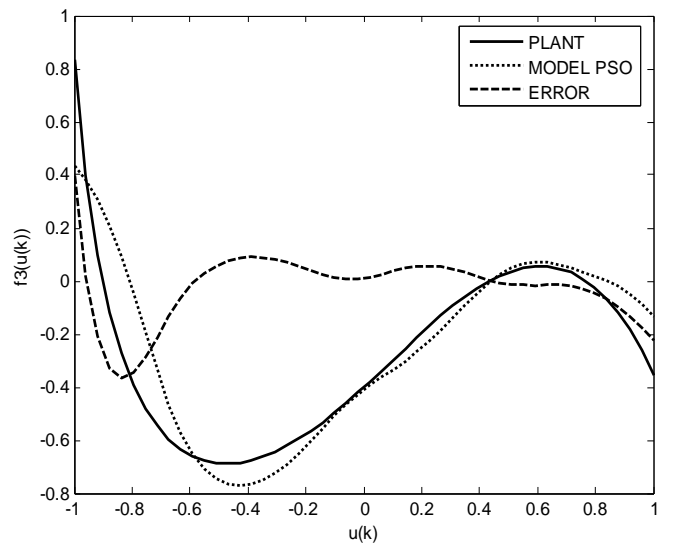


Fig.4.5 (d)

Fig.4.5. Identification of  $f_3$ : (a) using MLP-BP (b) using proposed FLANN-AIS (c) using FLANN-GA (d) using FLANN-PSO

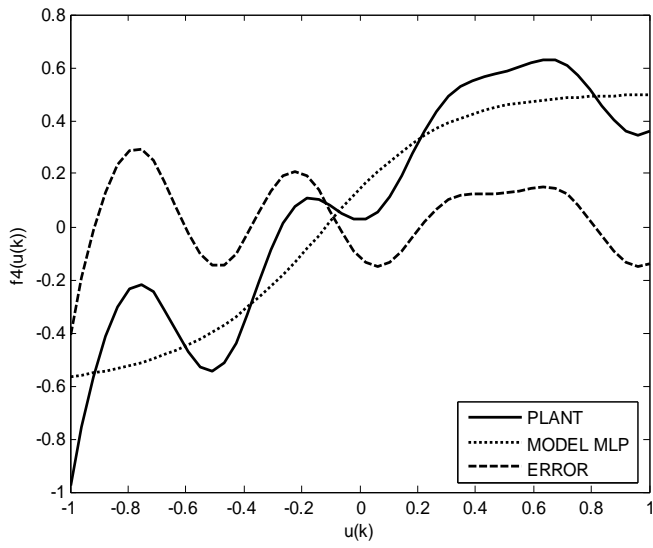


Fig.4.6 (a)

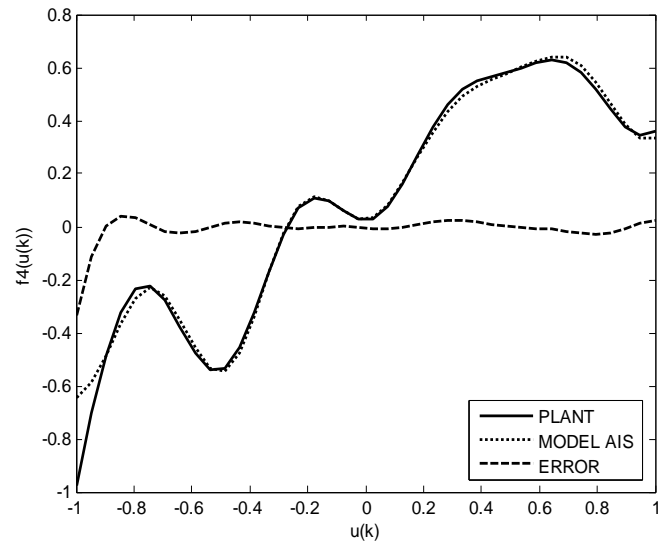


Fig.4.6 (b)

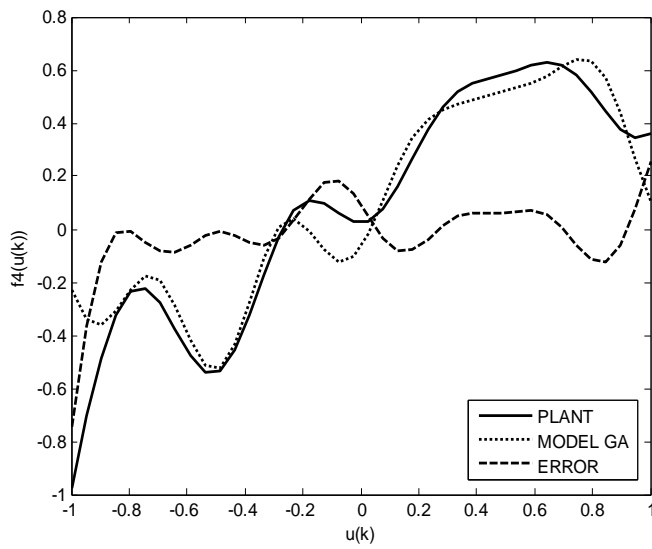


Fig.4.6 (c)

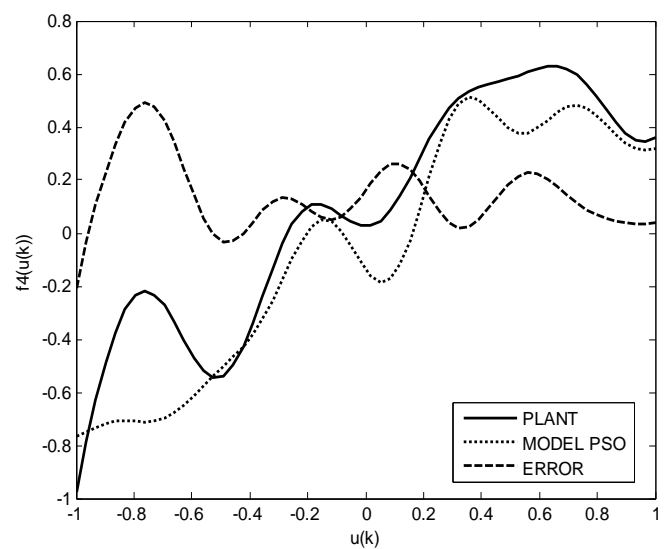


Fig.4.6 (d)

Fig.4.6. Identification of  $f_4$ : (a) using MLP-BP (b) using proposed FLANN-AIS (c) using FLANN-GA (d) using FLANN-PSO

---

## 4.6.2 Identification of Nonlinear Dynamic Plants

In simulation study the benchmark nonlinear plants considered here are taken from [4.3], [4.5]. In all the plants the input is a uniformly distributed random signal over the interval [-1, 1]. The testing is carried out by the sinusoidal input given by

$$u(k) = \begin{cases} \sin\left(\frac{2\pi k}{250}\right) & \text{for } k \leq 250 \\ 0.8\sin\left(\frac{2\pi k}{250}\right) + 0.2\sin\left(\frac{2\pi k}{25}\right) & \text{for } n > 250 \end{cases} \quad (4.24)$$

The different examples simulated are

### Example 1 :

This dynamic system is assumed to be of second order and is given by the difference equation

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + g[u(k)] \quad (4.25)$$

where function  $g[.]$  is given by

$$g(u) = 0.5\sin^3(\pi u) - \frac{2}{u^3 + 2} - 0.1\cos(4\pi u) + 1.125 \quad (4.26)$$

For identification of the plant the model used is of the form

$$\hat{y}(k+1) = 0.3y(k) + 0.6y(k-1) + N[u(k)] \quad (4.27)$$

where  $N[u(k)]$  represents either the proposed model or the MLP structure {1\_20\_10\_1}. For MLP both convergence factor and momentum term is taken as 0.1. The number of iteration and input sample for training used are 50000. The proposed FLANN-AIS based structure shown in Fig.4.7. is used for identification. In FLANN structure each input value is expanded to 5 trigonometric terms. Initial population of cell is taken as 20. The weights are trained for 200 iterations. The same

FLANN structure is also trained with GA and PSO in similar environment. The identification performance is compared in terms of estimated output and the error plot of different models shown in Fig.4.8.

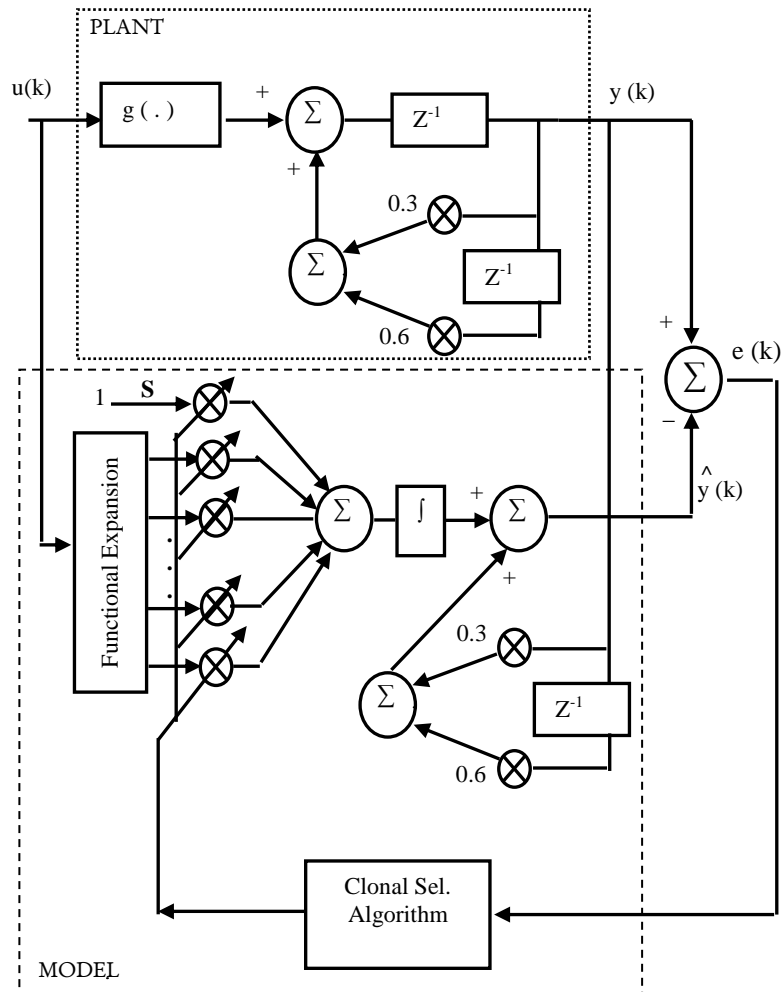


Fig.4.7. Proposed FLANN-AIS based model structure for identification of dynamic system of Example-1

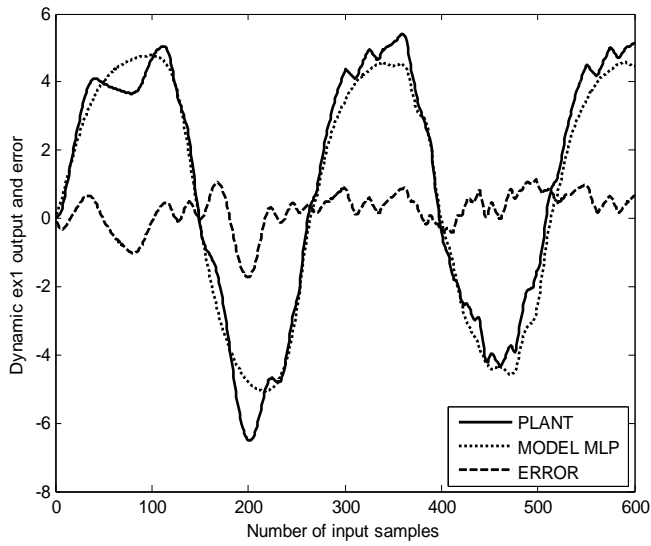


Fig.4.8 (a)

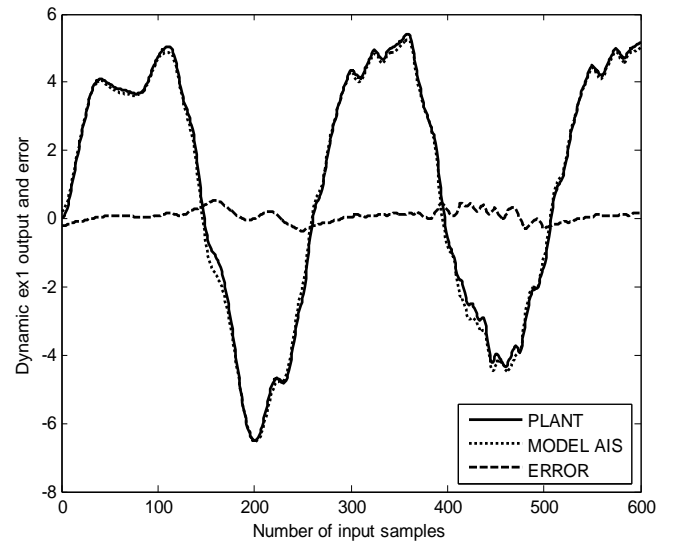


Fig.4.8 (b)

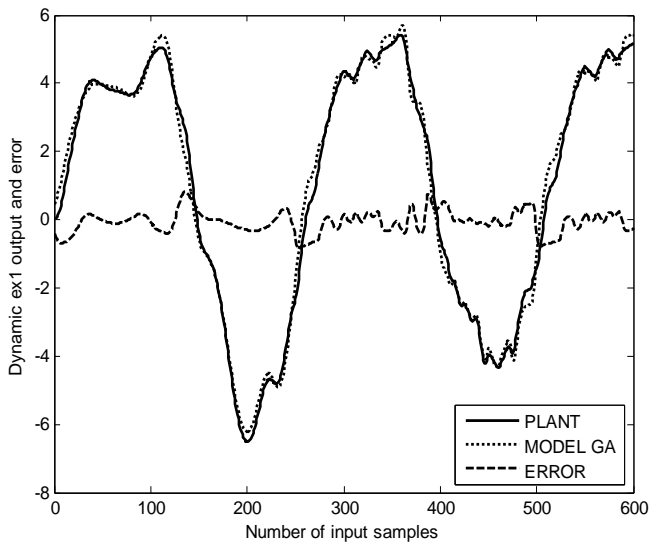


Fig.4.8 (c)

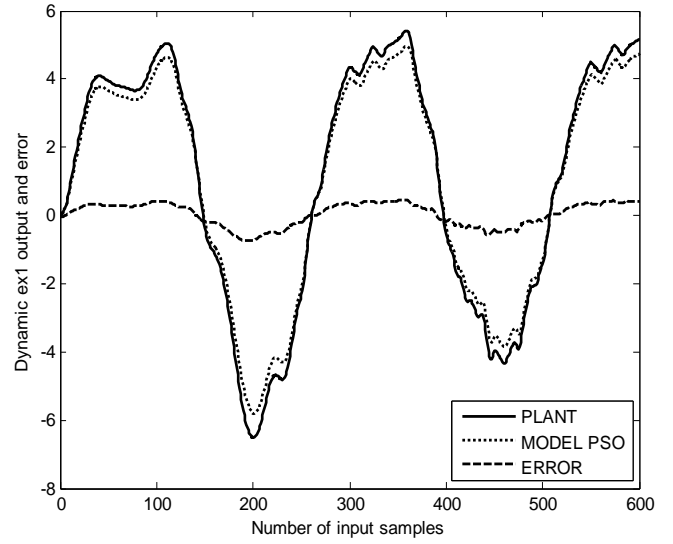


Fig.4.8 (d)

Fig.4.8. Identification of dynamic system Example-1 (a) using MLP-BP (b) using proposed FLANN-AIS (c) using FLANN-GA (d) using FLANN-PSO

---

### Example 2 :

The dynamic system to be identified is described by the difference equation of Model 2. The second order difference equation which represents the system is given by

$$y(k+1) = f[y(k), y(k-1)] + u(k) \quad (4.28)$$

The function  $f[.]$  is given by

$$f(y_1, y_2) = \frac{y_1 y_2 (y_1 + 2.5)(y_1 - 1.0)}{1.0 + y_1^2 + y_2^2} \quad (4.29)$$

The model used for identification task is represented by

$$\hat{y}(k+1) = N[y(k), y(k-1)] + u(k) \quad (4.30)$$

Here  $N$  represents the structure of either MLP {2\_20\_10\_1} or the new model structure of Fig.4.9. For MLP the convergence factor is taken as 0.05 and momentum term is 0.1. The number of iteration and input sample for training is taken 50000. In FLANN structure the inputs are expanded to 8 terms (each input 4 terms). Weights are trained for 20 iterations. Initial population of cells is taken as 60 in clonal selection algorithm. The same FLANN structure is also trained with GA and PSO in similar environment. The results of plant output, model output and error plot is shown in Fig.4.10.



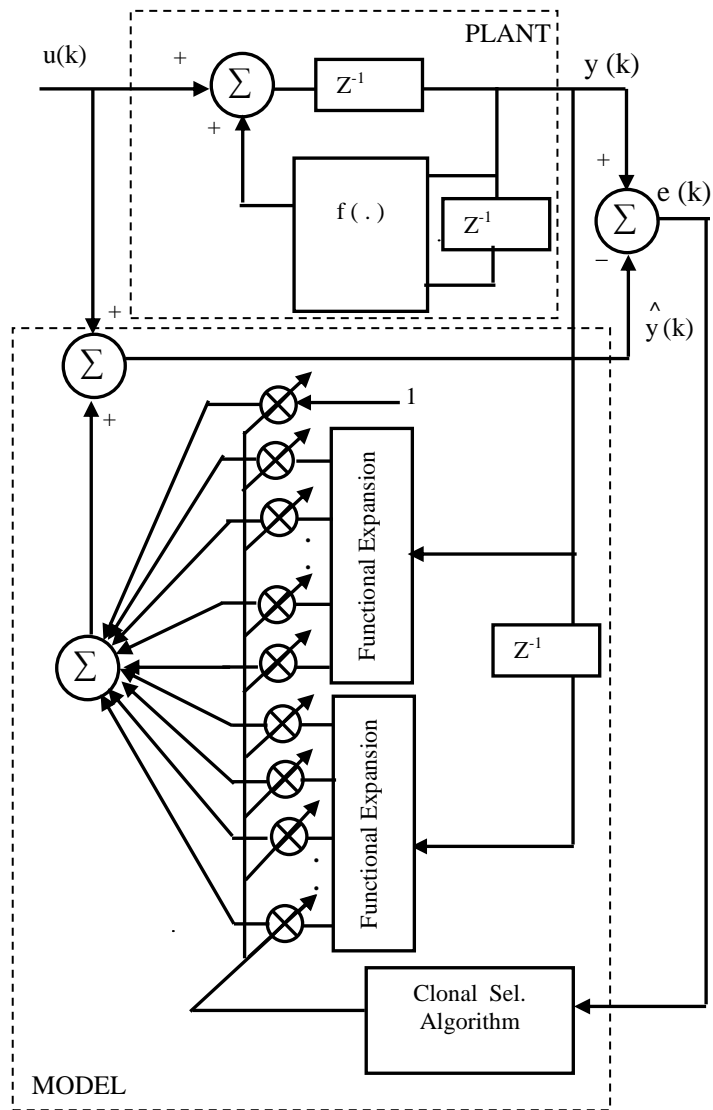


Fig.4.9. Proposed FLANN-AIS based model structure for identification of dynamic system of Example-2

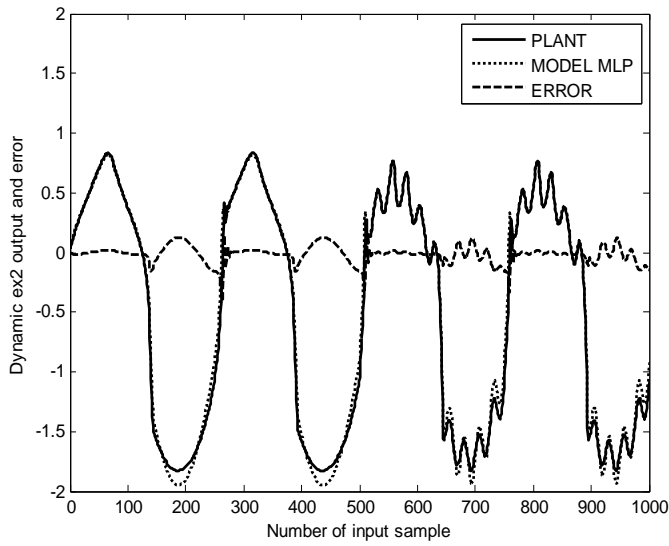


Fig.4.10 (a)

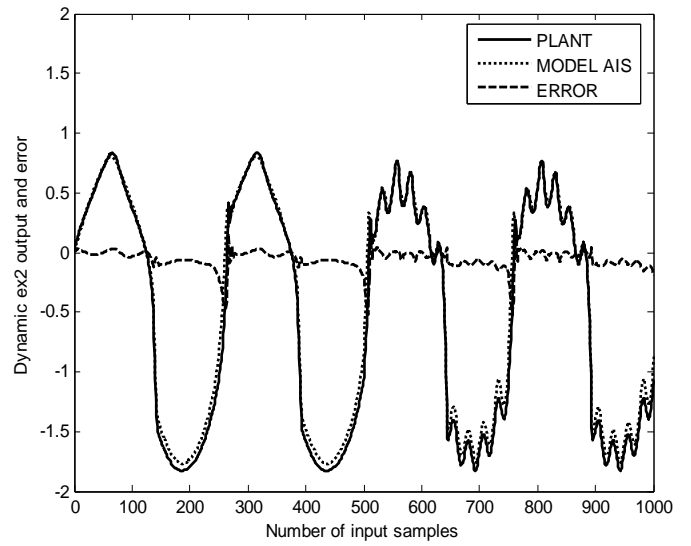


Fig.4.10 (b)

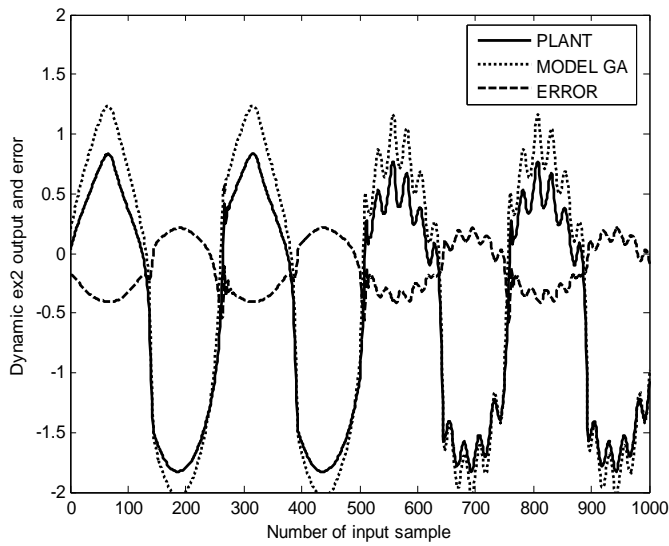


Fig.4.10 (c)

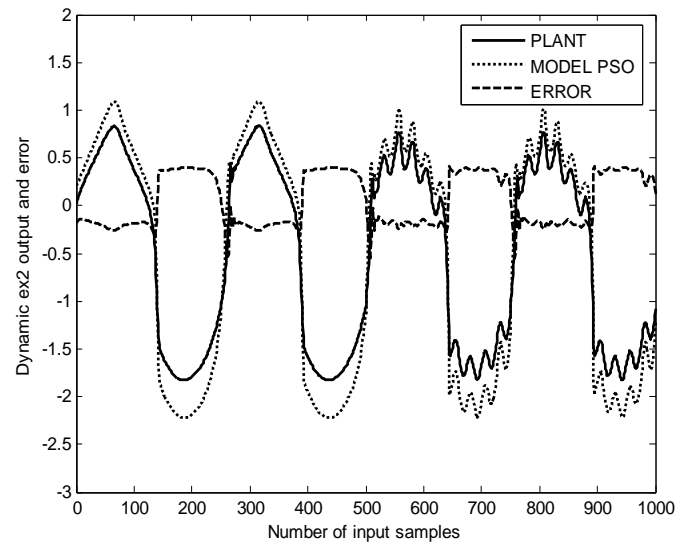


Fig.4.10 (d)

Fig.4.10. Identification of dynamic system Example-2 (a) using MLP-BP (b) using proposed FLANN-AIS (c) using FLANN-GA (d) using FLANN-PSO

---

### Example 3 :

The example of the plant taken here is described by the difference equation of Model 3 is represented by

$$y(k+1) = f[y(k)] + g[u(k)] \quad (4.31)$$

where the functions  $f[.]$  and  $g[.]$  is given by

$$f(y) = \frac{y(y+0.3)}{1.0+y^2} \quad (4.32)$$

$$g(u) = u(u+8)(u-0.5) \quad (4.33)$$

The model taken for identification is given by

$$\hat{y}(k+1) = N_1[y(k)] + N_2[u(k)] \quad (4.34)$$

where  $N_1$  and  $N_2$  represent two different FLANN structures.

For MLP both the structure of  $N_1[.]$  and  $N_2[.]$  are taken as {1\_20\_10\_1}. The convergence factor and momentum term both are taken as 0.1. Weights of the structure are trained for 50000 iterations. No of input samples taken is 50000. The proposed identification structure is shown in Fig.4.11. The layer of FLANN for  $N_1$  is taken as 4 where as 3 for  $N_2$ . No of input samples is 200 and training is carried out for 100 iterations. Initial population of cells is taken as 30 in the clonal selection algorithm. The weights of the FLANN structure are also updated with GA and PSO based algorithms and comparative results are presented in Fig.4.12.

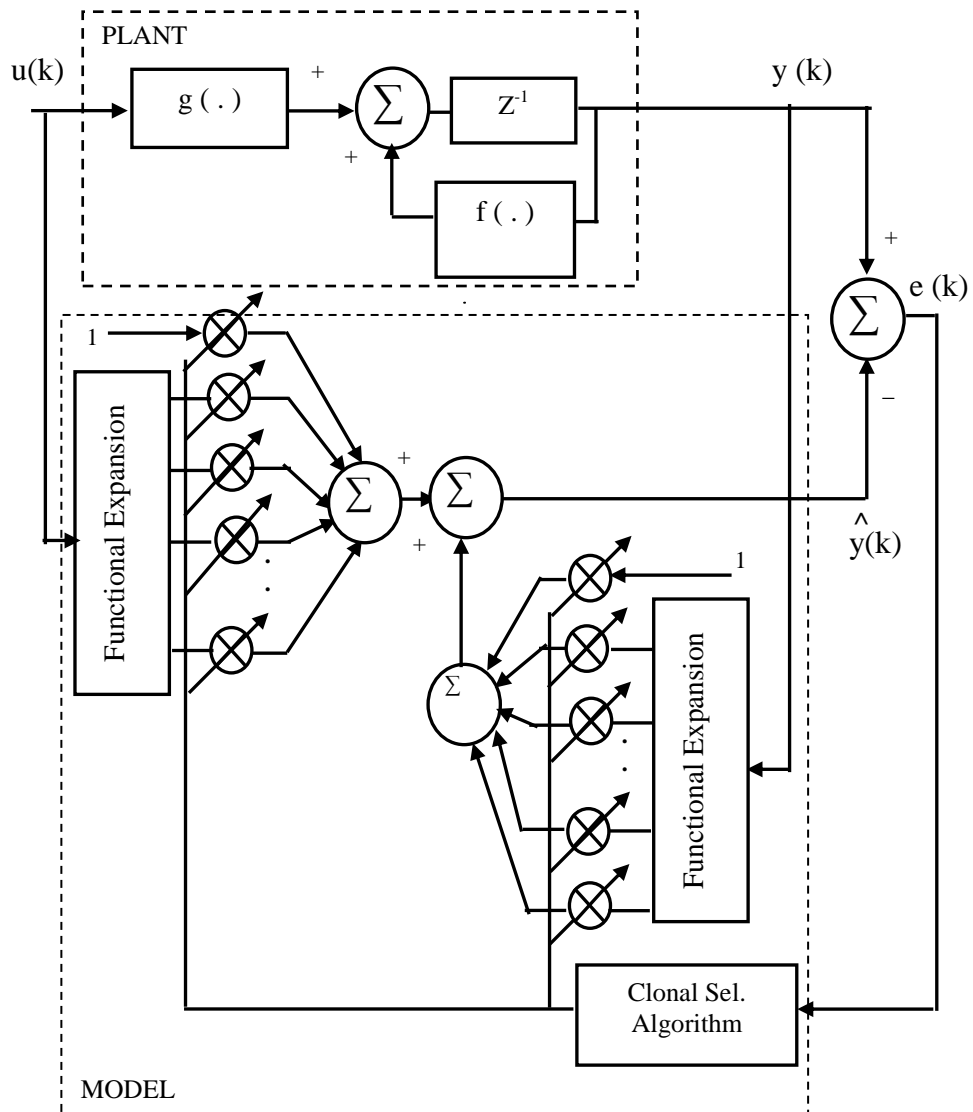


Fig.4.11 Proposed FLANN-AIS based model structure for identification of dynamic system of Example-3

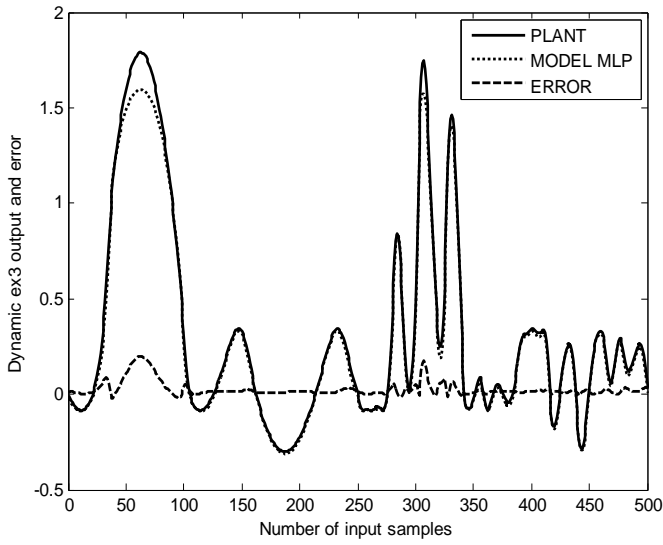


Fig.4.12 (a)

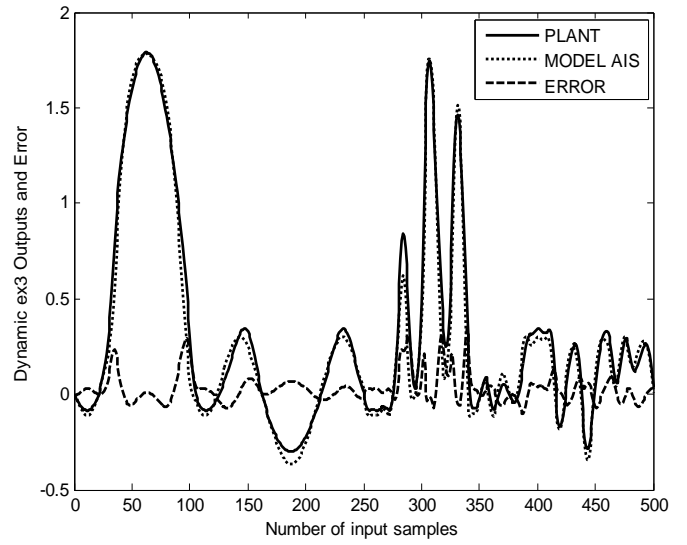


Fig.4.12 (b)

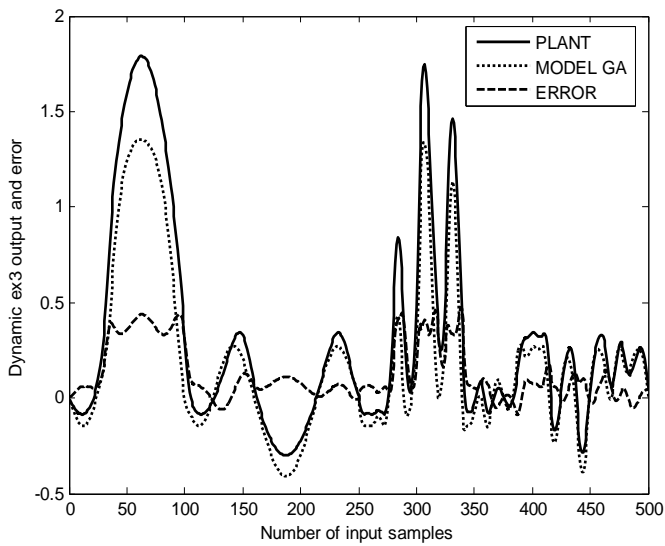


Fig.4.12 (c)

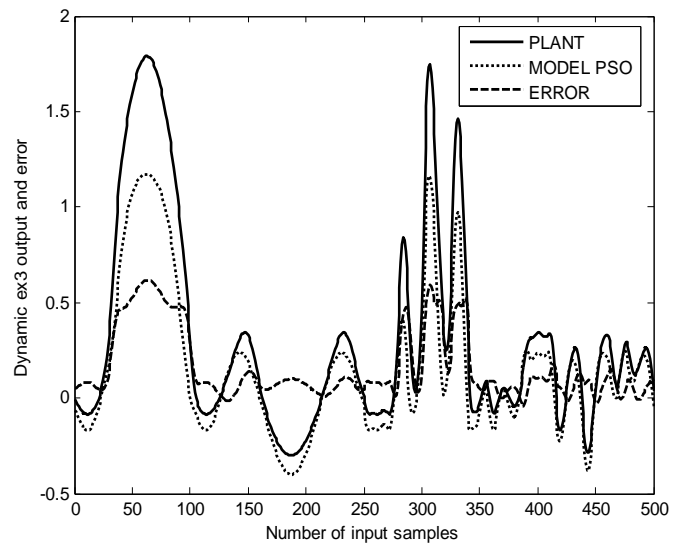


Fig.4.12 (d)

Fig.4.12. Identification of dynamic system Example-3 (a) using MLP-BP (b) using proposed FLANN-AIS (c) using FLANN-GA (d) using FLANN-PSO

---

**Example 4 :**

In this example the plant is of Model-4 type and is given by

$$y(k+1) = f[y(k), y(k-1), y(k-2), u(k), u(k-1)] \quad (4.35)$$

where the functions  $f[.]$  is represented as

$$f(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5) = \frac{\mathbf{a}_1 \mathbf{a}_2 \mathbf{a}_3 \mathbf{a}_5 (\mathbf{a}_3 - 1.0) + \mathbf{a}_4}{1.0 + \mathbf{a}_2^2 + \mathbf{a}_3^2} \quad (4.36)$$

The model for identification of plant is of the form

$$\hat{y}(k+1) = N[y(k), y(k-1), y(k-2), u(k), u(k-1)] \quad (4.37)$$

For MLP structure of N is taken as {5\_20\_10\_1}. The convergence factor and momentum term each is taken as 0.1. No of iterations and input samples for training is taken as 50000. The proposed identification structure used is shown in Fig.4.13. In FLANN for N the inputs are expanded to 20 terms (each input 4 terms). The number of input sample taken is 60. Weights are trained for 150 iterations. Initial population of cells is taken as 20. The weights of the FLANN structure are also trained with GA and PSO based algorithms. The responses of various models are displayed in Fig.4.14.

**Comparative Result :**

The comparisons of identification performance obtained in different examples are demonstrated in Table 4.2. It reveals that the proposed models offer smaller CPU time, lesser input samples and minimum sum squared errors compared to those obtained by other methods.

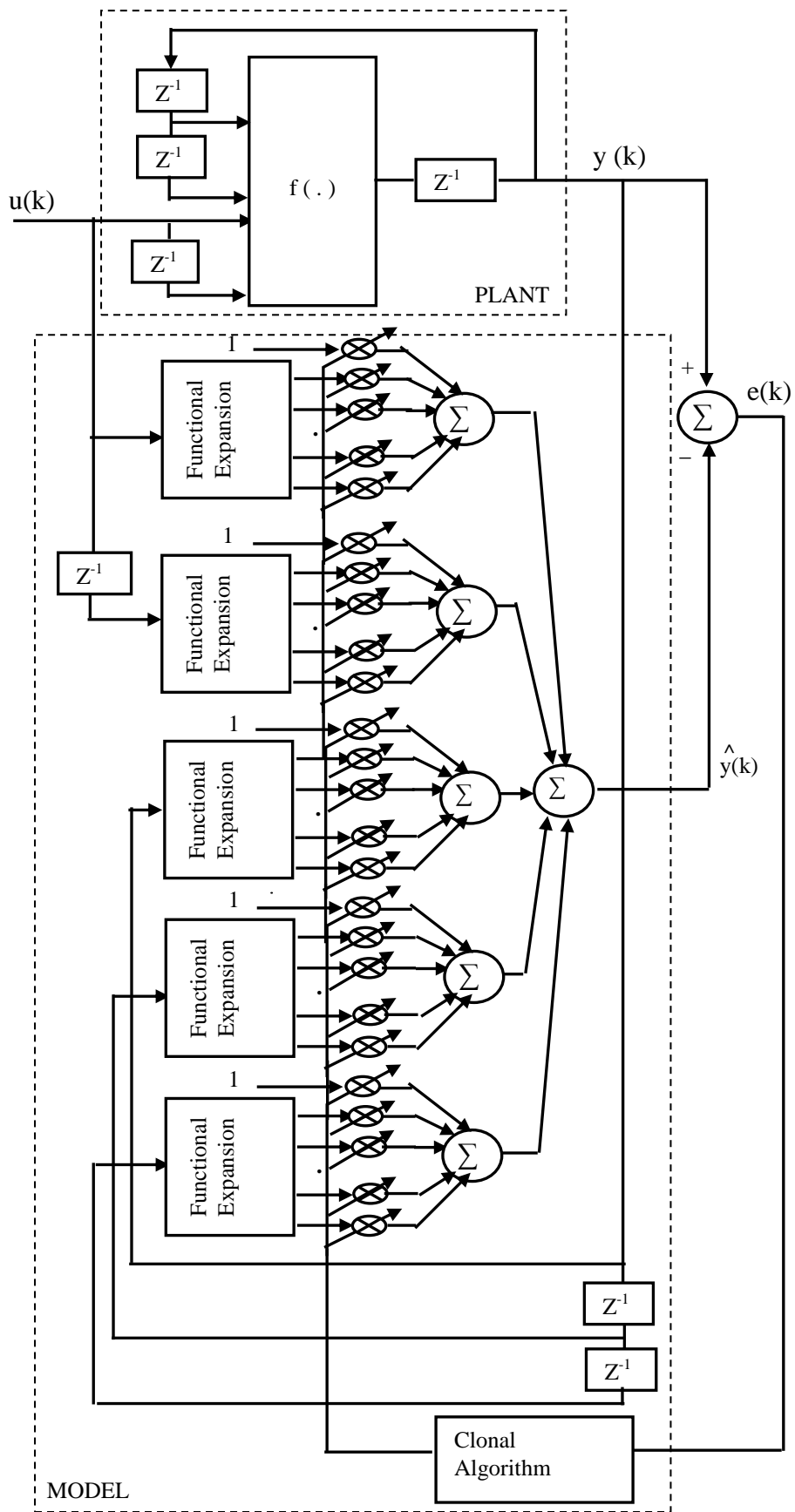


Fig.4.13. Proposed FLANN-AIS based structure for identification of dynamic system of Example-4

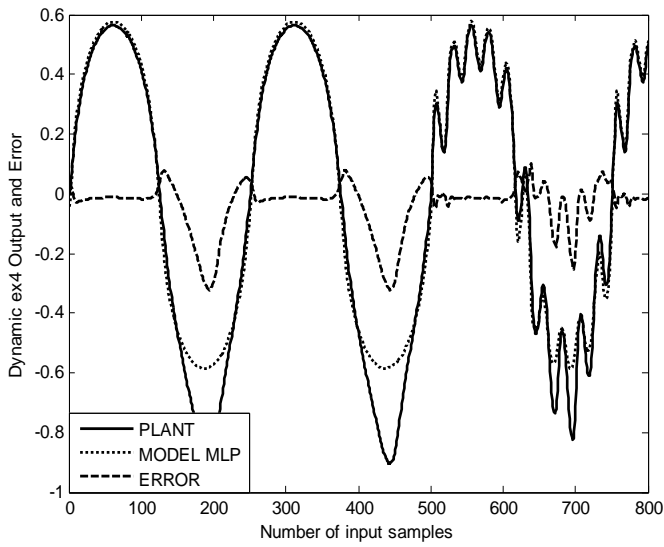


Fig.4.14 (a)

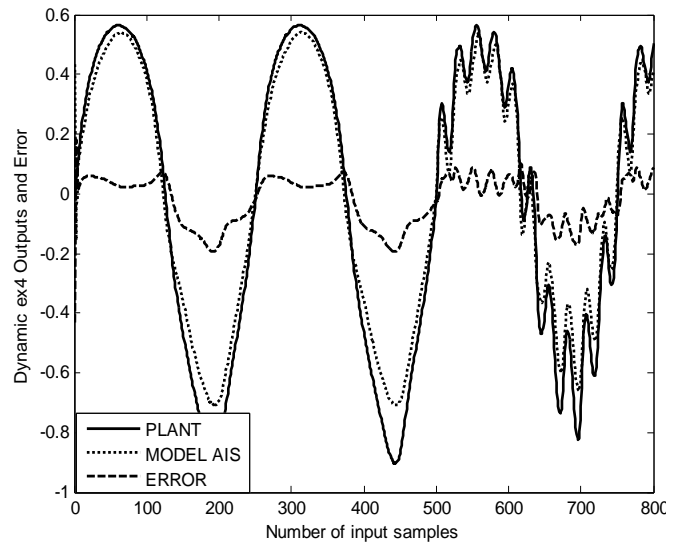


Fig.4.14 (b)

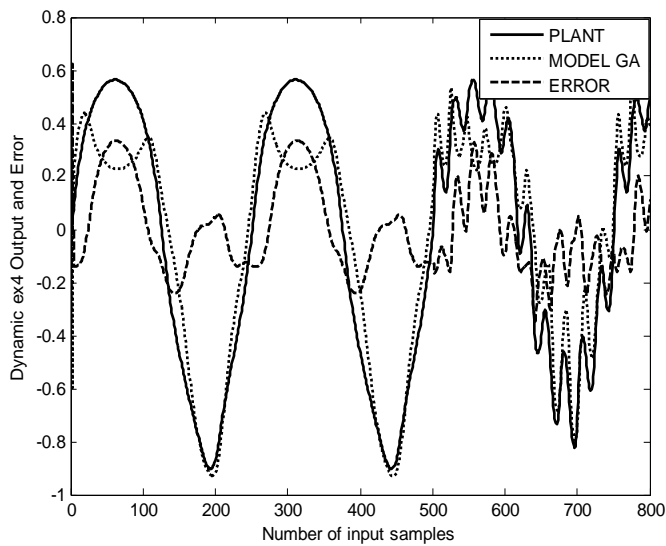


Fig.4.14 (a)

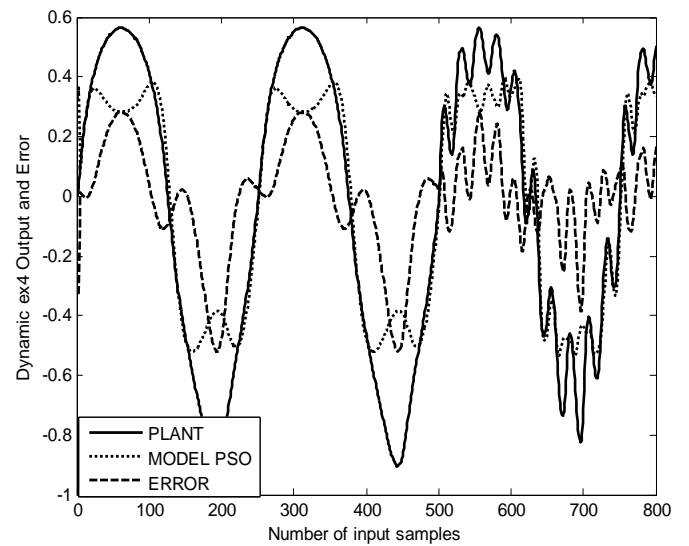


Fig.4.14 (b)

Fig.4.14. Identification of dynamic system Example-4 (a) using MLP-BP (b) using proposed FLANN-AIS (c) using FLANN-GA (d) using FLANN-PSO



TABLE 4.2

COMPARATIVE RESULTS OF DYNAMIC SYSTEMS OBTAINED THROUGH SIMULATION STUDY

Dynamic systems	Training						Testing				
	No. of input samples used by structures		CPU time required during Training (in Sec.)				No. of input samples used by both structures	Sum of square errors (SSE)			
	MLP	FLANN	MLP BP	FLANN AIS	FLANN GA	FLANN PSO		MLP BP	FLANN AIS	FLANN GA	FLANN PSO
Ex-1	50000	100	131.5	2.3	3.9	2.6	600	220.5	37.30	66.20	73.85
Ex-2	50000	150	114.8	8.8	16.6	9.1	1000	9.08	5.60	64.89	81.06
Ex-3	50000	200	92.0	23.4	42.4	27.8	500	1.16	1.99	19.00	30.20
Ex-4	50000	150	90.0	20.0	36.3	22.1	800	7.40	5.80	23.11	27.15

### 4.6.3 Identification of Nonlinear MIMO Plants

The example taken here from [4.21] is a two input two output nonlinear discrete time system given by

$$\begin{bmatrix} \mathbf{X}_1(k+1) \\ \mathbf{X}_2(k+1) \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{X}_2(k)}{1 + \mathbf{X}_1^2(k)} \\ \frac{\mathbf{X}_1(k)}{1 + \mathbf{X}_1^2(k)} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} + \begin{bmatrix} d_1(k) \\ d_2(k) \end{bmatrix} \quad (4.38)$$

where the inputs  $u_1(k)$  and  $u_2(k)$  is given by

$$u_1(k) = \cos\left(\frac{2\pi k}{100}\right) \quad (4.39)$$

$$u_2(k) = \sin\left(\frac{2\pi k}{100}\right) \quad (4.40)$$

The symbols  $d_1(k)$  and  $d_2(k)$  represent white Gaussian noise with zero mean and standard deviation (0.03). For identification the model used is of the form

$$\hat{\mathbf{x}}_1(k+1) = f_1[x_1(k), x_2(k), u_1(k), u_2(k)] \quad (4.41)$$

---

$$\hat{x}_2(k+1) = f_2[x_1(k), x_2(k), u_1(k), u_2(k)] \quad (4.42)$$

For identification of the MIMO plant the MLP structure of is taken as {2\_20\_10\_2}. Each of the convergence factor and momentum term is taken as 0.1. No of iterations and input samples for training is taken 50000. The structure of the model for above plant is shown in Fig.4.15. Each input is expanded to 4 terms using trigonometric expansion. The weights of Fig4.15 are updated using clonal selection algorithm. The initial population of cell is taken as 77. The weights of the model are trained for 100 iterations. The weights of the proposed structure are also trained with GA and PSO based algorithms under similar conditions. Both outputs of the MIMO plant, estimated output of the models and error are plotted in Fig.4.16-4.17. The identification performance is compared in Table 4.3. The results indicate that the suggested approach yields lesser CPU time and least MMSE errors compared to other models.

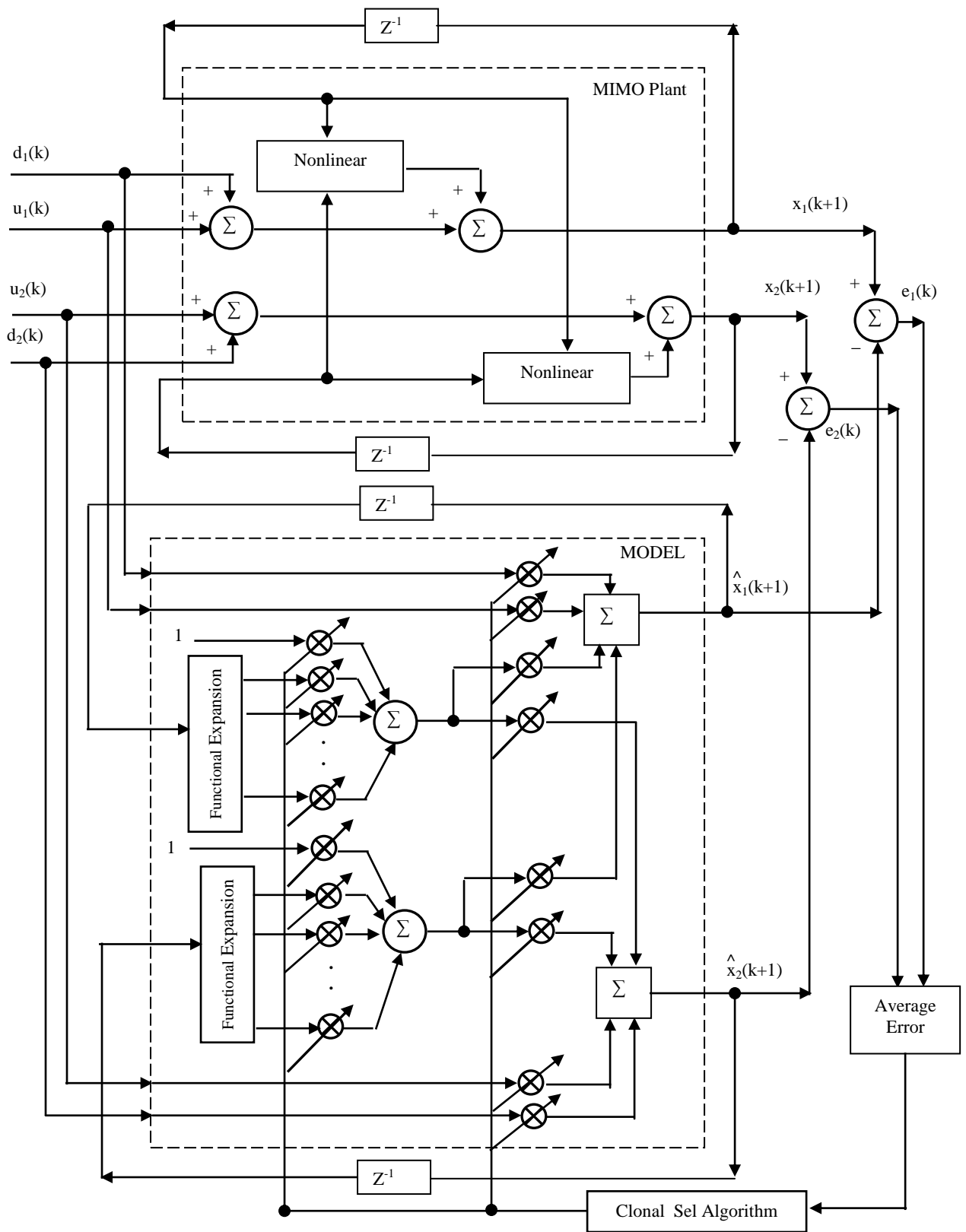


Fig.4.15. Proposed FLANN-AIS based structure for identification of MIMO Plant

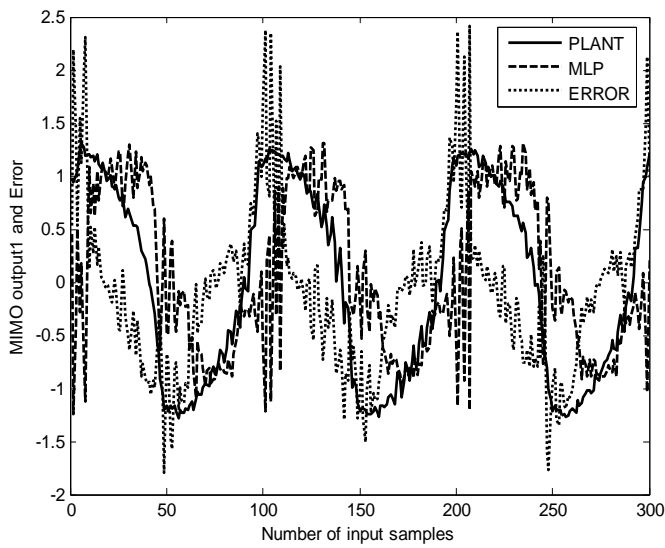


Fig.4.16 (a)

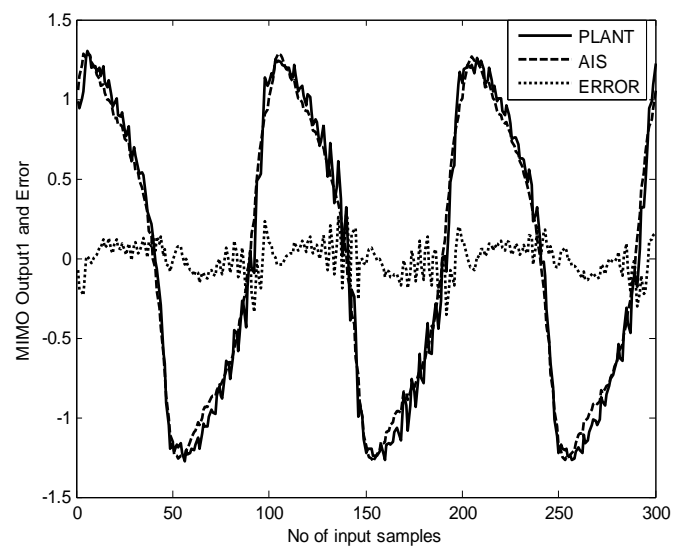


Fig.4.16 (b)

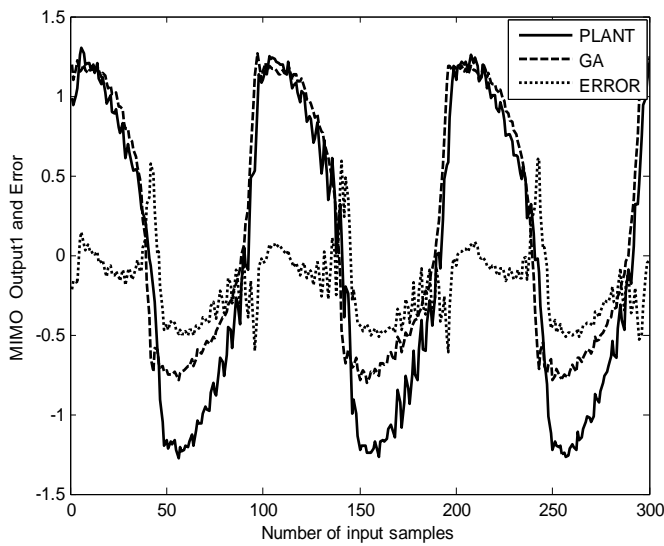


Fig.4.16 (c)

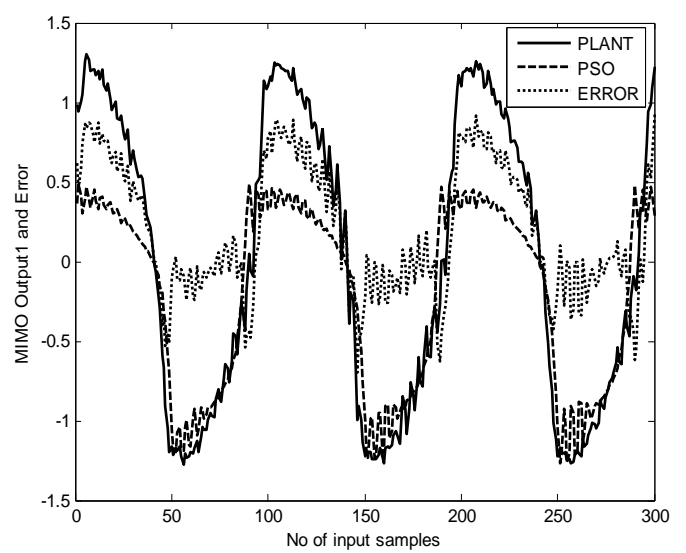


Fig.4.16 (d)

Fig.4.16. Identification of MIMO plant Output1: (a) using MLP-BP (b) using proposed FLANN-AIS (c) using FLANN-GA (d) using FLANN-PSO

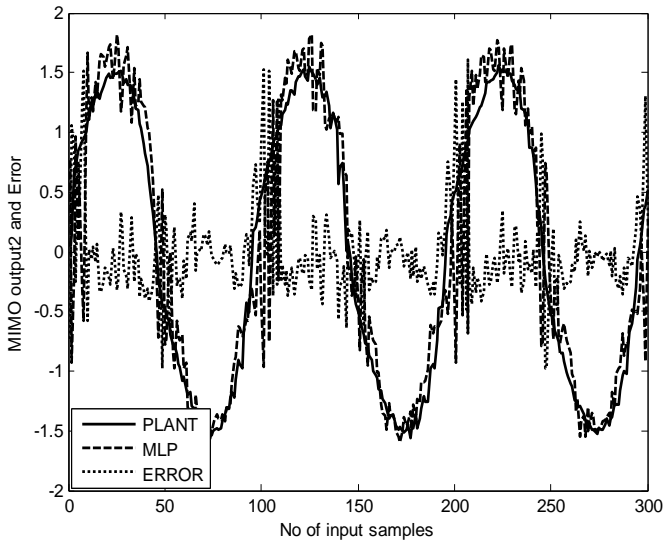


Fig.4.17 (a)

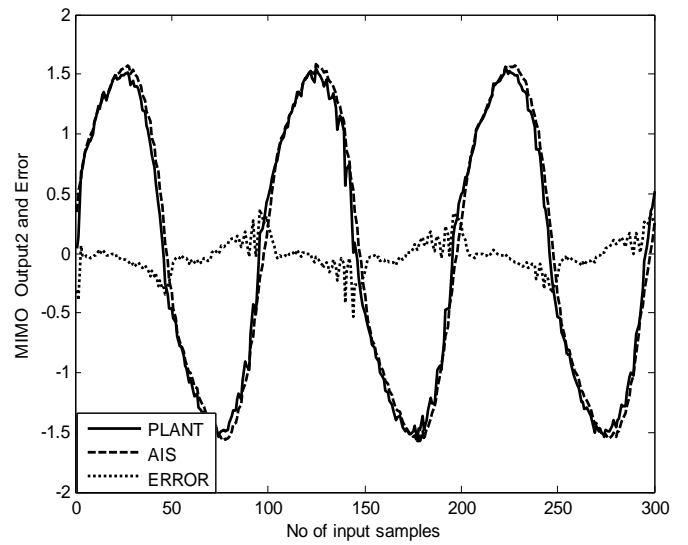


Fig.4.17 (b)

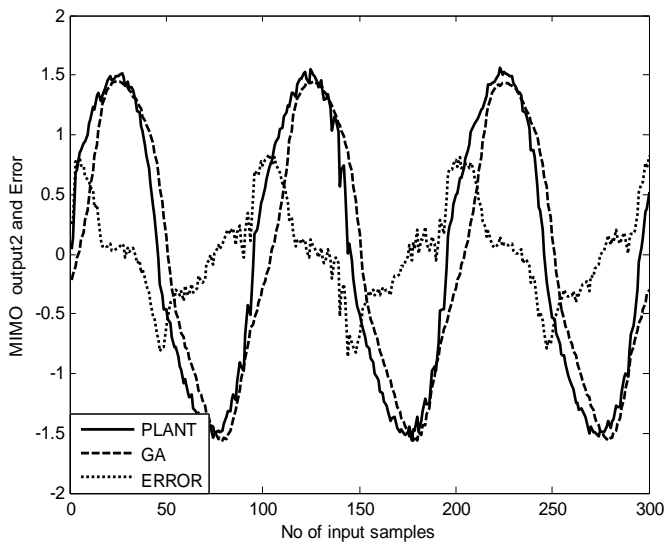


Fig.4.17 (c)

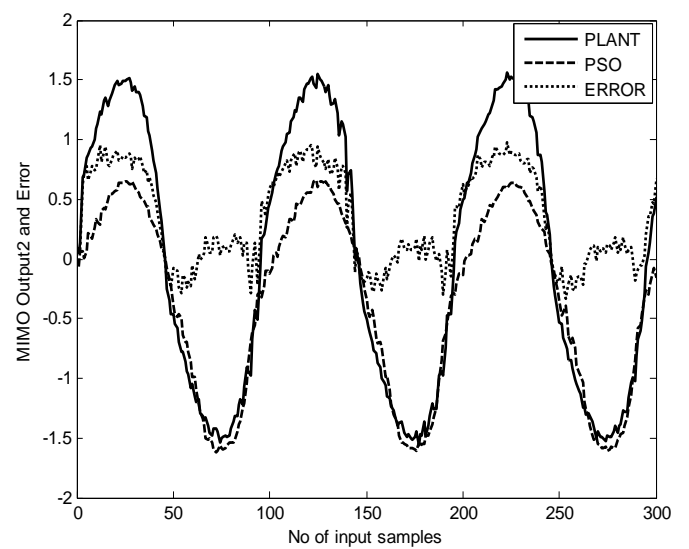


Fig.4.17 (d)

Fig.4.17. Identification of MIMO plant Output2: (a) using MLP-BP (b) using proposed FLANN-AIS (c) using FLANN-GA (d) using FLANN-PSO

TABLE 4.3

COMPARATIVE RESULTS OF MIMO SYSTEMS OBTAINED THROUGH SIMULATION STUDY

MIMO Plants		Training						Testing				
		No. of input samples used by structures		CPU time required during Training (in Sec.)				No. of input samples used by both structures	Sum of square errors (SSE)			
		MLP	FLANN	MLP BP	FLANN AIS	FLANN GA	FLANN PSO		MLP BP	FLANN AIS	FLANN GA	FLANN PSO
Ex	Out-1	50000	300	223.1	104.8	192.6	128.4	300	150.93	3.39	27.28	62.33
	Out-2								45.6	5.32	47.79	83.14

## 4.7 Conclusion

The proposed method requires less training sample compared to the MLP counterpart. It also involves lesser CPU time for learning the weights and smaller sum of squared errors. The simulation study also reveals that the Clonal selection algorithm is faster in training and more accurate (minimum sum of square errors) compared to GA and PSO base FLANN model. The close agreement of output responses of the plant and the model exhibit that the AIS is a potential learning tool for developing accurate identification model for complex nonlinear plants.

## 4.8 Summary

This chapter contains original contribution on system identification using AIS. In this chapter the nonlinear plants are classified into three categories such as static, dynamic and MIMO, depending upon its input-output relationship. The detail architecture of these plants is described. The identification model proposed here consists of a FLANN structure whose weights are trained with CLONAL selection

---

principle of AIS. Simulation study is carried out on some benchmark identification problems. The proposed model is compared with other standard models like MLP with back propagation and FLANN structure with different evolutionary algorithms like GA, PSO etc. The efficiency of identification is determined by comparison of overall output response of the plant and the estimated model, sum of square errors (SSE) and overall CPU time required to train the model.

**Chapter**  
**5**

**Application of AIS to Nonlinear  
Channel Equalization**



---

## 5.1 Introduction

**T**RANSMISSION and storing of high density digital information plays an important role in the present age of communication and information technology. These data are distorted while reading out of the recording medium or arriving at the receiver end due to inter symbol interference in the channel. The adaptive channel equalizer alleviates this distortion and reconstructs the transmitted data faithfully.

## 5.2 Need of Adaptive Channel Equalization

Digital Communication channels are often modeled as low pass FIR filter. When a sequence of symbols is transmitted, the low pass filtering effect of the channel distorts the transmitted symbols over successive time intervals causing symbols to spread and overlap with adjacent symbols. This resulting linear distortion is known as inter symbol interference (ISI). In addition to the linear distortion, the transmitted symbols are subjected to other impairments such as thermal noise, impulse noise and nonlinear distortion arising from the modulation/demodulation process, cross-talk interference, the use of amplifiers and converters, and the nature of the channel itself. Thus adaptive channel equalizers play an important role in recovering digital information from digital communication channels/storage media.

## 5.3 Background and problems associated with existing digital channel equalizers

Adaptive channel equalization was first proposed and analyzed by Lucky in 1965[5.1]. Adaptive channel equalizer employing a multilayer perceptron (MLP) structure has been reported [5.2], [5.4]. One of the major drawbacks of the MLP structure is the long training time required for generalization and thus, this network has very poor convergence speed which is primarily due to its multilayer architecture.

---

A single layer polynomial perceptron network (PPN) has been utilized for the purpose of channel equalization [5.3] in which the original input pattern is expanded using polynomials and cross-product terms of the pattern and then, this expanded pattern is utilized for the equalization problem. An ANN based equalization technique has been proposed [5.7] to alleviate the ISI present during read back from the magnetic storage channel. Recently Sun et al have reported [5.8] an improved Viterbi detector to compensate the nonlinearities and media noise. Preparta had suggested [5.9] a simple and attractive scheme for dispersal recovery of digital information based on the Discrete Fourier Transform. Subsequently Gibson et al have reported [5.10] an efficient nonlinear ANN structure for reconstructing digital signals from the corrupted ones. In a recent publication [5.11] the authors have proposed optimal preprocessing strategies for perfect reconstruction of binary signals from dispersive communication channels. Touri et al have developed [5.12] deterministic worst case frame work for perfect reconstruction of discrete data transmission through a dispersive communication channel. Thus in recent past new adaptive equalizers have been suggested using soft computing tools such as Artificial Neural Network (ANN), FLANN [5.14]. It has been reported that these methods are best suited for nonlinear and complex channels. Recently, Chebyshev Artificial Neural Network has also been proposed for nonlinear channel equalization [5.15]. The drawback of these equalizers are that during training, the estimated weights do not reach to their optimum values due to the mean square error (MSE) being trapped to local minimum. In other words true Weiner solution is not achieved because of gradient based training. When the channel is highly noisy and nonlinear in nature the gradient based techniques do not perform satisfactorily. To alleviate this problem this section a new AIS based derivative free method is proposed in this section to develop an efficient channel equalizer.

## **5.4 Baseband Communication System**

In an ideal communication channel, the received information is identical to that transmitted. However, this is not the case for real communication channels, where

signal distortions take place. A channel can interfere with the transmitted data through three types of distorting effects: power degradation and fades, multi-path time dispersions and background thermal noise. Equalization is the process of recovering the data sequence from the corrupted channel samples. A typical base band transmission system is depicted in Fig.5.1., where an equalizer is incorporated within the receiver.

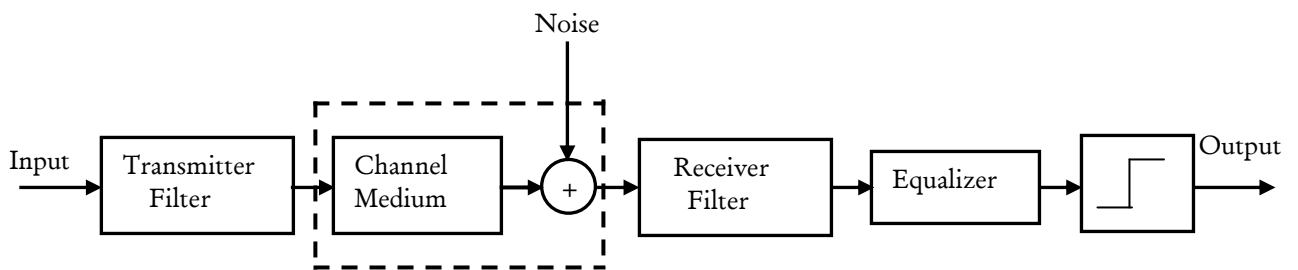


Fig.5.1. A baseband Communication System

## 5.5 Channel Interference

In a communication system data signals can either be transmitted sequentially or in parallel across a channel medium in a manner that can be recovered at the receiver. To increase the data rate within a fixed bandwidth, data compression in space and/or time is required. The interference occurs due to multipath propagation.

In telecommunication channels multiple paths of propagation commonly occur. Practically it means transmitting the same signal through a number of separate channels, each having a different attenuation and delay [5.17], [5.19]. Consider an open-air radio transmission channel that has three propagation paths, as illustrated in Fig.5.2 (a) [5.18]. These could be direct, earth and sky bound. The reception of the transmitted data at the receiver is shown in Fig.5.2 (b). The direct signal is received first whilst the earth and sky bound are delayed. All three of the signals are attenuated with the sky path suffering the most.

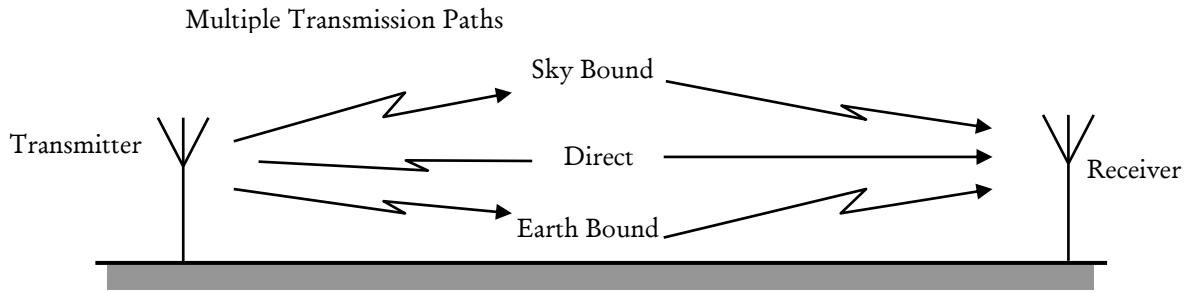


Fig.5.2 (a)

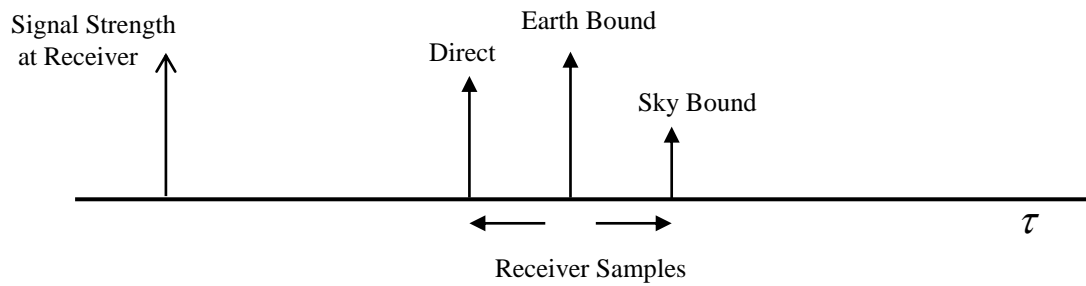


Fig.5.2 (b)

Fig.5.2. Impulse Response of a transmitted signal in a channel which has 3 modes of propagation (a) The signal transmitted paths, (b) The received samples.

Multipath interference between consecutively transmitted signals will take place if one signal is received while the previous signal is still being detected [5.18]. The case in Fig.5.2. will occur if the symbol transmission rate is greater than  $1/\tau$ . Because as bandwidth efficiency leads to high data rates the multi-path interference commonly occurs. The transfer function of a multi-path channel is given by

$$H(z) = \sum_{i=0}^m d(n-i)z^{-i} = d(n) + d(n-1)z^{-1} + d(n-2)z^{-2} + \dots \quad (5.1)$$

The model coefficients  $d(n-i)$  describe the strength of each multipath signal.

---

## 5.6 Intersymbol Interference

Inter-symbol interference (ISI) has already been described as the overlapping of the transmitted data. It is difficult to recover the original data from one channel samples because there is no statistical information about the multipath propagation. Increasing the dimensionality of the channel output vector helps characterize the multipath propagation. This has the effect of not only increasing the number of symbols but also increases the Euclidean distance between the output classes.

When additive Gaussian noise,  $\eta$ , is present within the channel, the input sample will form Gaussian clusters around the symbol centers. These symbol clusters can be characterized by a probability density function (pdf) with a noise variance  $\sigma_{\eta}^2$ , where the noise can cause the symbol clusters to interfere. Once this occurs, equalization filtering will become inadequate to classify all of the input samples. Error control coding schemes can be employed in such cases but these often require extra bandwidth.

The expected number of errors can be calculated by considering the amount of symbol interaction, assuming Gaussian noise. Taking any two neighboring symbols, the cumulative distribution function (CDF) can be used to describe the overlap between the two noise characteristics. The overlap is directly related to the probability of error between the two symbols and if these two symbols belong to opposing classes, a class error will occur.

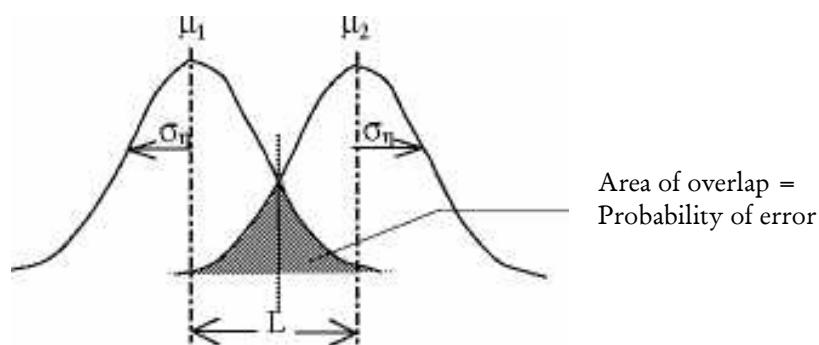


Fig.5.3. Interaction between two neighboring symbols

Fig.5.3 shows two Gaussian functions that could represent two symbol noise distributions. The Euclidean distance,  $L$ , between symbol centers and the noise variance,  $\sigma^2$ , can be used in the cumulative distribution function of (5.5) .

$$\text{CDF}(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{x^2}{2\sigma^2}\right] dx \quad (5.2)$$

The probability of error is given by

$$P(e) = 2 * \text{CDF}\left(\frac{L}{2}\right) \quad (5.3)$$

Since each channel symbol is equally likely to occur, the probability of unrecoverable errors occurring in the equalization space can be calculated using the sum of all the CDF overlap between each opposing class symbol. The probability of error is more commonly described as the BER given by

$$\text{BER}(\sigma_n) = \log\left[\frac{2}{N_{sp} + N_m} \sum_{i=1}^{N_{sp}} \text{CDF}\left(\frac{\Delta_i}{2\sigma_n}\right)\right] \quad (5.4)$$

where  $N_{sp}$  is the number of symbols in the positive class,  $N_m$  is the number of number of symbols in the negative class and,  $\Delta_i$  is the distance between the  $i$ th positive symbol and its closest neighboring symbol in the negative class.

## 5.7 Proposed AIS based Channel Equalizer

The basic block diagram of a digital channel equalizer with AIS based training is shown in Fig.5.4 in which the channel is considered as nonlinear in nature and is associated with additive white gaussian noise (AWGN). Since the equalizer is connected in series with the channel and its transfer function is inverse to the transfer function of the channel ( $1/H(z)$ ) where  $H(z)$  = channel transfer function. The symbols  $u(k)$ ,  $d(k)$ ,  $y(k)$  and  $e(k)$  represent the input from the data source, desired, estimated output of the equalizer and the error signal respectively.

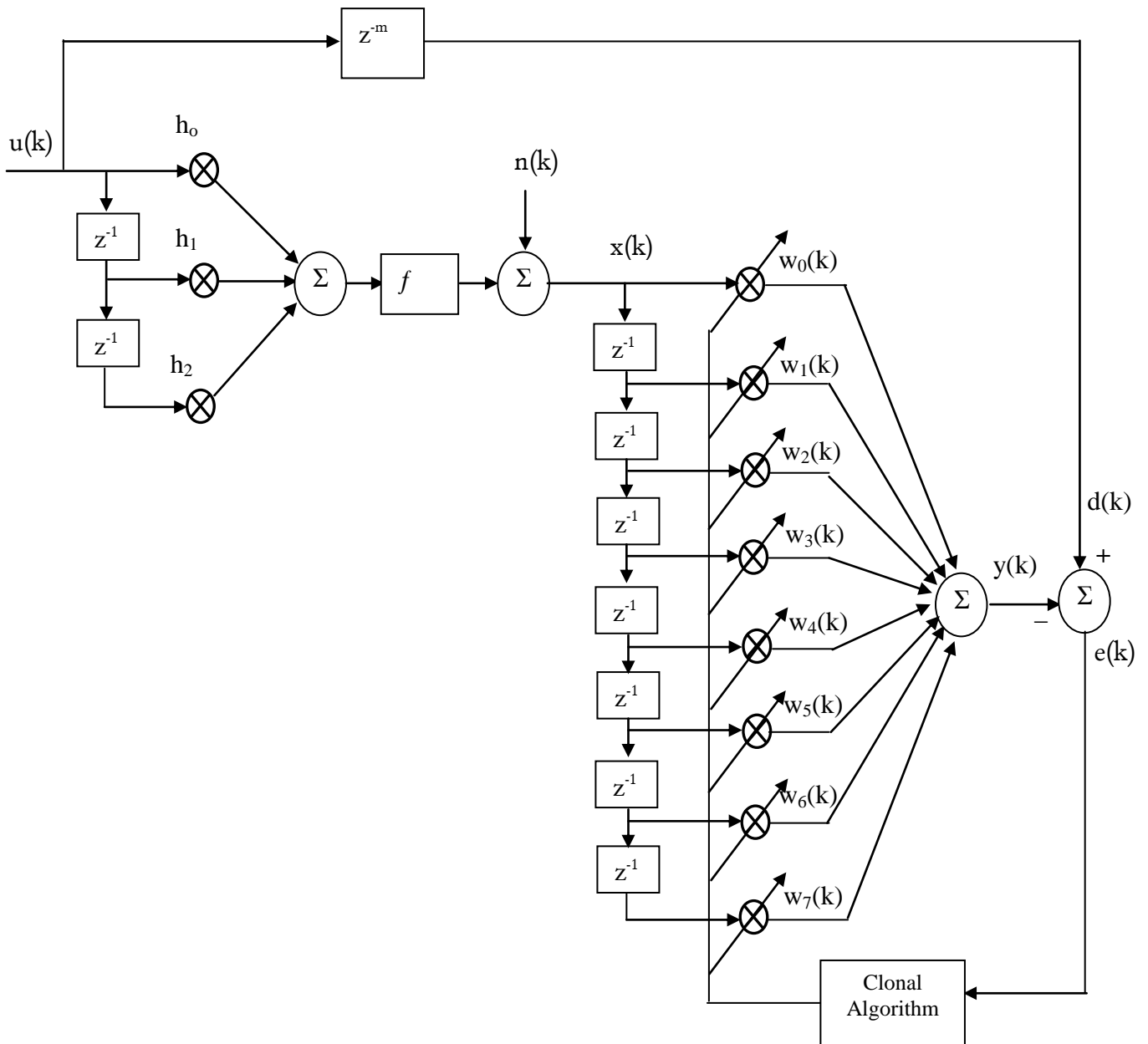


Fig 5.4. An AIS based adaptive digital channel equalizer

The output of the communication channel is represented by

$$x(k) = \sum_{k=1}^N f(h * U(k)) + n(k) \quad (5.5)$$

where  $h = \{h_0, h_1, h_2\}$  represents coefficient vector of the channel filter and  $U(k) = \{u(k), u(k-1), u(k-2)\}$  is the binary input vector applied. Symbols  $f$  and

$n(k)$  signifies the nonlinearity and AWGN associated with the channel. The symbol  $*$  denotes linear convolution operation and  $N$  is the number of taps of the channel filter. At the same  $k^{\text{th}}$  instant the output of the equalizer is given by

$$y(k) = \sum_{n=0}^{Q-1} x(k-n) * \bar{w}(k) \quad (5.6)$$

where  $Q$  is order of equalizer,  $\bar{w}(k)$  is the adaptive weight vector associated with it. The desired signal  $d(k)$  is formed by delaying the input sequence  $u(k)$  by  $m$  samples. In practice  $m$  is usually taken as  $(Q/2)$  or  $((Q+1)/2)$  depends on  $Q$  even or odd. The error signal  $e(k)$  is represented by

$$e(k) = d(k) - y(k) \quad (5.7)$$

The objective of designing an adaptive equalizer is to minimize the error  $e(k)$  recursively such that  $y(k)$  approaches  $d(k)$ . Here the mean square error (MSE) minimization is performed using clonal principle of AIS.

The steps involve in update of the weights of the equalizer are as follows :

### **Step 1. Determination of output of channel:**

The input is a random binary signal drawn from a uniform distribution. Let 'k' be the numbers of input samples taken. The input sample is then passed through the channel to produce output  $x(k)$  as given in (5.5).

### **Step 2. Equalizer input:**

The output of the channel  $x(k)$  is passed through the tap delay portion of the equalizer to produce the input vector.

### **Step 3. Initialization of a group of cells:**

Here a group of weight vector of equalizer is taken. A weight vector consists of  $Q$  no of elements. Each element of weight vector is represented by a cell which is basically a binary string of definite length. So a set of binary strings is initialized to represent a weight vector and  $n$  number of such weight vectors is taken each of which represent probable solution.



---

#### Step 4. Decoding :

As each cell constitute random binary bits so they need to be converted to decimal values lying between some ranges to compute the fitness function. The equation that converts the binary to real number is given by

$$RV = R_{\min} + \{(R_{\max} - R_{\min}) / (2^L - 1)\} \times DV \quad (5.8)$$

where  $R_{\min}$  ,  $R_{\max}$  ,  $RV$  and  $DV$  represent the minimum range, maximum range, decimal and decoded value of an  $L$  bit coding scheme representation.

#### Step 5. Calculation of desired output of the equalizer:

The desired signal  $d(k)$  is formed by delaying the input sequence  $u(k)$  by  $m$  samples.

#### Step 6. Fitness Evaluation:

The output of the equalizer  $y(k, n)$  due to  $k^{\text{th}}$  sample and  $n^{\text{th}}$  vector, is compared with the desired output  $d(k, n)$  to produce error signal given by

$$e(k, n) = d(k, n) - y(k, n) \quad (5.9)$$

For each  $n^{\text{th}}$  weight vector the mean square error (MSE) is determined and is used as fitness function given by

$$MSE(n) = \frac{\sum_{k=1}^K e^2(k, n)}{K} \quad (5.10)$$

**Step 7. Selection :** To select the weight vector (corresponding cells) for which MSE is minimum.

**Step 8. Clone:** The weight vector (corresponding cells) which yields best fitness value (minimum MSE) is duplicated.

**Step 9. Mutation:** Mutation operation introduces variations into the immune cells. Probability of mutation  $P_m$  is a smaller value which indicates that the operation occurs occasionally. Total number of bits to mutate is the product of total number of cells, number of bits in each cell and probability of mutation of each cell. Among the

cloned cells the cell to be mutated is chosen randomly. A random position of the cell is chosen first and then its bit value is altered.

**Step 10. Stopping Criteria:** The weight vector which provides the desired solution (minimum MSE) and corresponding cells are known as memory cells. Until a predefined MSE is obtained steps 4 -9 are repeated.

## 5.8 System Simulation

In designing of the new equalizer represented in Fig.5.4 is simulated using MATLAB for nonlinear channels whose linear part is given by either of

$$\text{CH1: } 0.2600 + 0.9300 Z^{-1} + 0.2600 Z^{-2} \quad (5.11)$$

$$\text{CH2: } 0.3040 + 0.9030 Z^{-1} + 0.3040 Z^{-2} \quad (5.12)$$

Each of the above channels is assumed to be associated with three different types of nonlinearities represented by

$$f_1(k) = \tanh(p(k)) \quad (5.13)$$

$$f_2(k) = p(k) + 0.2 * p^2(k) - 0.1 * p^3(k) \quad (5.14)$$

$$f_3(k) = p(k) + 0.2 * p^2(k) - 0.1 * p^3(k) + 0.5 * \cos(\pi p(k)) \quad (5.15)$$

where  $p(k)$  is the output of each of linear part of the channels(6). The additive noise is white Gaussian with -5dB and -10dB strengths. In this study a 8-tap adaptive FIR filter is used as an equalizer. The desired signal is generated by delaying the input binary sequence by half of the order (four samples in this case) of the equalizer. For simulation of AIS the no of input sample taken is 80. Weights are trained for 190 iterations. Initial population of cells is taken as 20. For simulating GA based equalizer the no of input sample taken is 80. Weights are trained for 190 iterations. Initial population of chromosome is taken as 20.

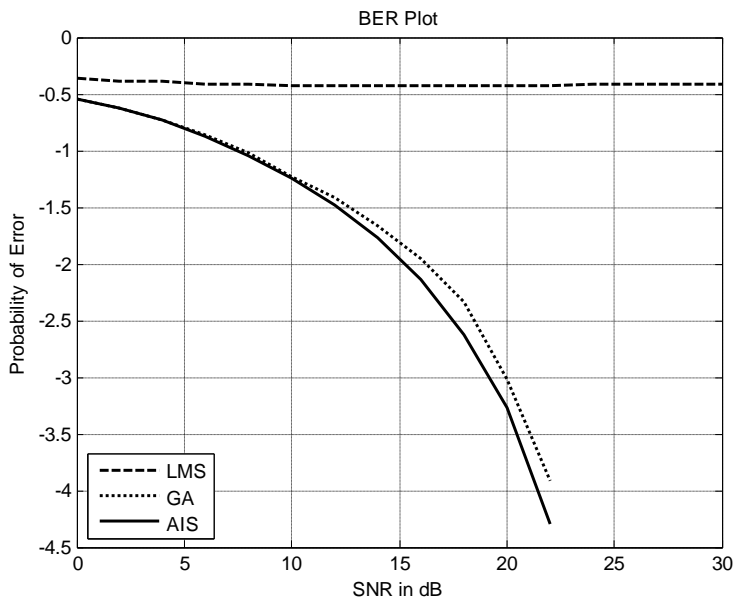


Fig.5.5 (a)

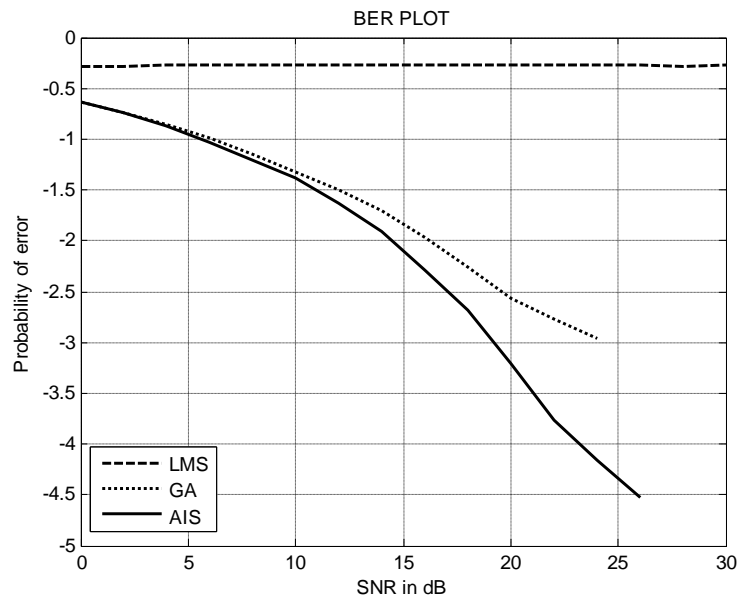


Fig.5.5 (b)

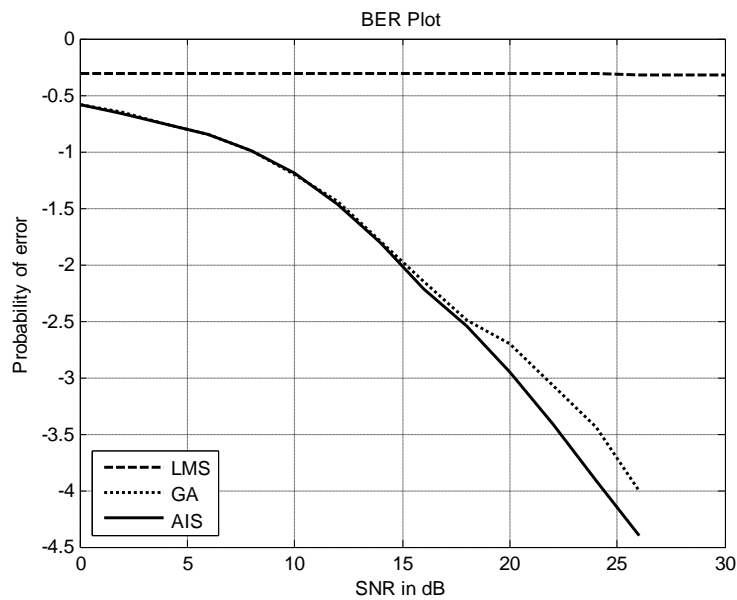


Fig.5.5 (c)

Fig.5.5 BER at 5dB SNR: (a) using  $f_1$  (b) using  $f_2$  (c) using  $f_3$  nonlinearity based channel CH1

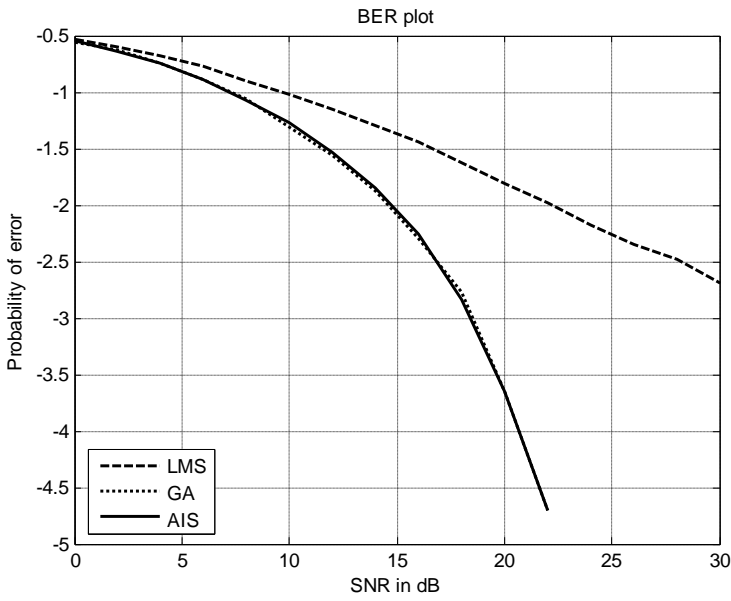


Fig.5.6 (a)

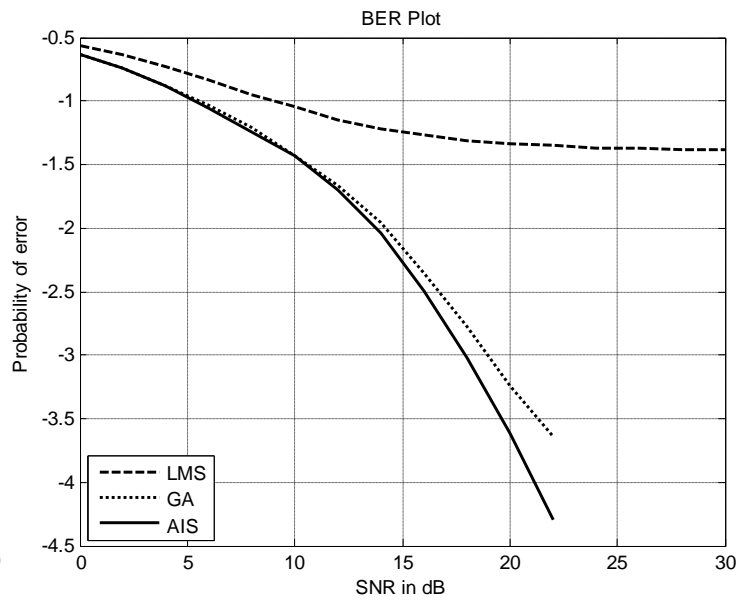


Fig.5.6 (b)

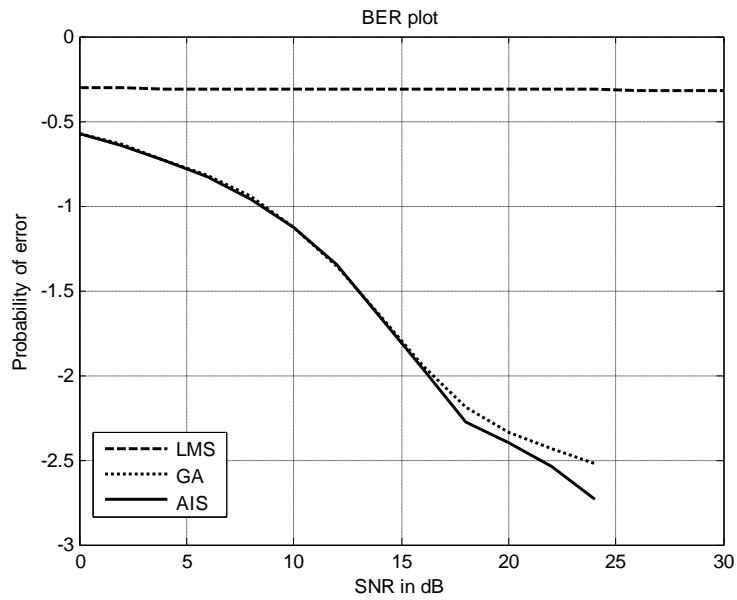


Fig.5.6 (c)

Fig.5.6 BER at 10dB SNR: (a) using  $f_1$  (b) using  $f_2$  (c) using  $f_3$  nonlinearity based channel CH1

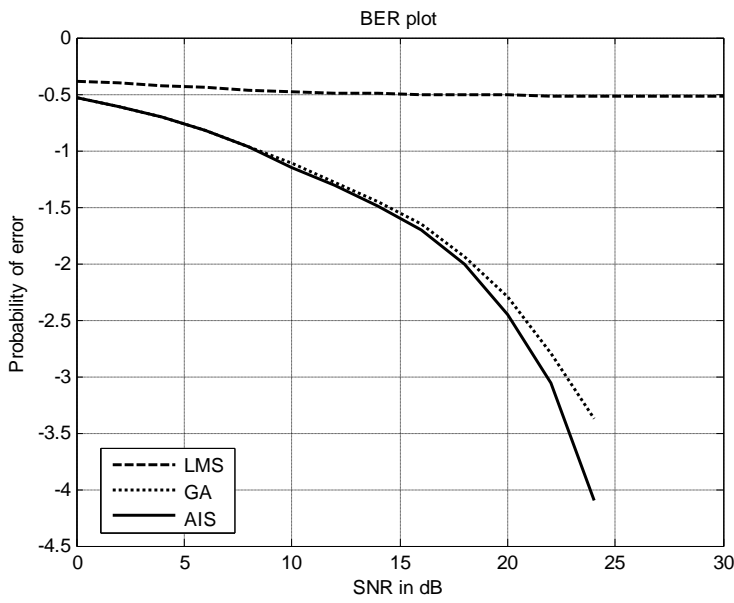


Fig.5.7 (a)

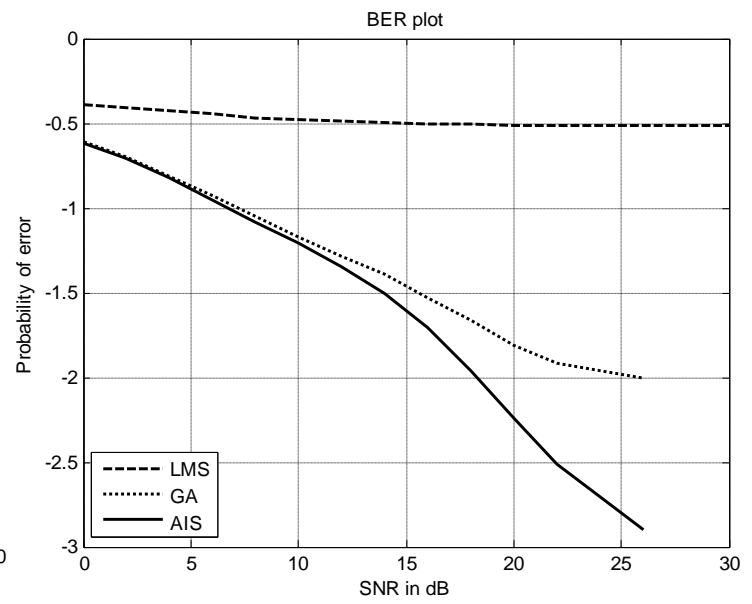


Fig.5.7 (b)

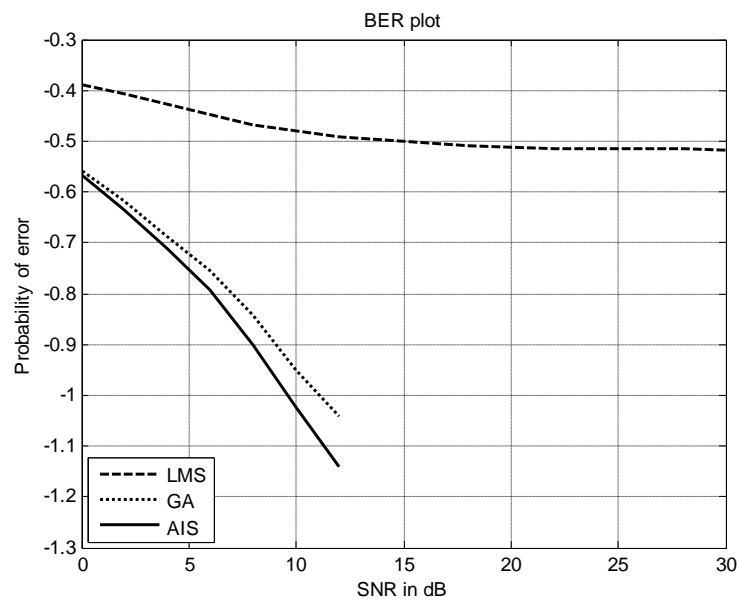


Fig.5.7 (c)

Fig.5.7 BER at 5dB SNR: (a) using  $f_1$  (b) using  $f_2$  (c) using  $f_3$  nonlinearity based channel CH2

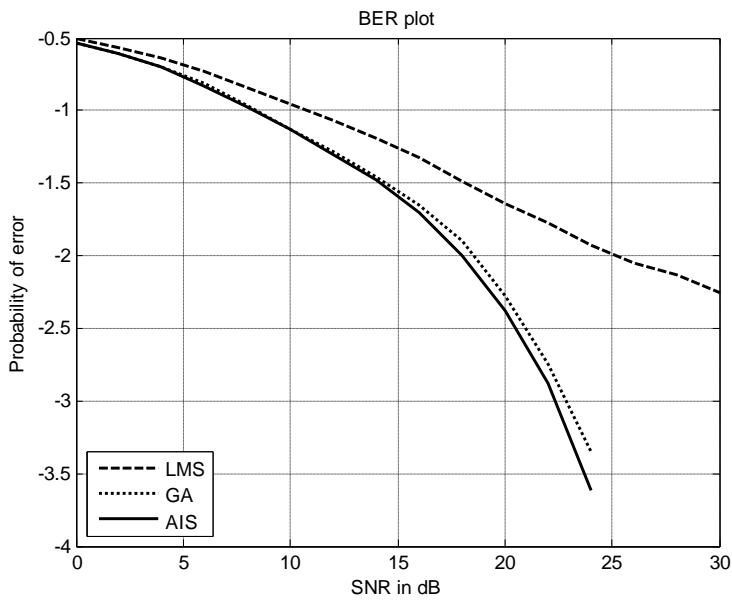


Fig.5.8 (a)

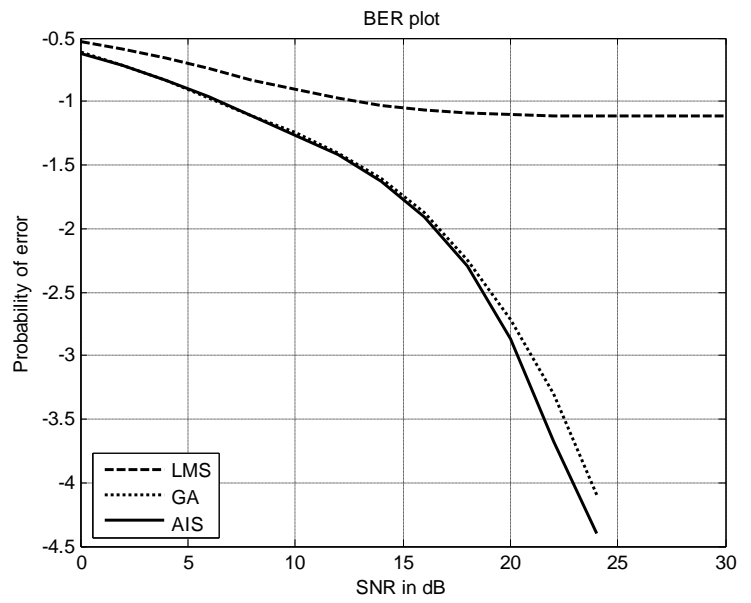


Fig.5.8 (b)

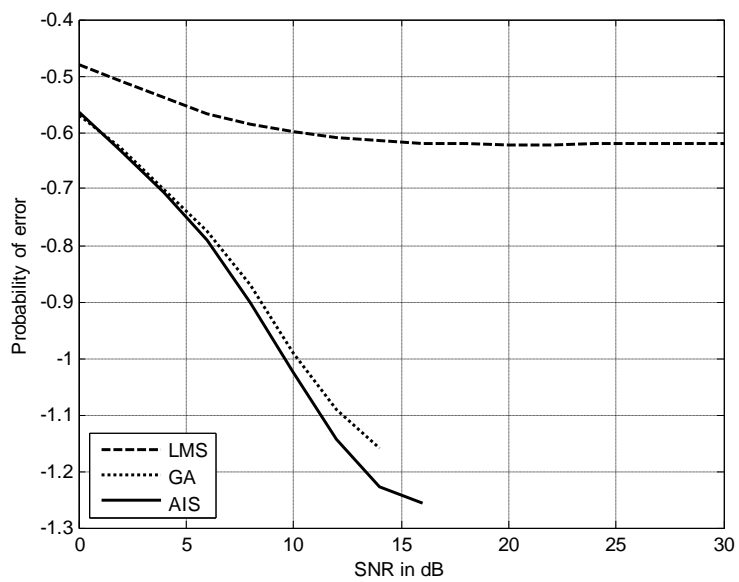


Fig.5.8 (c)

Fig.5.8 BER at 10dB SNR: (a) using  $f_1$  (b) using  $f_2$  (c) using  $f_3$  nonlinearity based channel CH2

## 5.9 Case Study on AIS based Equalizer

To performance of the proposed AIS based equalizer is analyzed by training the weights of the equalizer in four different benchmark approaches as per following

CASE 1 : The weights of the equalizer is trained at the middle of the range of NSR used for calculation of BER. Here the training is done at -15dB NSR which is at the middle of the BER range from 0-30dB.

CASE 2 : Here the training is done at an interval of 5dB SNR in the range 0-30dB for calculation of BER .

CASE 3 : The BER corresponds to taking the mean of the weight vectors trained at an interval of 5dB SNR.

CASE 4 : The BER corresponds to taking the root mean square(RMS) of the weight vectors trained at an interval of 5dB SNR.

The BER plot is done by considering the CH2 and using  $f_1$  nonlinearity

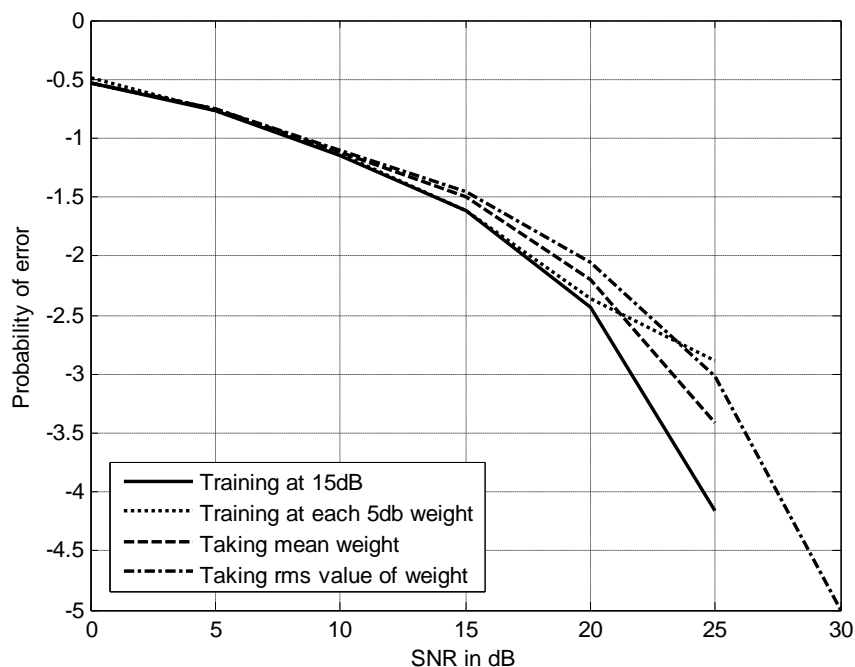


Fig.5.9 BER plot of all four case studies

---

## 5.10 Conclusion

The CPU time required for training a typical equalizer is 0.047 sec for LMS, 23.02 sec for GA and 16.265sec for AIS. Although LMS is much faster, from the BER plot it is observed that when the channel is highly noisy and nonlinear in nature it fails to equalize the transmitted output completely. Examination of BER plots Figs 5.5 -5.8 and the CPU time also reveals that AIS is a much better candidate for channel equalization as compared to its GA counter part. These types of equalizers are more suitable for offline equalization or inverse modeling problems such as nonlinearity compensations of sensors and intersymbol compensation in high density digital magnetic recording etc.

## 5.11 Summary

The contribution in this chapter is the use of AIS in adaptive channel equalization. Here a novel digital channel equalizer is proposed using CLONAL selection algorithm of AIS. Exhaustive simulation study of the proposed equalizer is carried out using benchmark examples to demonstrate its improved performance. The computed results show its superior performance compared to the LMS and GA based equalizers in terms of lower probability of error in BER plot. The CPU time required for training of AIS equalizer is also less as compared to the GA based equalizer. Thus it is concluded that the AIS is a potential learning tool for efficient equalization of complex nonlinear channels under high noise conditions.



**Chapter**  
**6**

**Development of New Evolutionary  
Hybrid Algorithms and Their  
Application to Hammerstein Model  
Identification**

---

## 6.1 Introduction

**M**ANY practical systems have inherently nonlinear characteristics due to harmonic generation, intermediation, desensitization, gain expansion and chaos. Identification of such nonlinear complex plants plays a significant role in analysis and design of control systems. The Hammerstein model is widely used because its structure effectually reflects the nonlinearity of practical dynamic systems. Several identification algorithms for the Hammerstein model has been investigated by using correlation theory [6.1], orthogonal functions [6.2], polynomials [6.3], piecewise linear model [6.4], artificial neural networks [6.5], genetic algorithm [6.6], RBF networks [6.7], PSO [6.8]-[6.9] and bacterial foraging optimization techniques [6.10]. From the literature survey it is observed that for identification of such complex plants, the recent trend of research is to employ nonlinear structures and to train their weights by evolutionary computing tools. In recent years the area of Artificial Immune System (AIS) has drawn attention of many researchers due to its broad applicability to different fields. Following the principles of AIS and PSO in this section, we propose two new hybrid intelligent algorithm called Clonal PSO (CPSO) and Immunized PSO (IPSO) which offers low complexity and better identification performance.

## 6.2 Proposed New Evolutionary Hybrid Algorithms

The Particle Swarm Optimization (PSO) is discussed in subsection 2.3.2.2. This algorithm gained a lot of attention in various optimal control system applications because of its faster convergence [6.19], reduced memory requirement, lower computational complexity and easier implementation as compared to other evolutionary algorithms. However, there are some problems associated with the basic PSO, such as premature convergence and stagnate at the local optimal solution. In [6.14] it is shown that the PSO performs well in early generations than any other evolutionary algorithm, but it degrades as the number of generations increase. So it

has a slow fine tuning ability of the solution. Several research works have been carried out to improve the performance of PSO [6.15]-[6.18].

The AIS discussed in section 3 is relatively a young field consists of many models, algorithms. By suitably combining the good features of PSO and AIS algorithms in this section we propose two new hybrid intelligent algorithms CPSO and IPSO.

### 6.2.1 Clonal PSO (CPSO)

In conventional PSO, the velocity of each particle in the next search is updated using the knowledge of its past velocity, personal and global best positions. Since the global best position after a search is the best among all personal best positions, their use in the updating the velocity has little contribution in moving to new positions. Therefore in the present investigation second term in the velocity update equation (2.56) of conventional PSO is eliminated.

Further according to clonal selection principle when an antigen or a pathogen invades the organism, a number of antibodies are produced by the immune cells. The fittest antibody undergoes cloning operation to produce number of new cells. These are used to eliminate the invading antigens. Employing this principle of AIS in PSO we propose here that each particle is led to the global best position wherefrom the next search is started. The above idea implies that during any  $k$ th search the position of the  $d$ th element of the  $i$ th particle becomes equal to the global best position i.e.  $x_{id}(k) = p_{gd}(k)$ . As a result the third term of the velocity updates equation (2.56) of conventional PSO becomes zero. Incorporating the above two ideas in the conventional PSO leads to a simplified velocity update equation by

$$V'_{id}(k+1) = w(k) * v'_{id}(k) \quad (6.1)$$

$$X'_{id}(k+1) = X'_{id}(k) + V'_{id}(k+1) \quad (6.2)$$

where  $i = 1, 2, \dots, N_1, d = 1, 2, \dots, D$ , The inertia weight  $w(k)$  is computed according to Eq. (2.60).

In the CPSO algorithm the initial position of each particle is represented as shown in Fig. 6.2(a). According to this algorithm after every stage of update all particles migrate to the global best position wherefrom each particle disperses again according to individual's magnitude and direction of velocity. This situation is depicted in Fig. 6.2(b). The same process is repeated until the position of gbest finally represents the optimal solution of the problem. Fig. 6.2(c) represents the new cloned position updated due to change in velocity.

## 6.2.2 Immunized PSO (IPSO)

The CPSO algorithm is simpler than conventional PSO and is expected to perform satisfactorily for different optimization problems. However one important observation in this new algorithm is that computation of every new position of a particle depends on two factors: that is time varying inertia weight  $w(k)$  and its initial magnitude of velocity. As a result the diversification in the solution space after each search becomes limited. Hence there is a chance that the final solution in this approach might lead to local minima. To overcome this shortcoming we propose another new algorithm called IPSO.

In this case, like the CPSO algorithm, each particle after a search occupies the global best position. Then the mutation operation is carried out on the position vector of the particles to enable random diversification of their positions. Since the position of each particle is changed unlike in CPSO, the third term remains. But the second term which contributes to change in velocity due to local best is not used. Thus the update equation becomes

$$V_{id}''(k+1) = w(k) * v_{id}''(k) + c_2'' * r_2'' * (p_{gd}(k) - x_{id}(k)) \quad (6.3)$$

$$X_{id}''(k+1) = X_{id}''(k) + V_{id}''(k+1) \quad (6.4)$$

From among the updated position of the particles the global best position is evaluated and cloned. Then the cloned cells undergo a mutation mechanism by following the hyper mutation concept of AIS. The mutation operation has fine-tuning capabilities which helps to achieve better optimal solution. Following [6.17] the single dimension mutation (SDM) operation is defined as

$$xm_{id}(k+1) = x_{T_1d}(k+1) + 0.01 * x_{T_2d}(k+1) \quad (6.5)$$

$$xm_{(i+1)d}(k+1) = x_{T_2d}(k+1) + 0.01 * x_{T_1d}(k+1) \quad (6.6)$$

where  $T_1$  and  $T_2$  represent the particles' positions to be mutated and are chosen randomly from the set of cloned positions. In order to increase the efficiency of mutation a new method Adaptive SDM (ASDM) is also proposed where the constants 0.01 is replaced by a parameter  $z$  whose values varies with the number of search. The value of  $z(k)$  at  $K$ th search is given by

$$z(k) = (z_i - z_f) \left( \frac{I-k}{k} \right) + z_f \quad (6.7)$$

where  $z_i$  and  $z_f$  are initial and final values of  $z$  and are selected within the range  $[0,1]$ . The symbol  $I$  represents the maximum number of search. The fitness values of updated position as well as the mutated position of particles are then evaluated and the overall best location is selected .In the next search the best location is again cloned and the process continues. The IPSO introduces improved search of particles in the  $D$ -dimensional space using mutation and updated velocity is shown in Fig.6.3 (b) and 6.3(c) .

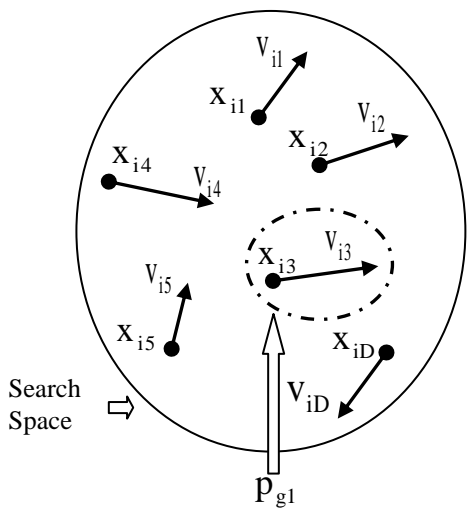


Fig. 6.1(a)

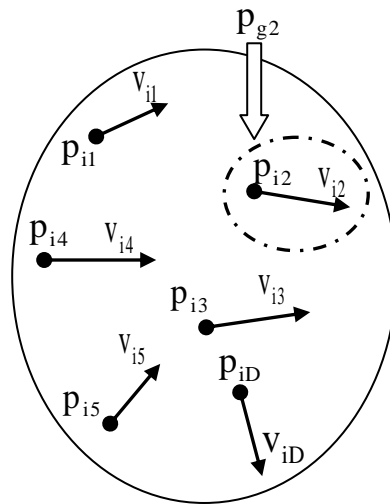


Fig. 6.1(b)

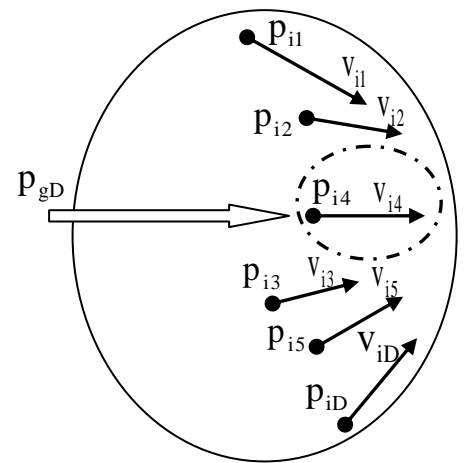


Fig. 6.1(c)

Fig.6.1 Representation of conventional PSO algorithm: (a) Initialization (b) Updated position and velocity after one stage (c) Particles movement towards solution

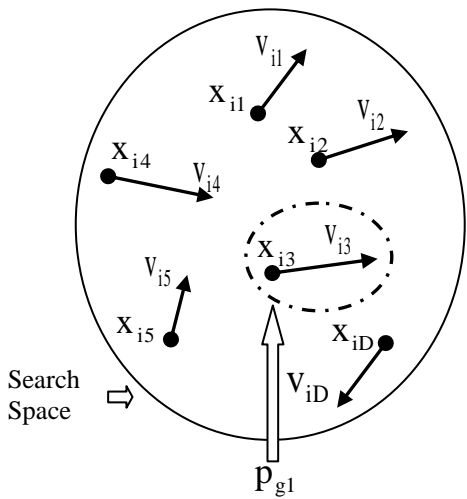


Fig. 6.2(a)

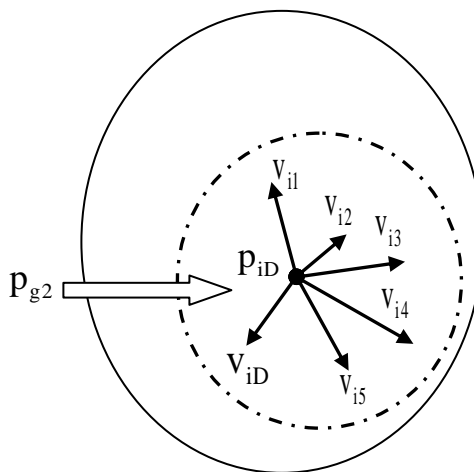


Fig. 6.2(b)

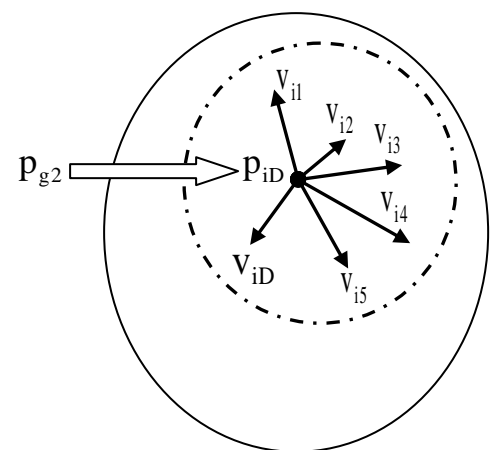


Fig. 6.2(c)

Fig.6.2 Representation of CPSO algorithm: (a) Initialization (b) Cloned position and Updated velocity after one stage (c) Updated new cloned position achieved due to new velocities

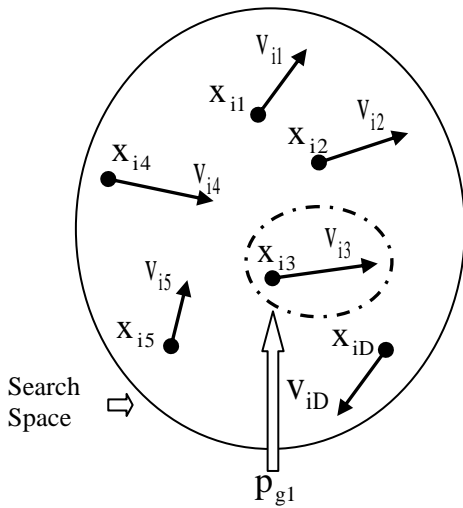


Fig. 6.3(a)

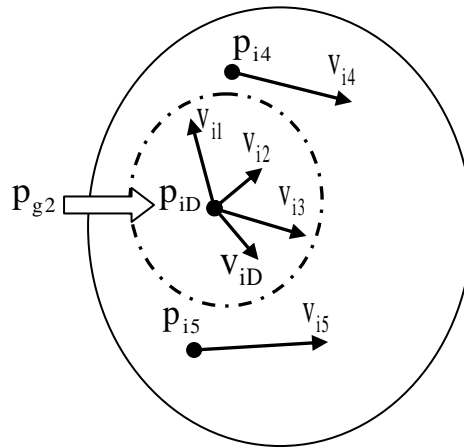


Fig. 6.3(b)

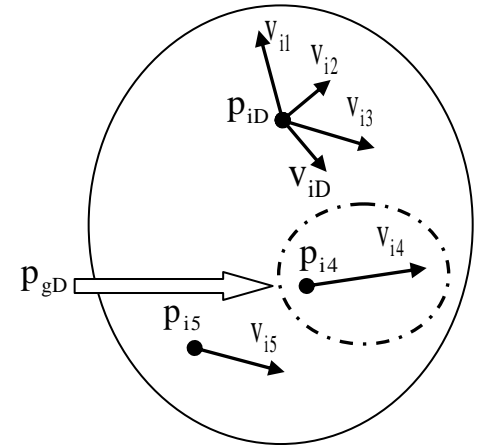


Fig. 6.3(c)

Fig.6.3 Representation of IPSO algorithm: (a) Initialization (b) Cloned position obtained through mutation and Updated velocity (c) Particles movement towards solution.

### 6.3 Identification of Hammerstein Model

The nonlinear dynamic system described by Hammerstein Model is composed of a nonlinear static block in series with a linear dynamic system block as shown in Fig.6.4. The model is described by

$$A(z^{-1})y(k) = B(z^{-1})x(k-1) + e(k) \quad (6.8)$$

$$x(k) = F(u(k)) \quad (6.9)$$

$$A(z^{-1}) = 1 + a_1z^{-1} + \dots + a_nz^{-n} \quad (6.10)$$

$$B(z^{-1}) = b_0 + b_1z^{-1} + \dots + b_rz^{-r} \quad (6.11)$$

where  $z^{-1}$  denotes an unit delay. In this model  $u(k)$ ,  $y(k)$  and  $e(k)$  represent the input, output, and noise samples at instant  $k$  respectively. The intermediate signal  $x(k)$  is not accessible for measurement. The symbols  $n$  and  $r$  are known degrees of polynomials

of  $A(z^{-1})$  and  $B(z^{-1})$  respectively. The function  $F(\cdot)$  is assumed to be nonlinear and unknown.

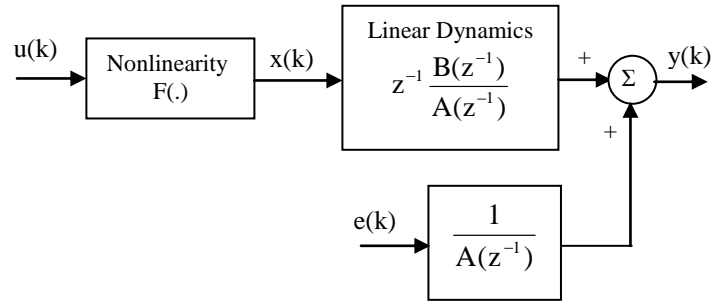


Fig.6.4. The Hammerstein Model

The objective of the identification task of the Hammerstein model is to determine the system parameters  $\{a_i\}$ ,  $\{b_j\}$  of the linear dynamic part and nonlinear static function  $F(\cdot)$  from the known input and output data  $u(k)$  and  $y(k)$ .

### 6.3.1 FLANN Structure of the Model

Here for identification purpose the nonlinear static part of the Hammerstein model is represented by a FLANN structure. The basic structure of FLANN with one input is shown in Fig.6.5. The input signal  $u(k)$  at the  $k$ th instant is functionally expanded to a number of nonlinear values to feed to an adaptive linear combiner whose weights are altered according to an iterative learning rule. The types of expansion suggested in the literature are either trigonometric, power series or square-cube expansion. For trigonometric expansion the linear matrix is given by

$$\Phi_i \{u(k)\} = \begin{cases} 1 & \text{for } i = 0 \\ u(k) & \text{for } i = 1 \\ \sin(i\pi u(k)) & \text{for } i = 2, 4, \dots, M \\ \cos(i\pi u(k)) & \text{for } i = 3, 5, \dots, M + 1 \end{cases} \quad (6.12)$$

where  $i = 1, 2, \dots, M/2$ . As a result the total expanded values including an unity bias input become  $2M+2$ . Let the corresponding weight vector be represented as



$w_i(k)$  having  $2M+2$  elements. The estimated output of the nonlinear static part as shown in Fig.6.5 is given by

$$F(u(k)) = \sum_{i=1}^{2M+2} w_i \Phi_i(u(k)) + \varepsilon(k) \quad (6.13)$$

where  $\varepsilon(k)$  is approximation error.

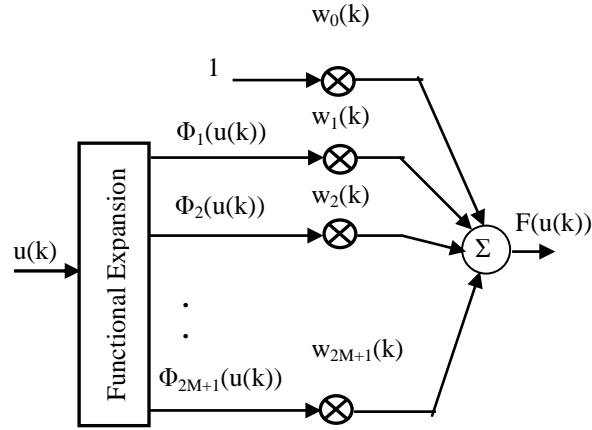


Fig.6.5. Structure of FLANN Model

Substituting (6.11) in (6.8) yields

$$A(z^{-1})y(k) = [b_0 F(u(k-1)) + b_1 F(u(k-2)) + \dots + b_r F(u(k-r-1)) + e(k)] \quad (6.14)$$

Similarly from (6.13) and (6.14) we get

$$A(z^{-1})y(k) = [b_0 \left( \sum_{i=1}^{2M+2} w_i \Phi_i(u(k-1)) + \varepsilon(k-1) \right) + \dots + b_r \left( \sum_{i=1}^{2M+2} w_i \Phi_i(u(k-r-1)) + \varepsilon(k-r-1) \right) + e(k)] \quad (6.15)$$

Rearrangement of (6.15) gives

$$A(z^{-1})y(k) = \left[ \sum_{i=0}^r b_i w_1 \Phi_1(u(k-i-1)) + \dots \right. \\ \left. + \sum_{i=0}^r b_i w_{2M+2} \Phi_{2M+2}(u(k-i-1)) + \sum_{i=0}^r b_i \varepsilon(k-i-1) \right] + e(k) \quad (6.16)$$

The identification structure of Hammerstein model corresponding to (6.16) is shown in Fig.6.6

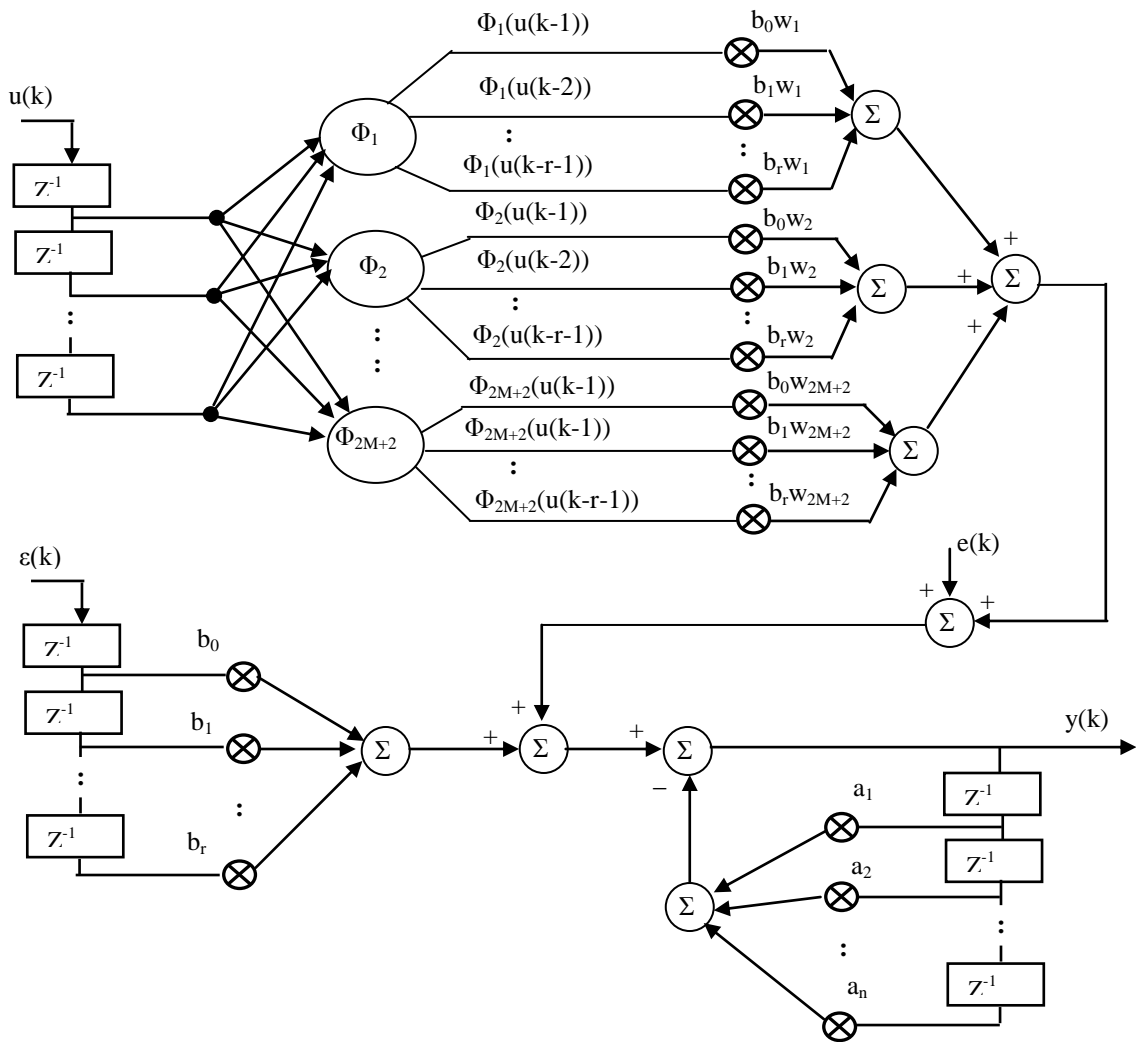


Fig 6.6 Identification structure of Hammerstein model

Here we represent

$$v(k) = e(k) + \sum_{i=0}^r b_i \varepsilon(k-i-1) \quad (6.17)$$

$$\theta = [\theta_a^T, \theta_{w_1}^T, \dots, \theta_{w_{2M+2}}^T]^T \quad (6.18)$$

where

$$\theta_a^T = [a_1, a_2, \dots, a_n]^T \quad (6.19)$$

$$\begin{aligned} \theta_{w_i} &= [\theta_{w_i}(1), \theta_{w_i}(2), \dots, \theta_{w_i}(r+1)]^T \\ &= [b_0 w_i, b_1 w_i, \dots, b_r w_i]^T \end{aligned} \quad (6.20)$$

At the  $k$ th instant  $\varphi(k)$  is given by

$$\varphi(k) = [\varphi_a^T(k), \varphi_{w_1}^T(k), \dots, \varphi_{w_{2M+2}}^T(k)]^T \quad (6.21)$$

where

$$\varphi_a(k) = [-y(k-1), -y(k-2), \dots, -y(k-n)]^T \quad (6.22)$$

$$\varphi_{w_i}(k) = [\Phi_i(u(k-1)), \dots, \Phi_i(u(k-r-1))]^T \quad (6.23)$$

Using (6.17)-(6.18) and (6.21), (6.15) can be expressed as

$$y(k) = \varphi^T(k)\theta + v(k) \quad (6.24)$$

The objective here is to estimate the system parameters defined in (6.18). If derivative based least square method is taken then the estimate of these system parameters is given by

$$\hat{\theta} = \left[ \sum_{k=N_s+1}^{N_s+N} \varphi(k)\varphi^T(k) \right]^{-1} \left[ \sum_{k=N_s+1}^{N_s+N} \varphi(k)y(k) \right] \quad (6.25)$$

where  $N$  is the number of input output data. Substituting  $\hat{w}_1 = 1$  the parameters of linear dynamic part are estimated as  $[\hat{a}_1, \dots, \hat{a}_n, \hat{b}_1, \dots, \hat{b}_r]$ . But the derivative based learning rules to update the weight of the structure at times lead to local minima and

---

thus provides incorrect estimate of the parameters. To alleviate this problem the proposed technique is in the next subsection.

## **6.3.2 Weight update of the Model**

The weights of the nonlinear FLANN structure is updated by the conventional PSO and proposed IPSO and CPSO algorithms.

### **6.3.2.1 Identification algorithm using FLANN structure and PSO based training.**

#### **Step 1. Determination of output of the Hammerstein Model:**

Uniformly distributed random ‘k’ samples are generated to be used as input during training. These are passed through the nonlinear static part then through the linear dynamic part of the Hammerstein model to produce output  $y(k)$ .

#### **Step 2. Functional expansion of input:**

The same input samples are also passed through the model consisting FLANN structure. Each input sample undergoes either trigonometric expansion as illustrated in (6.12), power series or square cube expansion.

#### **Step 3. Initialization of positions and velocities a group of particles:**

The weight vector of the Hammerstein model of Fig 6.3 is considered as a particle. Like in other evolutionary algorithm a set of particles representing a set of initial solutions is chosen. A weight vector consists of  $D$  number elements consisting of  $(M+2)$  number of elements for FLANN along with  $(n + r)$  elements for linear dynamic part. Each weight vector is represented by a particle which basically consists of a number of weights each initialized by a random number. For the  $i$  th particle, the position vector (which represents the weight vector) is given by

$$W_i = X_i = [w_{i1} w_{i2} \dots w_{id} \dots w_{iD}] \quad (6.26)$$

where  $w_{id}$  is the  $d$ th weight of the  $i$ th particle. Similarly the velocity assigned to the  $i$ th particle is expressed as

$$V_i = [v_{i1} \ v_{i2} \ \dots \ v_{id} \ \dots \ v_{iD}] \quad (6.27)$$

Initially the personal best position each  $i$ th particle has achieved is same as the initial  $i$ th weight vector  $W_i$  and is represented as

$$P_i = W_i = [w_{i1} \ w_{i2} \ \dots \ w_{id} \ \dots \ w_{iD}] \quad (6.28)$$

#### **Step 4. Calculation of output of model:**

The output of model is computed using FLANN model according to (6.24) and the weight vector defined in (6.18).

#### **Step 5. Fitness Evaluation:**

The output of the model  $\hat{y}_i(k)$  due to  $k^{\text{th}}$  sample and  $i^{\text{th}}$  particle, is compared with the output of the plant to produce error signal given by

$$e_i(k) = y_i(k) - \hat{y}_i(k) \quad (6.29)$$

For each  $i^{\text{th}}$  weight vector the mean square error (MSE) is determined and is used as the fitness function given by

$$\text{MSE}(i) = \frac{\sum_{k=1}^K e_i^2(k)}{K} \quad (6.30)$$

The identification task is then reduces to a minimization of the MSE defined in [35] using PSO and new algorithms.

#### **Step 6. Updation :**

The velocity and the position of the  $d$ th weight of each  $i$ th particle for the next search are obtained by the update rule given in Eq. (2.56)-(2.60).

---

### **Step 7. Evaluation of global best position of particle:**

The fitness values of all particles are evaluated following step 5. The best fitness value that is, the minimum MSE (MMSE) is obtained and its corresponding D weights are identified and termed as the global best. It is denoted by

$$P_g = W_g = [w_{g1} \ w_{g2} \ \dots \ w_{gd} \ \dots \ w_{gD}] \quad (6.31)$$

### **Step 8. Stopping Criteria:**

The search process described in steps 1 to 6 continues until all the particles in the swarm (the weight vectors) have attained to the global best position corresponding to a predefined MSE is obtained.

## **6.3.2.2 Identification algorithm using FLANN structure and CPSO based training.**

In CPSO steps 1 to 5 remain same as of basic PSO.

### **Step 6. Updation:**

The position and velocity of dth weight of each ith particle for the next search is obtained by the update rule given in (6.1)-(6.2). The maximum velocity and position of particles after updating is controlled by (2.58) and (2.59).

### **Step 7. Evaluation of global best position of particle:**

The global best position of particles is evaluated in similar way as described in step 7 of PSO.

### **Step 8. Cloning Operation:**

The global best position is cloned in the sense that all particles of the swarm starts their next search from this position..

### **Step 9. Stopping Criteria:**

The search process of steps 4-8 continues until all the particles in the swarm (the weight vectors) have attained the global best corresponding to a predefined MSE.

---

### **6.3.2.3 Identification algorithm using FLANN structure and IPSO based training.**

Steps 1 to 5 these steps are identical to that of CPSO.

#### **Step 6. Updation:**

The position and velocity of  $d$ th weight of each  $i$ th particle for the next search is obtained by the update rule given (6.3)-(6.4). The maximum velocity and position of particles after updating is governed by (20) and (22).

#### **Step 7. Evaluation of global best position of particle:**

The global best position of particles is evaluated in similar way following step 7 of PSO algorithm.

#### **Step 8. Cloning Operation :**

The global best particle position is cloned so that all particles occupy the same best position.

#### **Step 9. Mutation :**

Mutation process is incorporated to introduce variations in the cloned position. Probability of mutation  $P_m$  is taken to be greater than 0.5. The mutated children produced are given by

$$xm_{id}(k+1) = x_{T_1d}(k+1) + z(k) * x_{T_2d}(k+1) \quad (6.32)$$

$$xm_{(i+1)d}(k+1) = x_{T_2d}(k+1) + z(k) * x_{T_1d}(k+1) \quad (6.33)$$

#### **Step 10. Stopping Criteria:**

The search process from steps 4-9 continues until all the particles in the swarm (the weight vectors) have converged to the global best position yielding a predefined minimum MSE.

---

## 6.4 System Simulation

To demonstrate the improved identification performance of the two new algorithms simulation study using MATLAB is carried out. Four different standard Hammerstein examples have been identified using CPSO and IPSO algorithms. The accuracy of identification of the proposed models has been assessed by comparing the following.

1. True and estimated responses at the output of nonlinear static part.
2. The true and estimated coefficients of the linear dynamic part.
3. Comparison of overall output responses of estimated model and true output.
4. Comparison of sum of square errors (SSE) between true and estimated response.

The sum of square error is defined as

$$SSE(k) = \sum_{k=1}^K (y(k) - \hat{y}(k))^2 \quad (6.34)$$

where  $y(k)$  is true output and  $\hat{y}(k)$  is estimated output during testing.

### Example 1 :

The Hammerstein model described in [6.7] is given by

$$A(z^{-1})y(k) = B(z^{-1})x(k-1) + e(k) \quad (6.35)$$

$$x(k) = F(u(k)) = u(k) + 0.5u^3(k) \quad (6.36)$$

$$A(z^{-1}) = 1 + 0.8z^{-1} + 0.6z^{-2} \quad (6.37)$$

$$B(z^{-1}) = 0.4 + 0.2z^{-1} \quad (6.38)$$

The identification model of Fig.6.7 is simulated by uniformly distributed input lying between [-3.0, 3.0]. The noise  $e(k)$  is a zero mean white Gaussian distribution with standard deviation of 0.01. The number of input samples used to train the network is 300. In the model the nonlinear static part is represented by a FLANN structure.



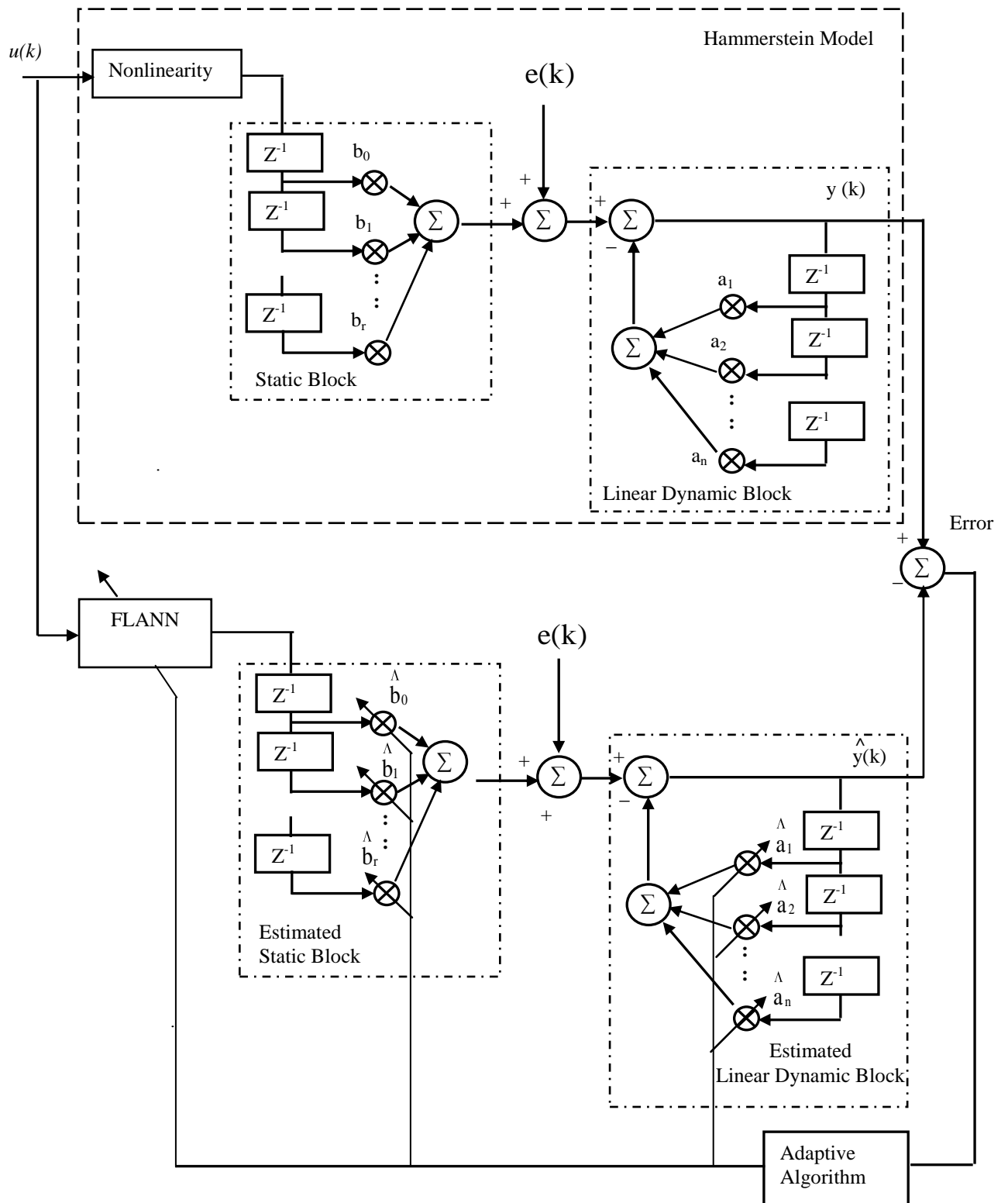


Fig 6.7 Identification structure of Hammerstein model for system simulation

Each input is passed through the FLANN structure where each is expanded to 4 terms given by

$$\Phi_i \{u(k)\} = \begin{cases} 1 & \text{for } i = 0 \\ u(k) & \text{for } i = 1 \\ u^i(k) & \text{for } i = 2,3 \end{cases} \quad (6.39)$$

The identification task is carried out using PSO, CPSO and IPSO algorithms. In all cases the initial population of particles is taken as 70. The weights of the model are trained for 40 generations. The positions of the birds are taken within range  $[-2, 2]$  and their corresponding velocities are taken in the range  $[-1.5, 1.5]$ . In case of IPSO the probability of mutation  $P_m$  is taken as 0.8. The values of  $z_i$  and  $z_f$  are set at 0.9 and 0.05 respectively. The true and estimated outputs of nonlinear static part of the given example are compared in Fig.6.8. Comparison of estimates of the system parameters of linear dynamic part are shown in Table 6.1. The CPU time required for training of model structure is presented in Table 6.2. The responses of the overall plant and model output obtained by IPSO, CPSO and PSO algorithm are shown in Fig.6.9 (a), (b) and (c) respectively. The comparative results of sum of square errors (SSE) obtained during testing is presented in Table 6.2.

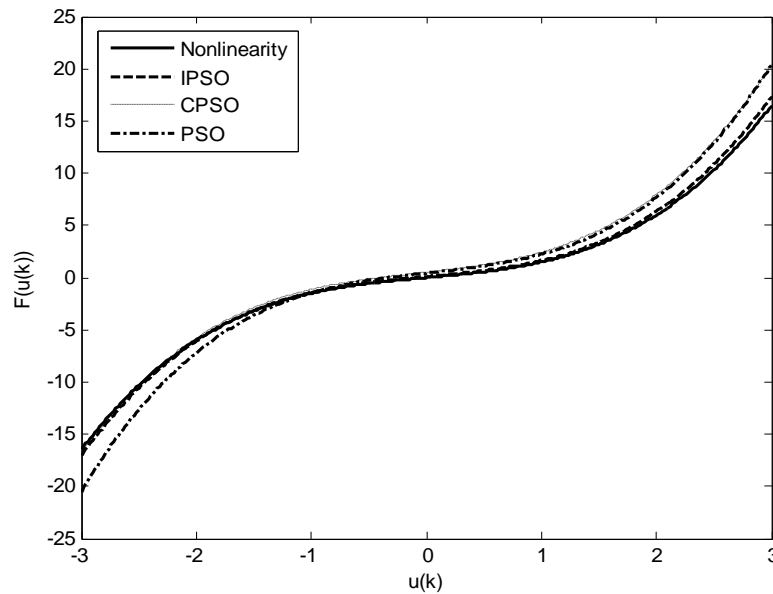


Fig 6.8 True and estimated nonlinear response of static part of the model of Example1

TABLE 6.1  
COMPARATIVE RESULTS OF ESTIMATES OF SYSTEM PARAMETERS FOR DYNAMIC PART OF  
MODEL OF EXAMPLE 1

Parameters	True Values	Estimated Values		
		IPSO	CPSO	PSO
$a_1$	0.8	0.805	0.900	0.850
$a_2$	0.6	0.590	0.606	0.650
$b_0$	0.4	0.409	0.350	0.336
$b_1$	0.2	0.210	0.240	0.240

TABLE 6.2  
COMPARATIVE RESULTS OF CPU TIME AND SSE FOR MODEL OF EXAMPLE 1 OBTAINED THROUGH  
SIMULATION STUDY

Algorithm	During training	During testing
	CPU time (In Sec.)	SSE
IPSO	28.468	0.661
CPSO	28.448	4.334
PSO	27.813	10.683

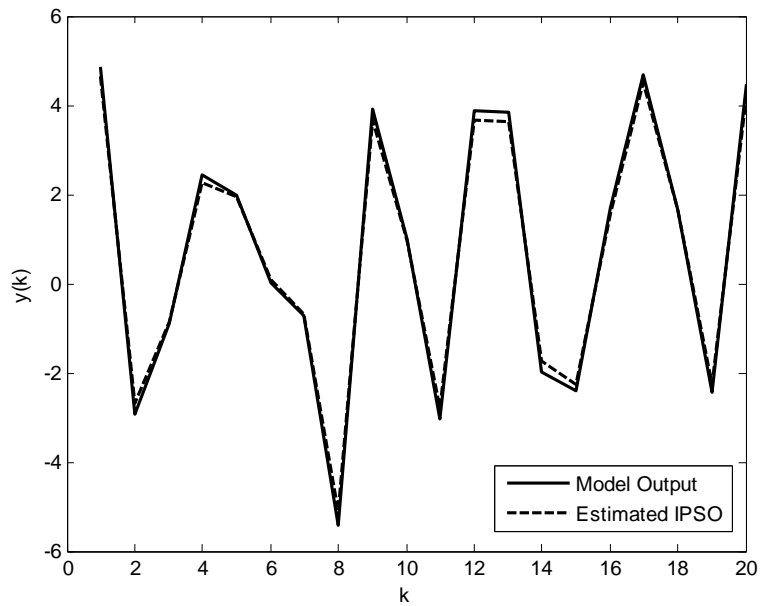


Fig 6.9 (a)

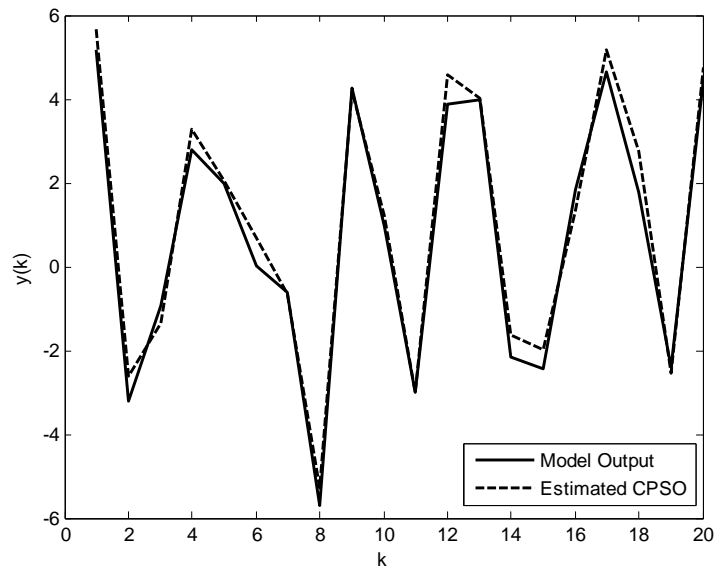


Fig 6.9 (b)

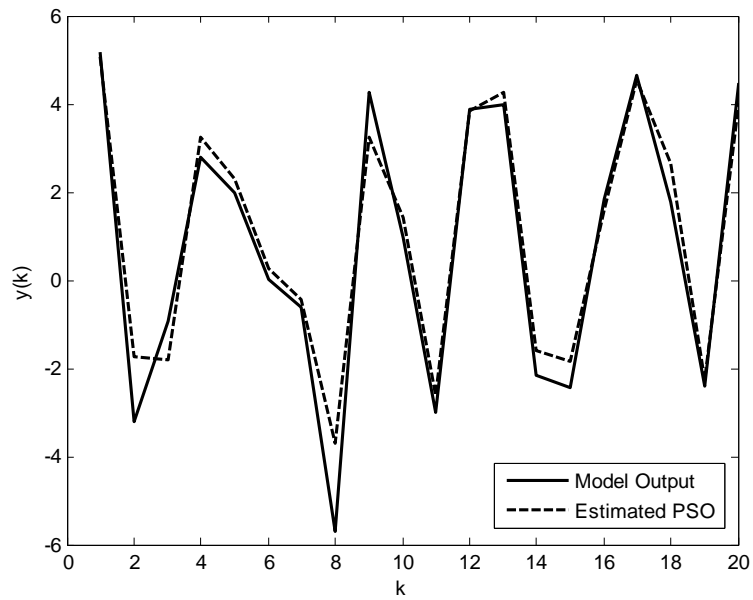


Fig 6.9 (c)

Fig.6.9 Response matching of Model and estimated output during testing of example1 using (a) IPSO (b) CPSO (c) PSO

---

**Example 2 :**

The standard model taken from [6.7] and is described by

$$A(z^{-1})y(k) = B(z^{-1})x(k-1) + e(k) \quad (6.40)$$

$$x(k) = F(u(k))$$

$$= \begin{cases} -2.0 & (-3.0 \leq u(k) < -1.8) \\ u(k)/0.6 + 1.0 & (-1.8 \leq u(k) < -0.6) \\ 0.0 & (-0.6 \leq u(k) < 0.6) \\ u(k)/0.6 - 1.0 & (0.6 \leq u(k) < 1.8) \\ 2.0 & (1.8 \leq u(k) \leq 3.0) \end{cases} \quad (6.41)$$

$$A(z^{-1}) = 1 + 0.8z^{-1} + 0.6z^{-2} \quad (6.42)$$

$$B(z^{-1}) = 0.4 + 0.2z^{-1} \quad (6.43)$$

This system has saturation and dead- zone nonlinearity. The input signal is a uniformly distributed signal lying between [-3.0, 3.0]. The number of samples of above input signal used to train the network is 100. The white Gaussian noise  $e(k)$  is zero mean with standard deviation 0.01. In the FLANN structure of the model shown in Fig.6.7, each input sample is expanded to 3 terms given by

$$\Phi_i\{u(k)\} = \begin{cases} 1 & \text{for } i = 0 \\ u(k) & \text{for } i = 1 \\ \sin(u(k)) & \text{for } i = 2 \end{cases} \quad (6.44)$$

The initial population of particles is taken as 70. The weights of the model are trained for 40 generations. The positions of the birds are taken within the range [-2, 2] and their corresponding velocities are taken with in the range [-1.5, 1.5]. In case of IPSO, the probability of mutation  $P_m$  is taken as 0.8. The values of  $z_i$  and  $z_f$  are taken values 0.9 and 0.05 respectively. The true and estimated output of nonlinear static part of the model is presented in Fig.6.10. Comparison of estimates of the system parameters of linear dynamic part are shown in Table 6.3. Table 6.4 represents the comparative result of CPU time required for training of model and square errors

(SSE) obtained during testing. The responses of the overall plant and model output obtained by IPSO, CPSO and PSO algorithm are shown in Fig.6.11 (a), (b) and (c) respectively.

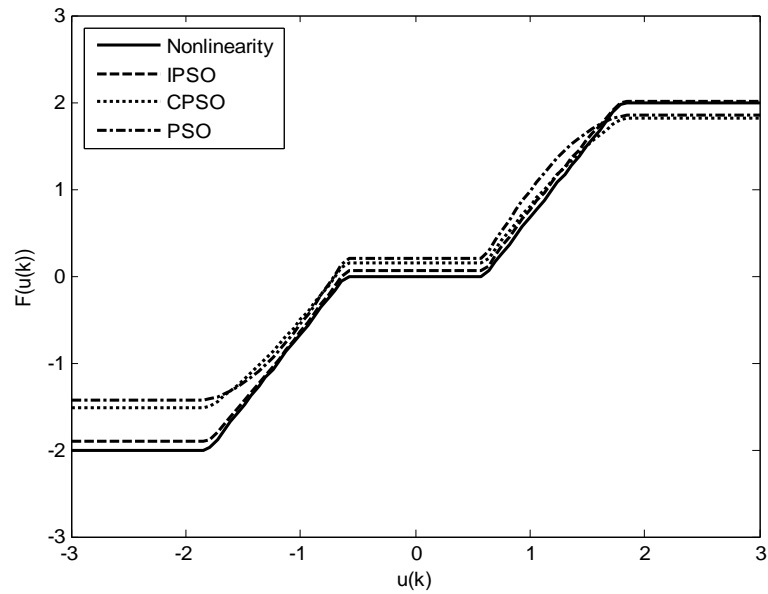


Fig 6.10 True and estimated nonlinear response of static part of the model of Example 2

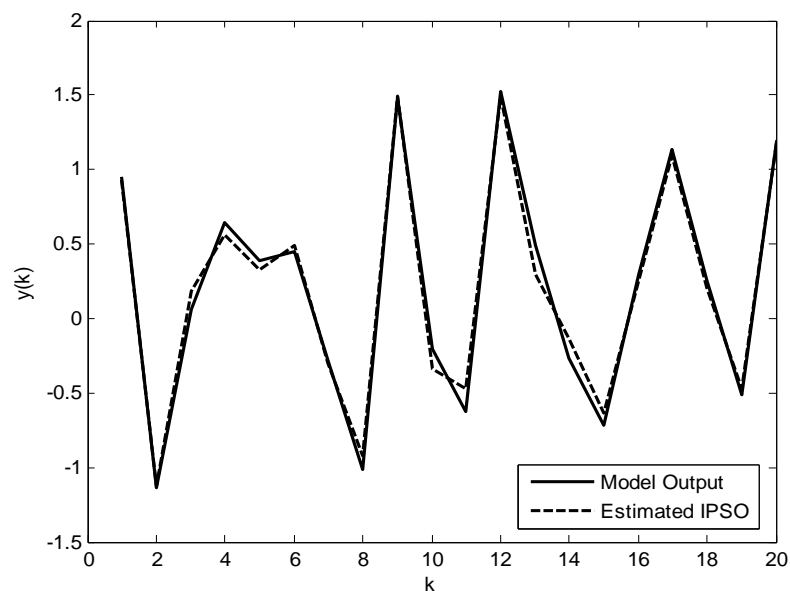


Fig 6.11 (a)

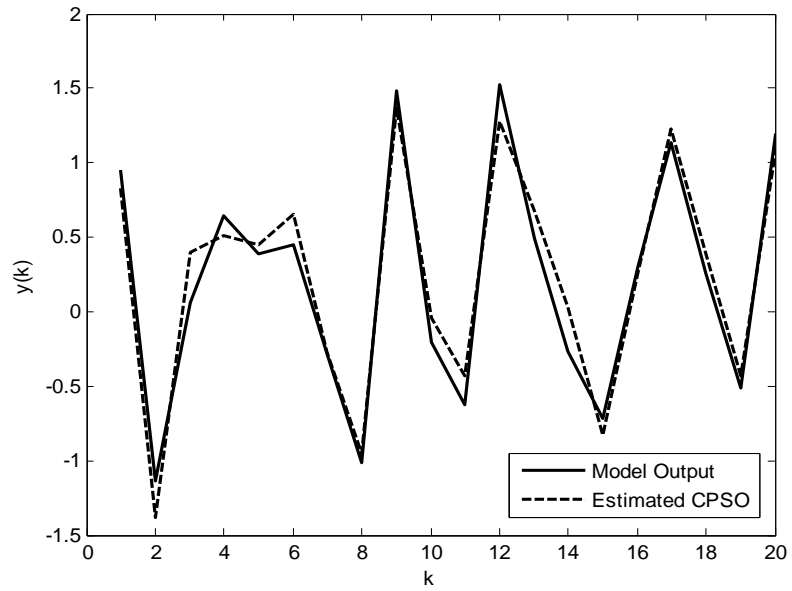


Fig 6.11 (b)

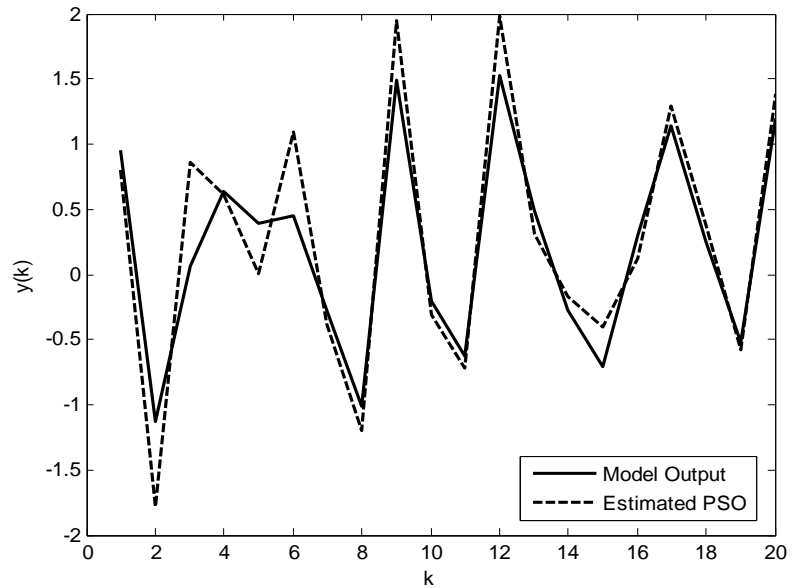


Fig 6.11 (c)

Fig.6.11 Response matching of Model and estimated output during testing of example2 using (a) IPSO (b) CPSO (c) PSO

TABLE 6.3  
COMPARATIVE RESULTS OF ESTIMATES OF SYSTEM PARAMETERS FOR DYNAMIC PART OF  
MODEL OF EXAMPLE 2

Parameters	True Values	Estimated Values		
		IPSO	CPSO	PSO
$a_1$	0.8	0.810	0.690	0.900
$a_2$	0.6	0.600	0.520	0.553
$b_0$	0.4	0.410	0.450	0.466
$b_1$	0.2	0.200	0.220	0.251

TABLE 6.4  
COMPARATIVE RESULTS OF CPU TIME AND SSE FOR MODEL OF EXAMPLE 2 OBTAINED THROUGH  
SIMULATION STUDY

Algorithm	During training	During testing
	CPU time (In Sec.)	SSE
IPSO	9.844	0.149
CPSO	9.687	0.513
PSO	9.578	2.410

**Example 3 :**

The Hammerstein model taken from [6.10] is described by

$$A(z^{-1})y(k) = B(z^{-1})x(k-1) + e(k) \quad (6.45)$$

$$\begin{aligned} x(k) &= F(u(k)) \\ &= u(k) + 0.5u^2(k) + 0.3u^3(k) + 0.1u^4(k) \end{aligned} \quad (6.46)$$

$$A(z^{-1}) = 1 + 0.9z^{-1} + 0.15z^{-2} + 0.02z^{-3} \quad (6.47)$$

$$B(z^{-1}) = 0.7 + 1.5z^{-1} \quad (6.48)$$

For identification of above system the model of Fig.6.7 is taken into consideration. The input to this model is a uniformly distributed signal lying between [-1.0, 1.0]. The number above input sample used to train the network is 300. The



white gaussian noise  $e(k)$  is zero mean with standard deviation 0.01. In FLANN each input sample is expanded to 5 terms given by

$$\Phi_i\{u(k)\} = \begin{cases} 1 & \text{for } i = 0 \\ u(k) & \text{for } i = 1 \\ \sin((i-1)*u(k)) & \text{for } i = 2,4 \\ \cos((i-1)*u(k)) & \text{for } i = 3 \end{cases} \quad (6.49)$$

The initial population of particles is taken as 90. The weights of the model are trained for 40 generations. The positions of the birds are taken within range  $[-2, 2]$  and its corresponding velocities are taken in the range  $[-1.5, 1.5]$ . In case of IPSO the probability of mutation  $P_m$  is taken as 0.8. The values of  $z_i$  and  $z_f$  used are 1 and 0.01 respectively. The true and estimated outputs of nonlinear static part of the given example are compared Fig.6.9. Comparison of estimates of the system parameters of linear dynamic part is shown in Table 6.5. The CPU time required for training of model structure is presented in Table 6.6. The responses of the overall plant and model output obtained by IPSO, CPSO and PSO algorithms are shown in Fig.6.10 (a), (b) and (c) respectively. The comparative result of sum of square errors (SSE) obtained during testing is presented in Table 6.6.

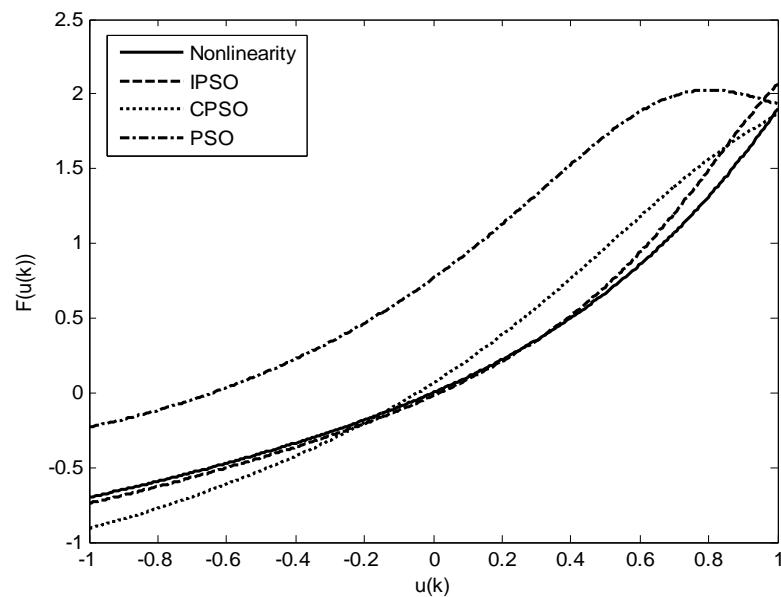


Fig 6.12 True and estimated nonlinear response of static part of the model of Example 3

TABLE 6.5  
COMPARATIVE RESULTS OF ESTIMATES OF SYSTEM PARAMETERS FOR DYNAMIC PART OF  
MODEL OF EXAMPLE 3

Parameters	True Values	Estimated Values		
		IPSO	CPSO	PSO
$a_1$	0.9	0.898	0.900	0.841
$a_2$	0.15	0.146	0.069	0.150
$a_3$	0.02	0.020	0.020	0.020
$b_0$	0.7	0.696	0.750	0.416
$b_1$	1.5	1.494	1.084	0.892

TABLE 6.6  
COMPARATIVE RESULTS OF CPU TIME AND SSE FOR MODEL OF EXAMPLE 3 OBTAINED THROUGH  
SIMULATION STUDY

Algorithm	During training	During testing
	CPU time (In Sec.)	SSE
IPSO	53.79	3.587
CPSO	51.63	6.550
PSO	50.84	10.325

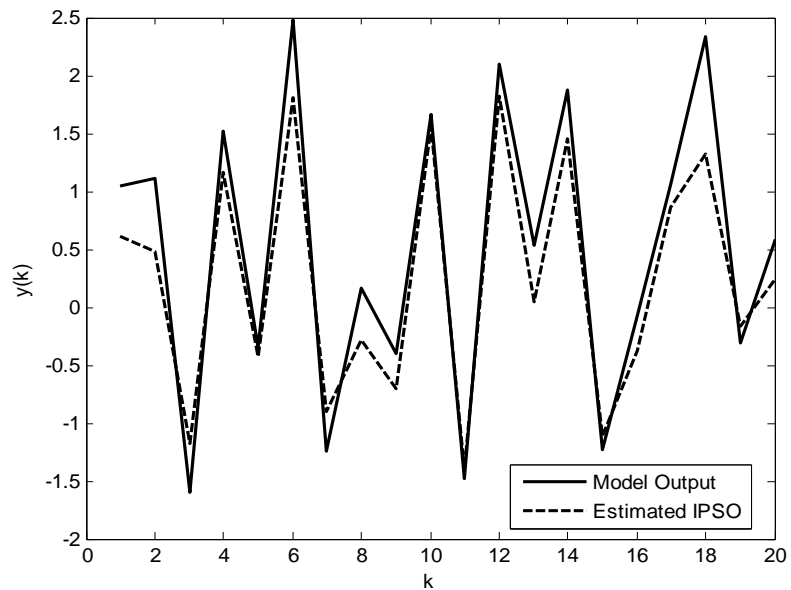


Fig 6.13 (a)

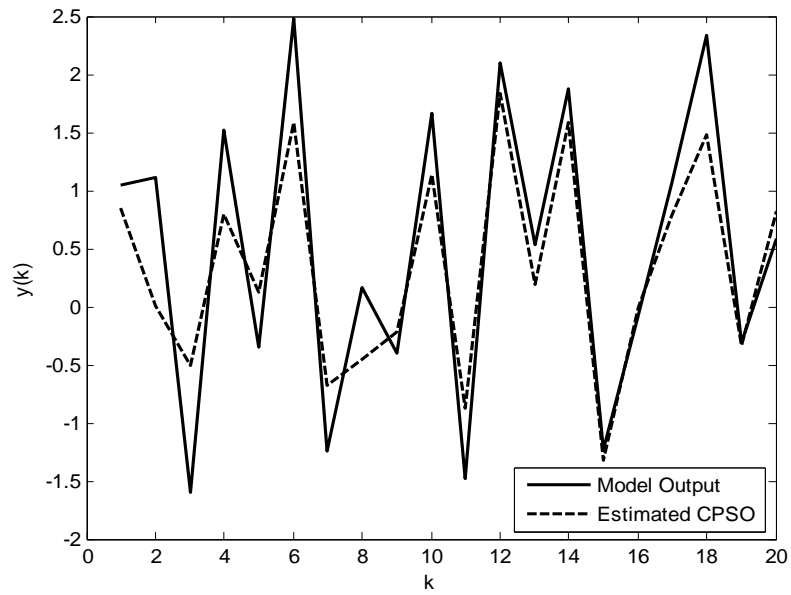


Fig 6.13 (b)

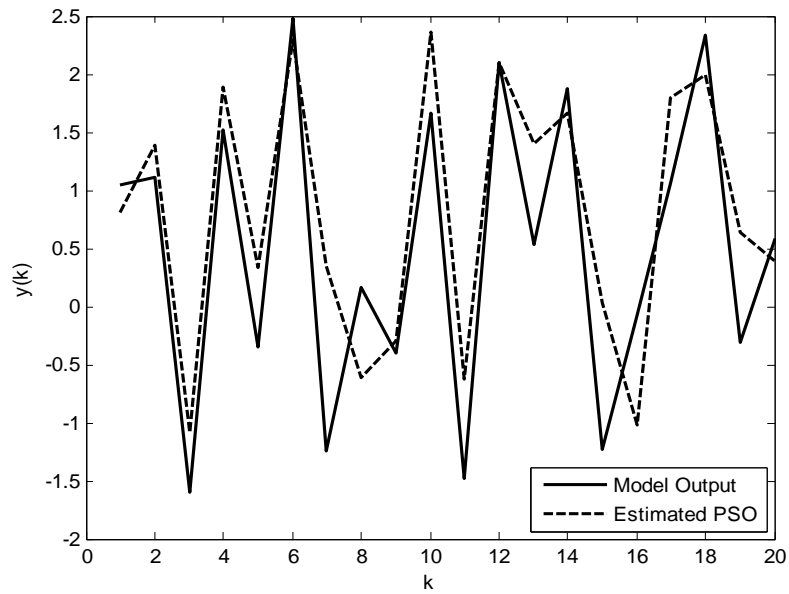


Fig 6.13 (c)

Fig.6.13 Response matching of Model and estimated output during testing of example3 using (a) IPSO (b) CPSO (c) PSO

---

#### Example 4 :

The Hammerstein model taken here is same as that of the example3, except that a different nonlinear static part given in (6.50) is used

$$x(k) = F(u(k)) = 0.5 * \sin^3(\pi u(k)) \quad (6.50)$$

For modeling of the above plant the weights of the model are trained for 40 generations. In case of IPSO the values of  $z_i$  and  $z_f$  are set to 0.9 and 0.05 respectively. The rest conditions of simulation are same as that used in example 3. The true and estimated outputs of nonlinear static part of the given example are compared in Fig.6.14. Comparison of estimates of the system parameters of linear dynamic part are shown in Table 6.7. Table 6.8 represents the comparative result of CPU time required for training of model and square errors (SSE) obtained during testing. The responses of the overall plant and model output obtained by all these algorithms are shown in Fig.6.15.

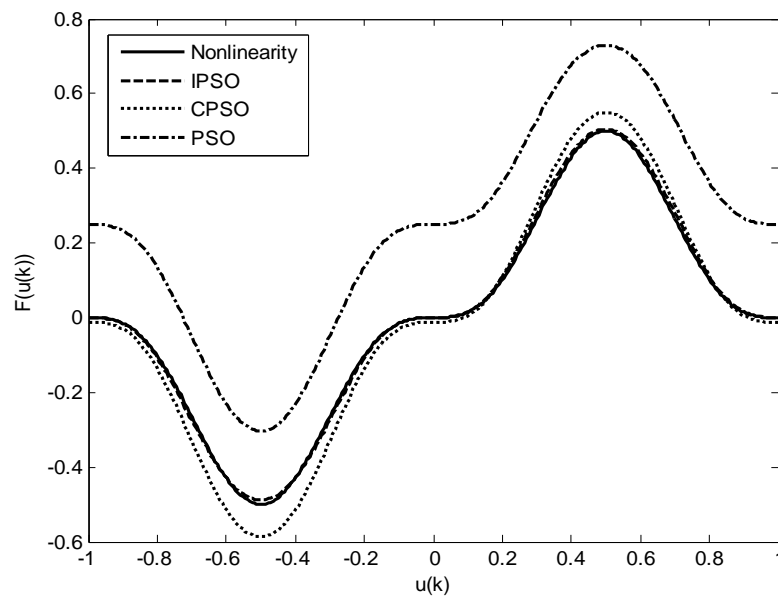


Fig 6.14 True and estimated nonlinear response of static part of the model of Example 4

TABLE 6.7  
COMPARATIVE RESULTS OF ESTIMATES OF SYSTEM PARAMETERS FOR DYNAMIC PART OF  
MODEL OF EXAMPLE 4

Parameters	True Values	Estimated Values		
		IPSO	CPSO	PSO
$a_1$	0.9	0.890	0.910	0.931
$a_2$	0.15	0.150	0.170	0.182
$a_3$	0.02	0.021	0.022	0.021
$b_0$	0.7	0.690	0.660	0.800
$b_1$	1.5	1.480	1.471	1.440

TABLE 6.8  
COMPARATIVE RESULTS OF CPU TIME AND SSE FOR MODEL OF EXAMPLE 4 OBTAINED THROUGH  
SIMULATION STUDY

Algorithm	During training	During testing
	CPU time (In Sec.)	SSE
IPSO	46.42	0.0016
CPSO	44.04	0.0066
PSO	44.00	0.1326

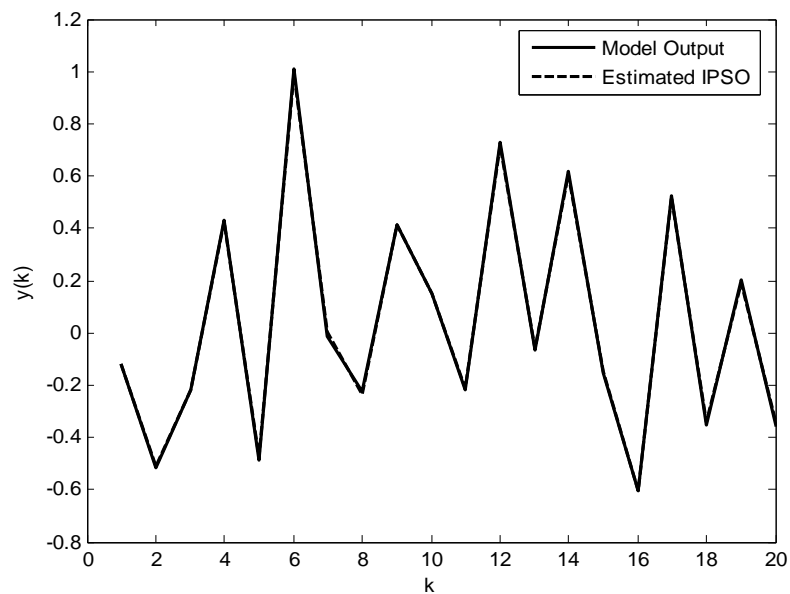


Fig 6.15 (a)

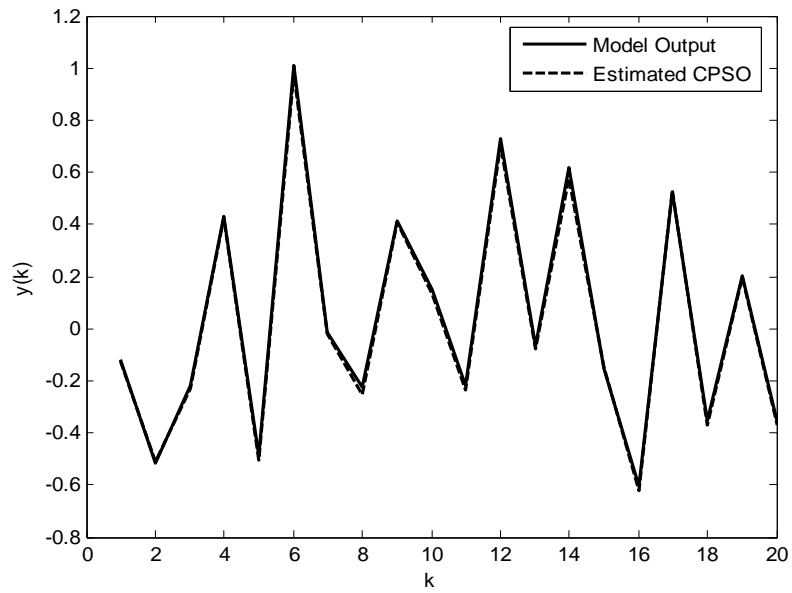


Fig 6.15 (b)

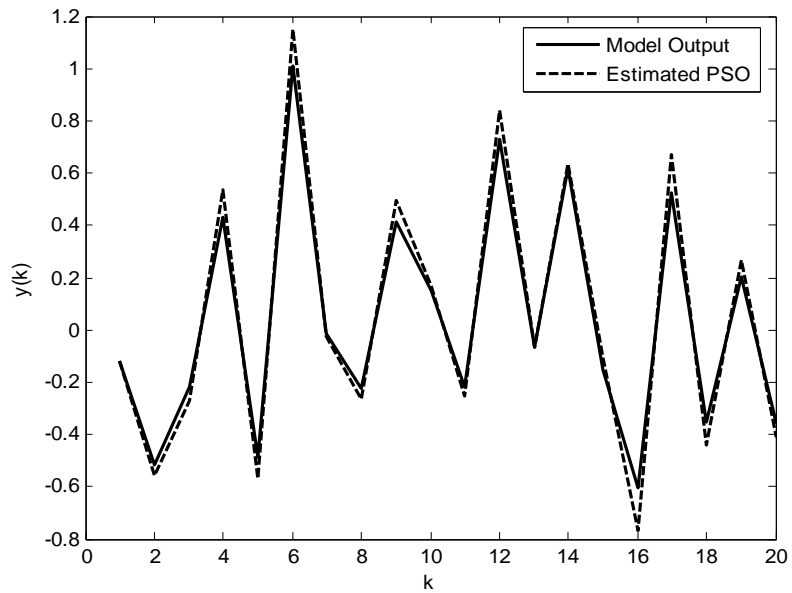


Fig 6.15 (c)

Fig.6.15 Response matching of Model and estimated output during testing of example4 using (a) IPSO (b) CPSO (c) PSO

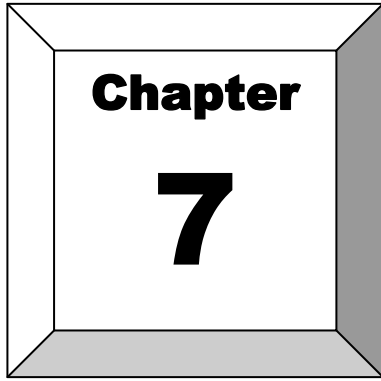
---

## 6.5 Conclusion

In this section both the proposed hybrid algorithms are relatively simple compared to the original PSO algorithm. But the simulation study reveals that the IPSO algorithm offers superior identification performance compared to the other two. Out of the two algorithms proposed, the CPSO is computationally simpler but offers identification performance nearly similar to its PSO counterpart. Under identical conditions the IPSO requires more CPU time followed by PSO and CPSO. The above observations have been arrived at by comparing the SSE, the output response and the true and estimated parameters obtained from the simulation study of four benchmark examples.

## 6.6 Summary

In this chapter we propose two new hybrid evolutionary algorithms known as Clonal PSO (CPSO) and Immunized PSO (IPSO) by suitably combining the good features of PSO and AIS algorithms. The details of these two algorithms are outlined. The performance of these new algorithms has been assessed by employing them in identification of various standard Hammerstein models. The nonlinear static part of the model to be estimated is represented by a single layer low complexity nonlinear functional link artificial neural network architecture. The weights of this structure and the dynamic part of the model are estimated by the proposed algorithms.



**Chapter**  
**7**



**CONCLUSIONS**



---

## 7.1 Conclusions

**T**HE thesis has proposed new hybrid models for identification of complex dynamics and Hammerstein plants by using functional link artificial neural network and different clonal selection principle of AIS. The FLANN is selected as it is a single layer low complex neural structure and its performance similar to that multilayer neural network. Clonal selection principle is relatively a new learning approach in the field of adaptive identification and equalization. Variants of this principle have been developed and have been applied for training the connective weights of the FLANN structure. In essence the identification and equalization models have been formulated as the squared error optimization problem of the FLANN and has been effectively solved by AIS approach. The performance of the new models has been obtained by simulation of bench mark identification and equalization examples. Further MLP structure with BP learning has been selected as identification model and its performance has been evaluated through simulation. Similarly GA based training of FLANN parameter has also been carried out and the performance of the resulting model has been obtained. The performance of all these models are compared. It is observed that the proposed models involve low complexity, consume less CPU time for training and testing and offer superior identification and equalization performance compared to corresponding conventional BP or GA based models.

---

## 7.2 Scope of future work

**T**HE present work can be extended in many directions. Firstly the same study can be extended using RBF structure as back bone and tuning its parameters such as standard deviation, centre and weights by clonal selection principle. Another extension can also be carried out by formulating the identification problem as a multiobjective problem and then solving the same by using a multiobjective GA , PSO or BFO algorithms. Similarly other AIS algorithms such as danger theory or new evolutionary computing tool such as honey bee or ant colony algorithm can also be applied to update the structure parameter of the model. ALL the above stated research methodology can also be applied to complex control, classification ,prediction and filtering problems.

## **REFERENCES**

---

## Chapter- 1

- [1.1] D. Dasgupta, “*Artificial Immune Systems and their Applications*”, Springer-Verlag, 1999.
- [1.2] L.N. de Castro, J. Timmis, “*An Introduction to Artificial Immune Systems: a New Computational Intelligence Paradigm*”, Springer-Verlag, 2002.
- [1.3] D. Dasgupta, “Advances in Artificial immune Systems,” *IEEE Computational Intelligence Magazine*, vol. 1, issue 4, pp.40 – 49, Nov. 2006.
- [1.4] L N de Charsto and J. V. Zuben , “Learning and Optimization using Clonal Selection Principle ,” *IEEE Trans on Evolutionary Computation ,Special issue on Artificial Immune Systems*, vol. 6,issue 3 , pp.239-251,2002.
- [1.5] K.S.Narendra and K. Parthasarathy, “Identification and Control of dynamical systems using neural networks,” *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27 , Mar. 1990.
- [1.6] J. C. Patra, R. N. Pal, B. N. Chatterji and G. Panda , “Identification of nonlinear dynamic systems using Functional Link Artificial Neural Networks,” *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, issue 2, pp. 254–262, Apr. 1999.
- [1.7] S. Haykin, “*Neural Networks: A comprehensive foundation*” 2<sup>nd</sup> Edition, Pearson Education Asia, 2002.
- [1.8] D. E. Goldberg, “Genetic algorithms in search, optimization and machine learning”, Addison-Wesley,1989.
- [1.9] J.Kennedy ,R. Eberhart and Y. Shi, “*Swarm intelligence*”, San Francisco: Morgan Kaufmann Publishers, 2001.
- [1.10] Marco Dorigo and Thomas Stutzle, “*Ant Colony Optimization*”, MIT Press, 2004.
- [1.11] K. M. Passino, “Biomimicry of Bacterial Foraging for distributed optimization and control”, *IEEE control system magazine*, vol 22, issue 3, pp. 52-67, June 2002.
- [1.12] W. Lin and P.X. Liu, “Hammerstein model identification based on bacterial foraging,” *IET Electronics Letters*, Vol. 42, No. 23, pp.1332-1333, 2006.
- [1.13] Proakis J. G., “*Digital Communications*”, third edition, McGraw Hill, 1995.

- 
- [1.14] J. C. Patra, R. N. Pal, R. N. Baliarsingh and G. Panda, “ Nonlinear channel equalization for QAM signal constellation using artificial neural networks,” *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, issue 2, pp. 262–271, Apr. 1999.

## Chapter- 2

- [2.1] B. Widrow and S.D. Sterns, “*Adaptive Signal Processing*”, Prentice-Hall, Inc. Engle-wood Cliffs, New Jersey, 1985.
- [2.2] S. Haykin, “*Adaptive Filter Theory*”, 4<sup>th</sup> edition, Pearson Education Asia, 2002.
- [2.3] S. Haykin, “*Neural Networks: A comprehensive foundation*” 2<sup>nd</sup> Edition, Pearson Education Asia, 2002.
- [2.4] M. T.Hagan, H.B.Demuth, M.Beale, “ *Neural Network Design*” Thomson Asia Pte. Ltd ,Singapore,2002
- [2.5] Dayhoff E.J., “*Neural Network Architecture – An Introduction*” Van Norstand Reilold, New York, 1990.
- [2.6] Bose N.K., and Liang P., “*Neural Network Fundamentals with Graphs, Algorithms, Applications*”, TMH Publishing Company Ltd, 1998.
- [2.7] Richard O. Duda, Peter E. Hart and David G. Stork, “*Pattern Classification*”, 2<sup>nd</sup> edition, John Wiley & Sons, INC.2001.
- [2.8] Y.H. Pao, “*Adaptive Pattern Recognition and Neural Networks*”, Addison Wesley, Reading, Massachusetts, 1989.
- [2.9] Y.H. Pao, S. M. Phillips and D. J. Sobajic, “Neural-net computing and intelligent control systems,” *Int. J. Conr.* , vol. 56, no.2, pp.263-289, 1992
- [2.10] Y.H. Pao, G.H. Park and D.J. Sobjic, “Learning and Generalization Characteristics of the Random Vector function”, *Neuro Computation*, vol.6, pp.163-180, 1994.
- [2.11] Fu L.M., Hsu H.H., and Principe J.C., “Incremental Back Propagation Learning Networks”, *IEEE Trans. on Neural Network*, vol.7, no.3, pp.757-762, 1996.
- [2.12] Zhao Q. and Higuchi T., “Evolutionary Learning of Nearest-Neighbor MLP”, *IEEE Trans. on Neural Network*, vol.7, no.3, pp.762-768, 1996.

- 
- [2.13] Wang G.J., and Chen C.C., "A Fast Multilayer Neural Network Training Algorithm based on the Layer-By-Layer Optimization Procedure", *IEEE Trans. on Neural Network*, vol.7, no.3, pp.768-776, 1996.
- [2.14] Carpenter G.A., and Grossberg S., "The Art of Adaptive Pattern Recognition by Self-Organizing Neural Network", *IEEE Computer Mag.* vol.21, no.3, pp.77-88, March 1988.
- [2.15] J.C. Patra., and R.N. Pal, "A Functional Link Artificial Neural Network for Adaptive Channel Equalization", *Signal Processing 43(1995)* 181-195, vol.43, no.2, May 1995.
- [2.16] Holland J. H., "Outline for a logical theory of adaptive systems," *J. ACM*, vol. 3, pp. 297 - 314, July 1962; also in A. W. Burks, Ed., *Essays on Cellular Automata*, Univ. Illinois Press, 1970, pp. 297-319.
- [2.17] Holland J.H., "*Adaptation in Natural and Artificial Systems*", University of Michigan Press, Ann Arbor, 1975.
- [2.18] D. E. Goldberg and R. Lingle, "Alleles, loci and the traveling salesman problem," *Proc. Int. Conf. Genetic Algorithms and Their Appl.*, pp. 154-159. 1985
- [2.19] D. E. Goldberg, "Genetic algorithms in search, optimization and machine learning", Addison-Wesley, 1989.
- [2.20] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in GA", *Foundations of genetic Algorithms*, I, pp. 53-69, 1991.
- [2.21] Kristinsson K. and Dumont G. A., "Genetic algorithms in system identification," *Third IEEE Int. Symp. Intelligent Contr.*, Arlington, VA, pp. 597-602, 1988.
- [2.22] G.Panda, BabitaMajhi, D.Mohanty, A.K.Sahoo, "A GA-Based Pruning Strategy and Weight Update Algorithm for Efficient Nonlinear System Identification", *Second International Conference Pattern Recognition and Machine Intelligence (PReMI 2007)*, Lecture Notes in Computer Science, pp.244-251, India,2007.
- [2.23] J.Kennedy ,R. Eberhart and Y. Shi, "*Swarm intelligence*", San Francisco: Morgan Kaufmann Publishers, 2001.
- [2.24] R.Eberhart, J.Kennedy. "A new optimizer using particle swarm theory", *The 6th Int'l Symposium on Micro Machine and Human Science*, Nagoya, Japan, pp:39-43, 1995.

- 
- [2.25] R. Eberhart, J. Kennedy, "Particle Swarm Optimization", *Proc. IEEE Int. Conf. On Neural Networks*, IEEE Press, USA, pp: 1942-1948, 1995.
- [2.26] P. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance difference", *Proceedings of the Evolutionary Programming Conference*, San Diego, USA, pp: 169-173, 1998.
- [2.27] M. Senthil Arumugam, A. Chandramohan and M.V.C. Rao, "Competitive Approaches to PSO Algorithms via New Acceleration Co-efficient variant with Mutation Operators", *Proc. of IEEE sixth Int. Conf. on Computational Intelligence and Multimedia Applications (ICCIMA'05)*, pp.225-230,2005.
- [2.28] G.Panda, D. Mohanty, B.Majhi, G.Sahoo, "Identification of nonlinear systems using particle swarm optimization technique," *IEEE Congress on Evolutionary Computation*, pp. 3253-3257, Sept. 2007.
- [2.29] V.Katari, S. Malireddi, S.K.S. Bendapudi, and G.Panda, "Adaptive Nonlinear System Identification using Comprehensive Learning PSO," *IEEE 3<sup>rd</sup> International Symposium on Comm., Cont. and Signal Processing*, pp.434-439, Mar. 2008.
- [2.30] Ajit Ku. Sahoo, "Adaptive Nonlinear System Identification and Channel Equalization using Functional Link Artificial Neural Network," M. Tech. thesis, National Institute of Technology, Rourkela, Orissa, India 2007.

### Chapter- 3

- [3.1] D. Dasgupta, "Advances in Artificial immune Systems," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp.40 – 49, Nov. 2006.
- [3.2] J.Kuby, *Immunology*, 3<sup>rd</sup> Ed., W. H. Freeman and Co., 1997.
- [3.3] C.A. Janeway, P.Travers, S. Hunt and M. Walport, *Immunobiology: the immune system in health and disease*, Garland Pub., 1997.
- [3.4] C.A. Janeway, "How the immune system recognizes invaders," *Scientific American*, vol.269, no. 3, pp. 72-79, 1993.

- 
- [3.5] S. A. Hofmeyr, "An interpretative introduction to the immune system," *Design principle for the immune system and other distributed autonomous systems*, I. Cohen and L. Segel, Eds. New York: Oxford University Press, 2000.
- [3.6] H. Baumann and J. Gauldie, "The acute response," *Immunology Today*, vol.15, no. 74, 1994.
- [3.7] I. Roitt, "Specific acquired immunology," *Essential immunology*, 9<sup>th</sup> Ed. Blackwell Science, pp.22-39, 1997.
- [3.8] D.Dasgupta, *Artificial Immune Systems and their Applications*, Springer-Verlag, 1999.
- [3.9] L.N. de Castro, J. Timmis, *An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm*, Springer-Verlag, 2002.
- [3.10] A. Coutinho, "The self non-self discrimination and the nature and acquisition of the antibody repertoire," *Annals of Immunology*, vol. 131, 1980.
- [3.11] J. Kappler, N. Roehm and P. Marrack, " T Cell Tolerance by Clonal Elimination in the Thymus," *Cell*, no. 49, pp.273-280, 1987.
- [3.12] N. K. Jerne, "Clonal selection in a lymphocyte network," *Cellular selection and regulation in the immune response*, Raven Press, pp.39-48, 1974.
- [3.13] S. Forrest and S. A. Hofmeyr, "Immunology as information processing," in *Design Principles for the Immune System and Other Distributed Autonomous Systems*, Santa Fe Institute Studies in the Sciences of Complexity, L. A. Segel and I. Cohen, Eds. Oxford, U.K. Oxford Univ. Press, pp. 361–387, 2000.
- [3.14] N. K. Jerne, "Towards a network theory of immune system," *Annals of Immunology*, vol. 125, 1974.
- [3.15] F. Varela, A. Coutinho, B. Dupire and N. Vaz, " Cognitive networks: immune and neural and other wise," *Theoretical Immunology : Part Two*, SFI Studies in the science of complexity, pp.359-371, 1988.
- [3.16] S. Forrest, A.S. Perelson, L. Allen and R. Cherukuri, "Self-Nonself Discrimination in a Computer," In *Proceedings of the 1994 IEEE Symposium on*



---

*Research in Security and Privacy*, Los Alamitos, CA: IEEE Computer Society Press, 1994.

- [3.17] S. Forrest, S. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in *Proc. IEEE Symp. Comput. Security Privacy*, pp. 120–128, 1996.
- [3.18] A. Somayaji, S. Hofmeyr and S. Forrest, "Principle of a computer immune system," Proceeding of the Second New Security Paradigms Workshop, pp.75-82 1997.
- [3.19] S.Hofmeyr, S. Forrest and A. Somayaji, "Intusion Detection Using Sequences of System Calls", *Journal of Computer Security*, vol. 6,pp.151-180,1998.
- [3.20] F. Esponda, S. Forrest and P. Helman , "A formal framework for positive and negative detection," *IEEE Trans. Syst., Man Cybernet.*, nol.34,pp.357–373, 2004.
- [3.21] U. Aickelin, J. Greensmith, and J. Twycross, "Immune System Approaches to Intrusion Detection - a Review", *In the Proceeding of the Third Int. Conf. on Artificial Immune Systems ,ICARIS-04*, pp.316-329, 2004.
- [3.22] D. Dasgupta, K. KrishnaKumar, D. Wong, and M. Berry, "Negative selection algorithm for aircraft fault detection," in *Proc. 3rd Int. Conf. on Artificial Immune System, ICARIS 2004*, pp. 13–16. 2004.
- [3.23] J. Kim, and P. Bentley, "Evaluating negative selection in an artificial immune systems for network intrusion detection," *Proc. Genetic and Evolutionary Comp. Conf.*, pp. 1330–1337,2001.
- [3.24] J. Kim, and P. Bentley, " Towards an artificial immune systems for network intrusion detection: an investigation of dynamic clonal selection," *Proc. IEEE Congress on Evolutionary Computation 2002*, pp. 1015– 1020, 2002.
- [3.25] L N de Charsto and J. V. Zuben , "Learning and Optimization using Clonal Selection Principle ," *IEEE Trans on Evolutionary Computation ,Special issue on Artificial Immune Systems*, vol. 6,issue 3 , pp.239-251,2002.
- [3.26] L N de Charsto and J. Timmis , "An Artificial Immune Network for Multimodal Function Optimization ,"*IEEE Congress on Evolutionary Computation (CEC'02)*, vol. 1,pp.699-674,May ,Hawaii,2002.
- [3.27] J. Timmis, "Artificial immune systems: a novel data analysis technique inspired by the immune network theory," Ph. D thesis, University of Wales, 2000.

- 
- [3.28] J. Timmis and M.J. Neal, "A resource limited artificial immune system for data analysis," in Research and development in intelligent systems XVII, proceedings of ES2000, Cambridge, UK, pp.19-32, 2000.
- [3.29] P. Matzinger, "Tolerance, Danger and the Extended Family", *Annual Review in Immunology*, vol.12, pp.991-1045, 1994.
- [3.30] P. Matzinger , "The Danger Model: a Renewed Sense of Self", *Science*, vol.296, pp.301-304, 2002.
- [3.31] J. Kubi, *Kubi Immunology*, 5th Ed., Freeman, San Francisco, 2002.
- [3.32] A. Tarakanov et. al., *Immunocomputing: Principles and Applications*, Springer, Berlin, 2003.
- [3.33] Andrew B. Watkins, "Exploiting immunological metaphors in the development of serial, parallel, and distributed learning algorithms," Ph. D thesis, University of Kent, 2005.
- [3.34] Fabio Gonzalez, "A study of artificial immune systems to anomaly detection," Ph.D thesis, University of Merphis, 2003.
- [3.35] D. W. Bradley and A. M. Tyrrell , "Immuotronics – Hardware fault tolerance inspired by the immune system ,"in *Proceedings of the 3<sup>rd</sup> International Conference on Evolvable Systems(ICES2000)* ,vol. 1801, Springer-Verlag ,Inc., pp.11-20, 2000.
- [3.36] D. W. Bradley and A. M. Tyrrell , "Immuotronics – Novel Finite-State-Machine Architectures With Built-in Self-Test Using Self–Nonself Differentiation ," *IEEE Trans. on Evolutionary Computation* ,vol. 6,issue 3 , pp.227-238,June ,2002.
- [3.37] P. K. Harmer, P. D. Williams, G. H. Gunsch, and G. B. Lamont, "An artificial immune system architecture for computer security applications," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 3, pp. 252–280, Jun. 2002.
- [3.38] P. S. Andrews and J. Timmis, "Inspiration for the next generation of artificial immune systems," in *Proc. 4th Int. Conf. Artif. Immune Syst., ICARIS -05*, Aug. 2005, vol. 3627, Lecture Notes in Computer Science, pp. 126–138,2005.
- [3.39] N.Cruz-Cortes, Daniel Trejo-P'erez, and C. A. Coello Coello, "Handling constraints in global optimization using an artificial immune system," *In the*

- 
- Proceeding of the Fourth Int. Conf. on Artificial Immune Systems ,ICARIS-05*, pp.234-247, 2005.
- [3.40] N. Cruz-Cortes, F. Rodriguez-Henriquez and C. A. Coello Coello, “An Artificial Immune System Heuristic for Generating Short Addition Chains,” *IEEE Trans. on Evolutionary Computation*, vol. 12, no 1 , pp.1-24,2008.
- [3.41] B. Sirisanyalak and O.Sornil , “An artificial immunity based spam detection system ” *IEEE Congress on Evolutionary Computation (CEC’07)*, pp.3392-3398, 2007.
- [3.42] Zhuhong Zhang, Tu Xin, “Immune Algorithm with Adaptive Sampling in Noisy Environments and Its Application to Stochastic optimization Problems,” *IEEE Computational Intelligence Magazine*, pp. 29-40, 2007.
- [3.43] Martin Drozda, Sven Schaust, Helena Szczerbicka, “AIS for Misbehavior Detection in Wireless Sensor Networks: Performance and Design Principles.” *IEEE Congress on Evolutionary Computation (CEC’07)*, pp.3719-3726, 2007.
- [3.44] Hong-Wei Ge , L Sun, Y.C. Liang and F. Qian , “An effective PSO and AIS-Based hybrid Intelligent Algorithm for Job Shop Scheduling,” *IEEE Trans. Syst., Man, Cybern. A*, vol. 38, issue 2, pp. 358–368, 2008.
- [3.45] Maoguo Gong, Licheng Jiao, Ling Wang, Haifeng Du, “An Artificial Immune System Algorithm for CDMA Multi-user Detection over Multi-Path Channels,” *Proceedings of conference on Genetic and evolutionary computation*, pp. 2105 - 2111 , 2005.
- [3.46] Steve Cayzer and U. Aickelin, “A Recommender System based on Idiotypic Artificial Immune Networks,” *Journal of Mathematical Modelling and Algorithms*, vol.4,no. 2, pp 181-198, 2005.
- [3.47] Q Chen, U Aickelin, “Movie recommendation systems using an artificial immune system,” *Proceedings of ACDM -04*,Bristol, UK, 2004
- [3.48] Z. Ji and D. Dasgupta, “Applicability issues of the real-valued negative selection algorithms,” *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 111–118, July 8–12, 2006.

---

## Chapter- 4

- [4.1] B. Widrow and S.D. Sterns, “*Adaptive Signal Processing*”, Prentice-Hall, Inc. Engle-wood Cliffs, New Jersey, 1985.
- [4.2] S. Haykin, “*Adaptive Filter Theory*”, 4<sup>th</sup> edition, Pearson Education Asia, 2002.
- [4.3] K.S.Narendra and K. Parthasarathy, “Identification and Control of dynamical systems using neural networks,” *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27 , Mar. 1990.
- [4.4] S. Jagannathan, F. L. Lewis, “Identification of a class of nonlinear dynamical systems using multilayered neural networks,” *IEEE Int. Symp. on Intelligent Control*, Columbus, Ohio, USA, pp. 345-351, 1994.
- [4.5] J. C. Patra, R. N. Pal, B. N. Chatterji and G. Panda , “Identification of nonlinear dynamic systems using Functional Link Artificial Neural Networks,” *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, no 2, pp. 254–262, Apr. 1999.
- [4.6] J. C. Patra, A. C. Kot, “Nonlinear dynamic system identification using Chebyshev Functional Link Artificial Neural Networks,” *IEEE Trans. Syst., Man, Cybern. B*, vol. 32, no 4, pp. 505–510, Aug. 2002.
- [4.7] S. Haykin, “*Neural Networks: A comprehensive foundation*” 2<sup>nd</sup> Edition, Pearson Education Asia, 2002.
- [4.8] Dayhoff E.J., “*Neural Network Architecture – An Introduction*” Van Nostrand Reilold, New York, 1990.
- [4.9] M. T.Hagan, H.B.Demuth, M.Beale, “ *Neural Network Design*” Thomson Asia Pte. Ltd ,Singapore, 2002
- [4.10] Richard O. Duda, Peter E. Hart and David G. Stork, “*Pattern Classification*”, 2<sup>nd</sup> edition, John Wiley & Sons, INC.2001.
- [4.11] Y.H. Pao, “*Adaptive Pattern Recognition and Neural Networks*”, Addison Wesley, Reading, Massachusetts, 1989.
- [4.12] Y.H. Pao, S. M. Phillips and D. J. Sobajic, “Neural-net computing and intelligent control systems,” *Int. J. Contr.* , vol. 56, no.2, pp.263-289, 1992

- 
- [4.13] Y.H. Pao, G.H. Park and D.J. Sobjic, "Learning and Generalization Characteristics of the Random Vector function", *Neuro Computation*, vol.6, pp.163-180, 1994.
- [4.14] Wang G.J., and Chen C.C., "A Fast Multilayer Neural Network Training Algorithm based on the Layer-By-Layer Optimization Procedure", *IEEE Trans. on Neural Network*, vol.7, no.3, pp.768-776, 1996.
- [4.15] Holland J. H., "Outline for a logical theory of adaptive systems," *J. ACM*, vol. 3, pp. 297 - 314, July 1962; also in A. W. Burks, Ed., *Essays on Cellular Automata*, Univ. Illinois Press, 1970, pp. 297-319.
- [4.16] Holland J.H., "*Adaptation in Natural and Artificial Systems*", University of Michigan Press, Ann Arbor, 1975.
- [4.17] J. Wang and Y. Chen, "A Fully Automated Recurrent Neural Network for unknown Dynamic System Identification and control," *IEEE Trans. On circuits and systems-I*, Vol.. 53, no.6, pp. 1363-1372, 2006.
- [4.18] G.Panda, BabitaMajhi, D.Mohanty, A.K.Sahoo, "A GA-Based Pruning Strategy and Weight Update Algorithm for Efficient Nonlinear System Identification", *Second International Conference Pattern Recognition and Machine Intelligence (PReMI 2007)*, Lecture Notes in Computer Science, pp.244-251, India,2007.
- [4.19] G.Panda, D. Mohanty, B.Majhi, G.Sahoo, "Identification of nonlinear systems using particle swarm optimization technique," *IEEE Congress on Evolutionary Computation*, pp. 3253-3257, Sept. 2007.
- [4.20] V.Katari, S. Malireddi, S.K.S. Bendapudi, and G.Panda, "Adaptive Nonlinear System Identification using Comprehensive Learning PSO," *IEEE 3<sup>rd</sup> International Symposium on Comm., Cont. and Signal Processing*, pp.434-439, Mar. 2008.
- [4.21] S. Purwar, I.N. Kar, A.N. Jha , " Online system identification of complex systems using Chebyshev neural networks," *Applied Soft Computing*, *ELSEVIER*, no 7,pp.364-372,2007.
- [4.22] G.Panda, B.Majhi, "Bacterial foraging based identification of nonlinear dynamic system." *IEEE Congress on Evolutionary Computation*, pp. 1636-1641, 2007.

- 
- [4.23] L N de Charsto and J. V. Zuben , “Learning and Optimization using Clonal Selection Principle ,” *IEEE Trans on Evolutionary Computation ,Special issue on Artificial Immune Systems*, vol. 6,issue 3 , pp.239-251,2002.
- [4.24] L N de Charsto and J. Timmis , “An Artificial Immune Network for Multimodal Function Optimization ,”*IEEE Congress on Evolutionary Computation (CEC’02)*, vol. 1,pp.699-674,May ,Hawaii,2002.
- [4.25] L.N. de Castro, J. Timmis, *An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm*, Springer-Verlag, 2002.
- [4.26] D.Dasgupta, *Artificial Immune Systems and their Applications*, Springer-Verlag, 1999.
- [4.27] Zhuhong Zhang, Tu Xin, “Immune Algorithm with Adaptive Sampling in Noisy Environments and Its Application to Stochastic optimization Problems,” *IEEE Computational Intelligence Magazine*, pp. 29-40, 2007.
- [4.28] D.G.Manolakis, V.K. Ingle, S. M. Kagon, “*Statistical and Adaptive Signal Processing*”, McGraw Hill, 2005.
- [4.29] E.S. Gopi, “Algorithm Collection for Digital Signal Processing Application”, Springer, 2007.
- [4.30] J.V. Canedy, “*Model based Signal Processing*,” IEEE Press, Eds. J Wiley and Sons, 2006.

## Chapter- 5

- [5.1] R.W.Lucky, “Automatic equalization for digital communications,” *Bell Syst. Tech. Journal*, no. 44, pp. 547-588, 1965.
- [5.2] S. Chen, G. J. Gibson , C.F.N.Cowan and P.M.Grant., “Adaptive equalization of finite nonlinear channels using multilayer perceptrons,” *EURASIP Signal Processing Jnl.*, vol.20, pp.107-119, 1990.
- [5.3] S. Chen , G. J. Gibson , C.F.N.Cowan., “Adaptive channel equalization using a polynomial perceptron structure,” *Proc. IEE, Pt.I*, vol.137, pp.257-264, 1990.
- [5.4] M. Meyer and G. Pfeiffer, “Multilayer perceptron based decision feedback equalizers for channels with intersymbol interference,” *Proc. IEE, Pt- I*, vol. 140, no. 6, pp 420-424, 1992.

- 
- [5.5] S. Arcens , J. C. Sueiro and A.R.F.Vidal, "Pao networks for data transmission equalization," *Proc. Int. Jt. Conf. on Neural Networks*, Baltimore, Maryland, pp. 11.963-11.968, 1992.
- [5.6] W.S. Gan, J. J. Saraghan, and T. S. Durrani, "New functional-link based equalizer," *Electronics Letters*, vol.28, no. 17, pp. 1643-1645, 1992.
- [5.7] S. K. Nair and Jaekyun Moon, "A theoretical study of linear and nonlinear equalization in nonlinear magnetic storage channels", *IEEE Trans. on neural networks*, vol. 8, no. 5, pp. 1106-1118, 1997.
- [5.8] H. Sun, G. Mathew and B. Farhang-Boroujeny, "Detection techniques for high density magnetic recording", *IEEE Trans. on magnetics*, vol. 41, no. 3, pp. 1193-1199, March 2005
- [5.9] F. Preparata, "Holographic dispersal and recovery of Information", *IEEE Trans. Inform. Theory*, vol. 35, no. 5, pp. 112 -1124 , 1989.
- [5.10] G. J. Gibson, S. Siu and C. F. N. Cowan, "The application of nonlinear structures to the reconstruction of binary signals", *IEEE Trans. signal processing*, vol. 39, no. 8, pp. 1877-1884, Aug. 1991.
- [5.11] P. G. Voulgaris and C. N. Hadjicostics, "Optimal processing strategies for perfect reconstruction of binary signals under power-constrained transmission", *Proc. IEEE conference on decision and control*, Atlantis, Bahamas, vol. 4, pp. 4040-4045, 2004.
- [5.12] R. Touri, P. G. Voulgaris and C. N. Hadjicostis, "Time varying power limited preprocessing for perfect reconstruction of binary signals", *Proc. of the 2006 American control conference*, Minneapdis, USA, pp. 5722-5727, 2006.
- [5.13] J.C. Patra and R. N. Pal, "A functional link artificial neural network for adaptive channel equalization", *EURASIP Signal Processing Jnl.*, vol. 43, no. 2, pp.662-670, 1995.
- [5.14] J. C. Patra, R. N. Pal, R. Baliarsingh and G. Panda, "Nonlinear channel equalization for QAM signal constellation using Artificial Neural Network," *IEEE Trans. On Systems, Man and Cybernetics – Part B*, vol. 29, No. 2, pp.262-272, 1999.

- 
- [5.15] J. C. Patra, Wei Beng Poh, N. S. Chaudhari and Amitabha Das, “Nonlinear channel equalization with QAM signal using Chebyshev artificial neural network”, Proc. of International joint conference on neural networks, Montreal, Canada, pp. 3214-3219, August 2005.
- [5.16] J.G. Proakis , “*Digital Communications*,” third edition, McGraw Hill, 1995.
- [5.17] S. Haykin, “*Adaptive Filter Theory*”, 4<sup>th</sup> edition, Pearson Education Asia, 2002.
- [5.18] B. Widrow and S.D. Sterns, “*Adaptive Signal Processing*”, Prentice-Hall, Inc. Engle-wood Cliffs, New Jersey, 1985.
- [5.19] D.G.Manolakis, V.K. Ingle, S. M. Kagon, “*Statistical and Adaptive Signal Processing*”, McGraw Hill, 2005.
- [5.20] S. M. Kuo, B.H. Lee, W Tian, “*Real-Time Digital Signal Processing*”, 2<sup>nd</sup> Edn., J Wiley and Sons, 2006.
- [5.21] L N de Charsto and J. V. Zuben , “Learning and Optimization using Clonal Selection Principle ,” *IEEE Trans on Evolutionary Computation ,Special issue on Artificial Immune Systems*, vol. 6,issue 3 , pp.239-251,2002.
- [5.22] L N de Charsto and J. Timmis , “An Artificial Immune Network for Multimodal Function Optimization ,”*IEEE Congress on Evolutionary Computation (CEC’02)*, vol. 1,pp.699-674,May ,Hawaii,2002.
- [5.23] L.N. de Castro, J. Timmis, *An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm*, Springer-Verlag, 2002.
- [5.24] D.Dasgupta, *Artificial Immune Systems and their Applications*, Springer-Verlag, 1999.
- [5.25] Zhuhong Zhang, Tu Xin, “Immune Algorithm with Adaptive Sampling in Noisy Environments and Its Application to Stochastic optimization Problems,” *IEEE Computational Intelligence Magazine*, pp. 29-40, 2007.
- [5.26] E.S. Gopi, “Algorithm Collection for Digital Signal Processing Application”, Springer, 2007.
- [5.27] J.G. Proakis and D.G.Manolakis , “*Digital Signal Processing*,” third edition, Prentice-Hall, 2007.



---

## Chapter- 6

- [6.1] S.A. Billings and S. Y. Fakhouri, "Identification of Systems Containing Linear Dynamic and Static Nonlinear Elements", *Automatica*, Vol. 18, No.1, pp. 15-26, 1982.
- [6.2] F. C. Kung and D. H. Shih , " Analysis and Identification of Hammerstein Model Non-linear Delay Systems Using Block-pulse Function Expansions ",*Int. J. Control*, Vol. 43, No. 1, pp. 139-147, 1986.
- [6.3] S. Adachi and H. Murakami , "Generalized Predictive Control System Design Based on Non-linear Identification by Using Hammerstein Model (in Japanese) ", *Trans. of the Institute of Systems, Control and Information Engineers*, Vol. 8, No. 3, pp. 115-121, 1995.
- [6.4] H. Al-Duwaish and M. N. Karim, "A New Method for the Identification of Hammerstein Model ", *Automatica* , Vol. 33, No. 10, pp. 1871-1875, 1997.
- [6.5] Y. Kobayashi, M. Oki and T. Okita , " Identification of Hammerstein Systems with Unknown Order by Neural Networks ", *Trans. on the IEE Japan*, Vol. 120-C, No.6, pp. 871-878, 2000.
- [6.6] H.X. Li, "Identification of Hammerstein models using genetic algorithms", *IEE Proc. Control Theory Appl.* , Vol. 146, No. 6 ,pp.499-504, 1999.
- [6.7] Tomohiro Hachino, Katsuhisa Deguchi and Hitoshi Takata, "Identification of Hammerstein Model using Radial Basis Function and Genetic Algorithms", *5th Asian Control Conference*, pp. 124-129 , 2004.
- [6.8] W. Lin, Huidi Zhang and Peter X. Liu, "A New Identification Method for Hammerstein Model Based on PSO", *IEEE International Conference on Mechatronics and automation*, pp. 2184-2188, 2006 .
- [6.9] W. Lin, C.G. Jiang and J.X. Qian, "The identification of Hammerstein model based on PSO with fuzzy adaptive inertia weight," *J. Syst. Sci. Inf.*, Vol. 3, No. 2 , pp.381-391,2005.
- [6.10] W. Lin and P.X. Liu, "Hammerstein model identification based on bacterial foraging," *Electronics Letters*, Vol. 42, No. 23, pp.1332-1333, 2006.

- 
- [6.11] W. Lin , R Liu, P.X. Liu amd Max. Q.H. Meng, “Parameter estimation using Biologically Inspired Methods,” *IEEE Int. Conf. on Robotics and Biomimetics* , pp.1339-1343,China, 2007.
- [6.12] R.Eberhart, J.Kennedy. “A new optimizer using particle swarm theory”, *The 6th Int’l Symposium on Micro Machine and Human Science*, Nagoya, Japan, pp:39-43, 1995.
- [6.13] R. Eberhart, J. Kennedy, “Particle Swarm Optimization”, *Proc. IEEE Int. Conf. On Neural Networks*, IEEE Press, USA, pp: 1942-1948, 1995.
- [6.14] P. Angeline, “Evolutionary optimization versus particle swarm optimization: philosophy and performance difference”, *In: Proceedings of the Evolutionary Programming Conference*, San Diago, USA, pp: 169-173, 1998.
- [6.15] J. Liu, W. Xu and J. Sun , “ Quantam-behaved Particle Swarm Optimization with Mutation Operator,” *IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI’ 05)*, 2005.
- [6.16] J. Liu, X. Fan and Z. Qu , “ An Improved Particle Swarm Optimization with Mutation Based on Similarity,” *IEEE International Conference on Natural Computation (ICNC 07)*, pp.824-828,2007.
- [6.17] M. Senthil Arumugam, A. Chandramohan and M.V.C. Rao, “Competitive Approaches to PSO Algorithms via New Acceleration Co-efficient variant with Mutation Operators”, *Proc. of IEEE sixth Int. Conf. on Computational Intelligence and Multimedia Applications (ICCIMA ’05)*,pp.225-230,2005.
- [6.18] V.Katari, S. Malireddi, S.K.S. Bendapudi, and G.Panda, “Adaptive Nonlinear System Identification using Comprehensive Learning PSO,” *IEEE 3rd International Symposium on Comm., Cont. and Signal Processing*, pp.434-439, Mar. 2008.
- [6.19] G.Panda, D. Mohanty, B.Majhi, G.Sahoo, “Identification of nonlinear systems using particle swarm optimization technique,” *IEEE Congress on Evolutionary Computation*, pp. 3253-3257,Sept. 2007.
- [6.20] B.Majhi, G.Panda , “Bacterial foraging based identification of nonlinear dynamic system,” *IEEE Congress on Evolutionary Computation*, pp. 1636-1641,Sept. 2007.

- 
- [6.21] G.Panda, BabitaMajhi, D.Mohanty, A.K.Sahoo, "A GA-Based Pruning Strategy and Weight Update Algorithm for Efficient Nonlinear System Identification", *Second International Conference Pattern Recognition and Machine Intelligence (PReMI 2007)*, Lecture Notes in Computer Science, pp.244-251, India,2007.
- [6.22] D. Dasgupta, "Advances in Artificial immune Systems," *IEEE Computational Intelligence Magazine*, vol. 1, issue 4, pp.40 – 49, Nov. 2006.
- [6.23] D. Dasgupta and N. Attoh-Okine, "Immunity-based systems: A survey," *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Orlando, FL, Oct. 12–15, 1997, pp. 369 – 374.
- [6.24] L N de Charsto and J. V. Zuben , "Learning and Optimization using Clonal Selection Principle ," *IEEE Trans on Evolutionary Computation ,Special issue on Artificial Immune Systems*, vol. 6,issue 3 , pp.239-251,2002.
- [6.25] L N de Charsto and J. Timmis , "An Artificial Immune Network for Multimodal Function Optimization ,"*IEEE Congress on Evolutionary Computation (CEC'02)*, vol. 1,pp.699-674,May ,Hawaii,2002.
- [6.26] Zhuhong Zhang, Tu Xin, "Immune Algorithm with Adaptive Sampleing in Noisy Environments and Its Application to Stochastic optimization Problems," *IEEE Computational Intelligence Magazine*, pp. 29-40, Nov.2007.
- [6.27] L.N. de Castro, J. Timmis *An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm*, Springer-Verlag, 2002.
- [6.28] D.Dasgupta, *Artificial Immune Systems and their Applications*, Springer-Verlag, 1999.