9-6-2018

FIR Digital Filter and Neural Network Design using Harmony Search Algorithm

ABIRA PAUL
University of Windsor

FIR Digital Filter and Neural Network design using Harmony Search Algorithm

By

Abira Paul

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for the Degree
of Master of Applied Science at the University of Windsor

Windsor, Ontario, Canada

2018

FIR Digital Filter and Neural Network design using Harmony Search Algorithm

by

Abira Paul

APPROVED BY:

_____

F. Baki

Odette School of Business

_____

H. Wu

Department of Electrical and Computer Engineering

_____

H. K. Kwan, Advisor

Department of Electrical and Computer Engineering

2 August, 2018

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Harmony Search (HS) is an emerging metaheuristic algorithm inspired by the improvisation process of jazz musicians. In the HS algorithm, each musician (= decision variable) plays (= generates) a note (= a value) for finding the best harmony (= global optimum) all together. This algorithm has been employed to cope with numerous tasks in the past decade. In this thesis, HS algorithm has been applied to design digital filters of orders 24 and 48 as well as the parameters of neural network problems. Both multiobjective and single objective optimization techniques were applied to design FIR digital filters. 2-dimensional digital filters can be used for image processing and neural networks can be used for medical image diagnosis. Digital filter design using Harmony Search Algorithm can achieve results close to Parks McClellan Algorithm which shows that the algorithm is capable of solving complex engineering problems. Harmony Search is able to optimize the parameter values of feedforward network problems and fuzzy inference neural networks. The performance of a designed neural network was tested by introducing various noise levels at the testing inputs and the output of the neural networks with noise was compared to that without noise. It was observed that, even if noise is being introduced to the testing input there was not much difference in the output. Design results were obtained within a reasonable amount of time using Harmony Search Algorithm.

DEDICATION

Dedicated to,

My beloved parents: Sanchita Paul and Ajit Paul

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. H.K. Kwan for introducing me the methodologies of designing digital filters, neural networks, and fuzzy neural networks using Harmony Search algorithm and suggesting it as the project of my thesis. His guidance and support throughout my M.App.Sc helped me to achieve successes. I would also like to thank my committee members for providing their valuable inputs. A special thanks to my fellow students Rija Raju, Jiajun Liang and Miao Zhang for discussions.

I would like to pay my special regards to the University of Windsor for providing an excellent environment for carrying out my study. Last but not the least I would extend my thanks to my beloved parents who have always supported me during my study period.

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY iii

ABSTRACT iv

DEDICATION v

ACKNOWLEDGEMENTS vi

LIST OF TABLES ix

LIST OF FIGURES x

LIST OF ABBREVIATIONS/SYMBOLS xi

Chapter 1: Introduction 1

1.1 Why Digital Filters? 1

1.2 Types of Filters 2

1.2.1 Based on frequency response 2

1.2.2 Based on impulse response 2

1.3 FIR Filters 2

1.3.1 Linear phase FIR filters 4

1.3.2 Equiripple design of linear phase FIR filters 6

1.3.3 Constrained equiripple FIR filters 9

1.3.4 Difference between equiripple design and least squared FIR filter design 9

1.3.5 Why are FIR filters preferred over IIR filters? 10

1.3.6 General phase FIR filters 10

1.4 Quantization of coefficients 12

1.4.1 Initial Coefficient Values for Populations 13

1.5 Filter Design Problems 13

1.5.1 Deterministic algorithms 16

1.5.2 Heuristic algorithms 16

1.5.3 Metaheuristic algorithms 17

1.5.4 Evolutionary algorithms 18

1.6 Neural Networks 20

1.7 Contributions 20

1.8 Motivation and Outline of Thesis 20

Chapter 2: Review of Literature 22
2.1 Survey of Harmony Search Applications 23
2.2 What is Harmony Search Algorithm? 24
2.3 Design of Harmony Search Algorithm 24
2.4 Improvements in Harmony Search Algorithm 26
2.5 Why is Harmony Search successful? 27
2.6 Introduction to Neural Networks 28
2.7 Why Neural Networks? 29
2.8 Types of Neural Networks 30
2.8.1 Feedforward neural networks 30
2.8.2 Fuzzy neural networks 30
Chapter 3: Methodology and Analysis 32
3.1 FIR Filter Design 32
3.2 Neural Network Design 33
3.2.1 Design using XOR neural network 33
3.2.2 Design using feedforward neural networks 34
3.2.3 XOR design using min-sum fuzzy inference network 37
Chapter 4: Experiments and specifications 40
4.1 Results 40
4.1.1 Linear phase order 24 FIR filter design obtained Using HS 40
4.1.2 Linear Phase Results compared with FIRPM for order 24 44
4.1.3 Linear Phase order 48 results obtained Using HS 47
4.1.4 Linear Phase Results compared with FIRPM for order 48 51
4.1.5 General FIR Results obtained Using HS for Order 24 54
4.1.6 Design using XOR neural network 57
4.1.7 Digit recognition using feedforward neural network 59
4.1.8 XOR design using min-sum fuzzy inference network 65
Chapter 5: Conclusions 67
REFERENCES 70
VITA AUCTORIS 72

LIST OF TABLES

Table 1.1 Symmetric filters 4
Table 1.2 Comparison between FIR and IIR filters 10
Table 1.3 Comparing the performance of the HSA algorithm with other algorithms using Test functions 19
Table 3.1 Training data sets: Nine training Samples for Fuzzy Exclusive XOR Problem 38
Table 4.1 Coefficients of order 24 Type I Lowpass LP-FIR filter by HS 41
Table 4.2 Coefficients of order 24 Type1 Bandpass LP-FIR filter by HS 42
Table 4.3 Coefficients of order 24 Type1 Highpass LP-FIR filter by HS 43
Table 4.4 Coefficients of order 24 Type1 Bandstop LP-FIR filter by HS 44
Table 4.5 Order 24 FIR type 1 filter design results comparison (PM: Parks McClellan; HS: Harmony Search) 46
Table 4.6 Coefficients of order 48 Type1 Lowpass LP-FIR filter by HS 48
Table 4.7 Coefficients of order 48 Type1 Bandpass LP-FIR filter by HS 50
Table 4.8 Order 48 FIR type 1 filter design results comparison (PM: Parks McClellan; HS: Harmony Search) 52
Table 4.9 Lowpass, Highpass, Bandpass, and Bandstop digital filter cutoff frequencies 52
Table 4.10 Linear Phase FIR Filter Coefficients (Order 24) 52
Table 4.11 Linear Phase FIR Filter Coefficients (Order 48) 53
Table 4.12 Coefficients of order 24 type1 Lowpass LP-GFIR filter by HS 55
Table 4.13 Coefficients of order 24 type1 Bandpass LP-GFIR filter by HS 56
Table 4.14 Order 24 General FIR Type 1 filter design results using HS 56
Table 4.15 2-input one output Neural network design parameters 57
Table 4.16 2-input one output Neural network design 57
Table 4.17 Computational results of the feedforward neural network design for ten hidden neurons (without noise) 59
Table 4.18 Comparison of results with the ideal output and the Mean Square errors for each of the four output neurons 59
Table 4.19 Computational results of the feedforward neural network design for ten hidden neurons (40% noise) 60
Table 4.20 Comparison of results with the ideal output and the Mean Square errors for each of the four output neurons 60
Table 4.21 Computational results of the feedforward neural network design for eight hidden neurons (without noise) 62
Table 4.22 Computational results of the feedforward neural network design for eight hidden neurons (without noise) 62
Table 4.23 Computational results of the feedforward neural network design for eight hidden neurons (40% noise) 63
Table 4.24 Computational results of the feedforward neural network design for eight hidden neurons (40% noise) 63
Table 4.25 Computational results of the min sum fuzzy neural network design (without noise) 65
Table 4.26 Comparison of results of the min-sum fuzzy inference network 65

LIST OF FIGURES

Figure 1.1 Types of filters based on frequency response 2
Figure 1.2 Block diagram of FIR filter 3
Figure 1.3 FIR in direct form 3
Figure 1.4 FIR filter in transposed direct form 3
Figure 1.5 Type I Linear phase FIR coefficients 5
Figure 1.6 Phase response of linear phase FIR filter 6
Figure 1.7 Lowpass filter design specifications (Kwan[18], 2016, Pg 18) 8
Figure 1.8 Design of equiripple FIR filters using FIRPM 9
Figure 3.1 Neural network design for two input XOR problem [18] 33
Figure 3.2 Training pattern pairs (Kwan[22], 1992, pg 2) 36
Figure 4.1 Order 24 linear phase lowpass FIR digital filter using HS 40
Figure 4.2 Order 24 linear phase bandpass FIR digital filter using HS 41
Figure 4.3 Order 24 linear phase highpass FIR digital filter using HS 42
Figure 4.4 Order 24 linear phase bandstop FIR digital filter using HS 43
Figure 4.5 Lowpass FIR filter comparing HS and FIRPM 44
Figure 4.6 Bandpass FIR filter comparing HS and FIRPM 45
Figure 4.7 Highpass FIR filter comparing HS and FIRPM 45
Figure 4.8 Bandstop FIR filter comparing HS and FIRPM 46
Figure 4.9 Order 48 linear phase lowpass FIR digital filter using HS 47
Figure 4.10 Order 48 linear phase bandpass FIR digital filter using HS 49
Figure 4.11 Lowpass FIR Filter comparing HS and FIRPM 51
Figure 4.12 Bandpass FIR Filter comparing HS and FIRPM 51
Figure 4.13 Order 24 general phase lowpass FIR digital filter using HS 54
Figure 4.14 Order 24 general phase bandpass FIR digital filter using HS 54
Figure 4.15 Two dimensional view of XOR neural network 58
Figure 4.16 Plot of the 1054 weight vectors for ten hidden neurons 61
Figure 4.17 Plot of the 844 weight vectors for eight hidden neurons 64
Figure 4.18 Plot of the 64 best solution vectors for eight hidden neurons 66

LIST OF ABBREVIATIONS/SYMBOLS

HS	Harmony Search
HSA	Harmony Search Algorithm
Fig	Figure
$H(w)$	Frequency response of the Filter
h_n	nth Filter coefficient
PM	Parks McClellan
Gd	Group delay of the passband
δ	Passband ripple
δ_p	Specified Passband ripple
δ_s	Specified Stopband ripple
$W_s(w_i)$	Weighing function
w_p	Passband cut off frequency
w_s	Stopband cut off frequency

Chapter 1: Introduction

1.1 Why Digital Filters?

What are Digital Filters?

In signal processing, a digital filter is a system that performs mathematical operations on sampled, discrete-time signal to reduce or enhance certain aspects of the signal as it may be necessary.

The applications of digital filters are widespread some of which includes the following:

- Communication Systems
- Image processing and enhancement
- Instrumentation
- Processing of biological signals

The digital filters are used in communication systems because of their ability to minimize error probability and producing quality signal. In image processing, the digital filters are used mainly to suppress either the high frequencies in the image, or the low frequencies i.e. enhancing or detecting the edges in the image. Filters alter the frequency spectrum of the input signal and thus are used in instrumentation. The digital filters have also been used for biomedical signal processing.

Before proceeding with the details and moving onto further depth, I will like to mention that in this thesis I have tried to show how to solve complex problems such as Digital Filters and Neural network designs in less time in an efficient manner as an alternative to other metaheuristic algorithms using a particular metaheuristic algorithm called Harmony Search Algorithm. This metaheuristic algorithm was initially used with Particle Swarm Optimization for the design of Infinite Response Filters for improved performance and also for many industrial applications discussed in the Chapter 2 in Section 2.2. However designing of FIR digital Filters and feedforward neural networks is a new concept which have not been addressed by others before.

1.2 Types of Filters

1.2.1 Based on frequency response

Based on the frequency response, the filters can be categorized into four common types as shown in Figure 1-1.

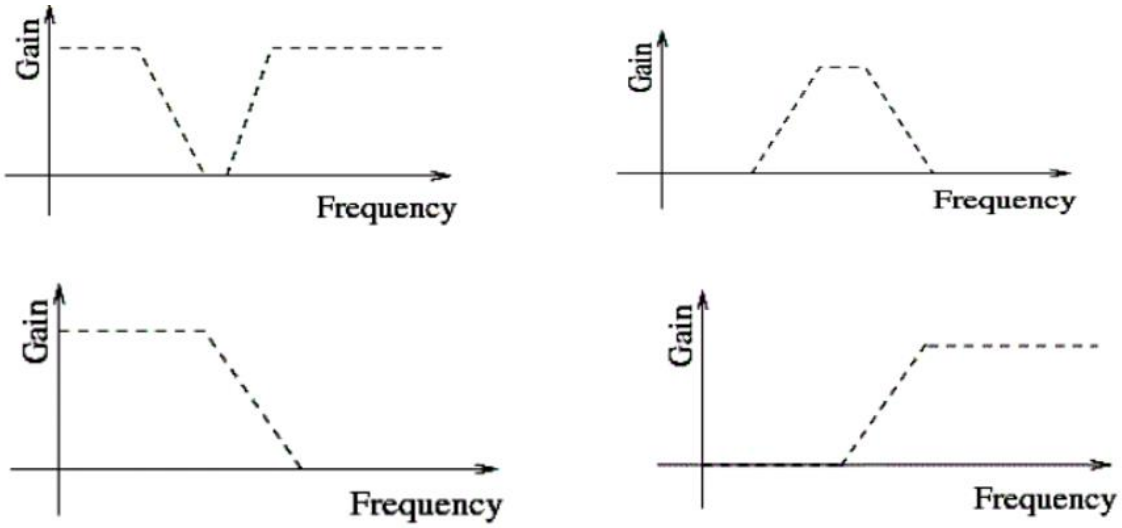

Figure 1.1 Types of filters based on frequency response

1.2.2 Based on impulse response

Based on the impulse response, there are two categories of digital filters; namely finite impulse response filters(FIR) and infinite impulse response filters(IIR). The significant difference between FIR and IIR filter is that in case of FIR filter, the output decays to 0 in a finite amount of time, whereas in case of IIR filter the output takes an infinite amount of time to decay to 0.

1.3 FIR Filters

FIR Filters are digital filters with the finite impulse response. They are also known as non-recursive digital filters as they do not have feedback.

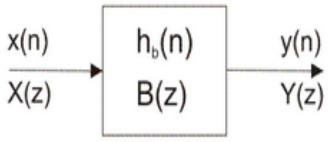

Figure 1.2 Block diagram of FIR filter

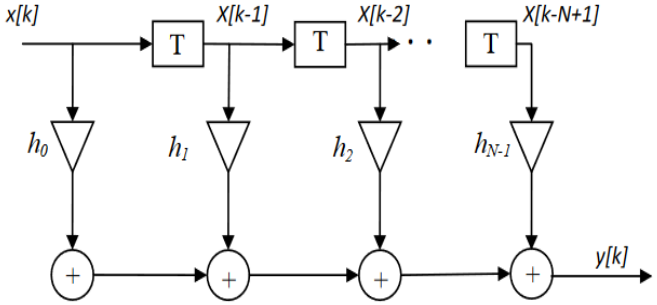

Figure 1.3 FIR in direct form

T=decay

$h_0, h_1 \dots \dots h_{n-1}$ = Filter coefficients

$x[k], x[k-1], \dots \dots x[k-N+1]$=Input and a delayed version of the input

$y[k]$ = Output of the filter

The output of the filter can be written in the following equation form:

$$y[k] = h_0 x[k] + h_1 x[k-1] + \cdots + h_{n-1} x[k-N+1] \qquad (1.1)$$

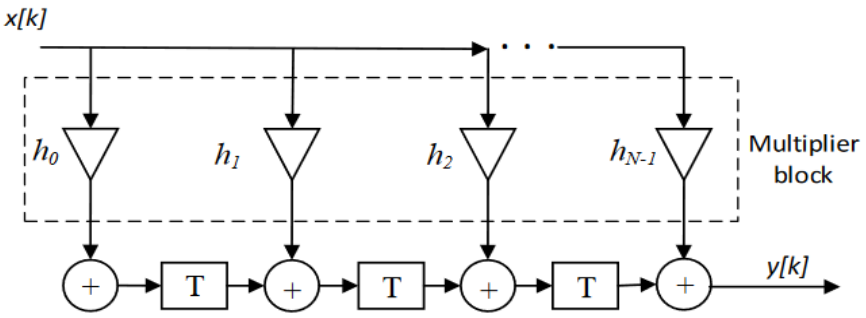

Figure 1.4 FIR filter in transposed direct form

1.3.1 Linear phase FIR filters

Depending on the order of the filters and the symmetry of the filter coefficients, the linear phase filters can be of four types [25] as shown in the following Table:

Table 1.1 Symmetric filters

	Order	**Symmetry**	$\boldsymbol{H(w)\ at\ w=0}$	$\boldsymbol{H(w) at\ w=\pi/T}$
Type I	Even	Even	Any	Any
Type II	Odd	Even	Any	0
Type III	Even	Odd	0	Any
Type IV	Odd	Odd	0	0

If $h_0, h_1 \dots\dots\dots h_{N-1}$ are the filter coefficients, where N is the length of the filter, then the following relations hold for the coefficients of the different kinds of filters.

Type I: $h_k = h_{N-k+1}, N$ is odd

Type II:$h_k = h_{N-k+1}, N$ is even

Type III:$h_k = -h_{N-k+1}, N$ is even

Type IV:$h_k = -h_{N-k+1}, N$ is odd

The amplitude response of the four types of filters can be expressed in the following equations:

Type I:

$$H(w) = h(M) + 2\sum_{n=0}^{M-1} h_n \cos\big((M-n)w\big) \tag{1.2}$$

Type II:

$$H(w) = 2\sum_{n=0}^{N/2-1} h_n \cos\big((M-n)w\big) \tag{1.3}$$

Type III

$$H(w) = 2\sum_{n=0}^{M-1} h_n \sin\big((M-n)w\big) \tag{1.4}$$

Type IV

$$H(w) = 2\sum_{n=0}^{N/2-1} h_n \cos\big((M-n)w\big) \qquad (1.5)$$

Where, $M = (N-1)/2$

In the following figure, the filter coefficient values of Type I Linear phase filter can be shown:

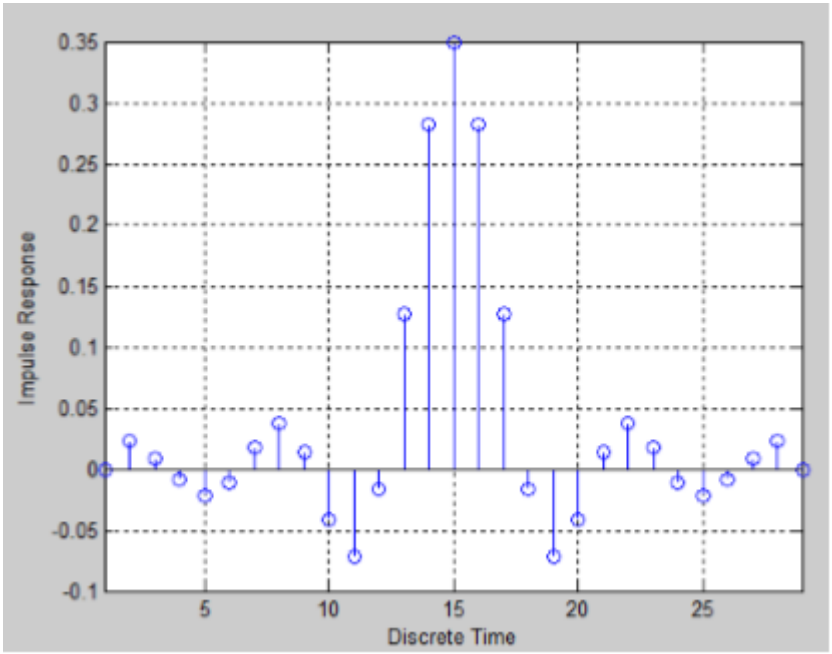

Figure 1.5 Type I Linear phase FIR coefficients

The coefficients are symmetric around the central coefficient. The phase response of the linear phase FIR filter is given below:

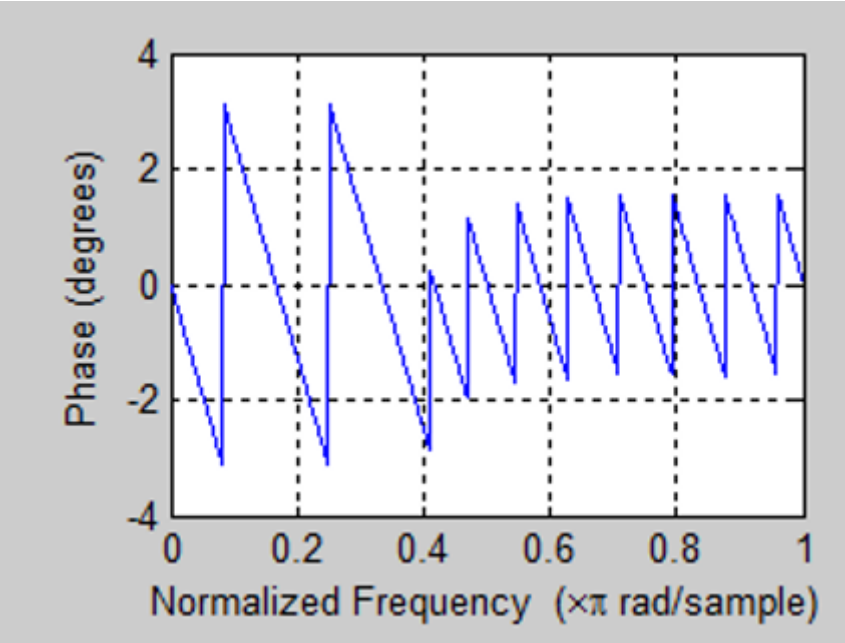

Figure 1.6 Phase response of linear phase FIR filter

We can see that the phase response of the filter varies linearly. The discontinuities are mainly due to two reasons:

1)$2\pi + \theta = \theta$ resulting the phase being confined from $-\pi \; to \; \pi$

2) The sign reversal of the frequency response.

1.3.2 Equiripple design of linear phase FIR filters

The finite length Impulse response of Filter (FIR) has the exact linear phase. FIR filters can be realized by the causal system because after time delaying any non-causal sequence can be causal.

The transfer function of the Type I Filter is given as:

$$H(c,w) = e^{-j\left(\frac{M-1}{2}\right)wT}\left\{h\left(\frac{M-1}{2}\right) + \sum_{n=0}^{\frac{M-3}{2}} 2h(n)cos\left[\left(\frac{M-1}{2} - n\right)wT\right]\right\} \qquad (1.6)$$

$$H(c,w) = e^{-j\left(\frac{M-1}{2}\right)wT}A(c,w) \qquad (1.7)$$

Where,

$$A(c, w) = \boldsymbol{c^T} cos\, w \tag{1.8}$$

And,

$$\cos w = \left[1 \quad cos(wT) \quad cos(2wT) \quad \ldots\ldots \quad cos\left(\frac{M-1}{2} wT\right)\right]^T \tag{1.9}$$

The coefficient vector c^T is optimized initially using some random values and then the objective function value is reduced at every iteration by following the structure of the algorithm which has been reduced by minimizing the value of the error at every iteration. The minimax error approximation method is used to calculate the error. It calculates the difference in error between the frequency response of the passband and stopband. An ideal filter has a magnitude of 1 and 0 in the passbands and stopband. The expression of the minimax function has been shown below:

$$e_p(c) = \left[\sum_{i=1}^{l_p} W_p(w_i) \left||A(c, w_i)| - A_d(w_i)\right|^{2p}\right]^{1/2p} \tag{1.10}$$

$$for\ W_p(w_i) \geq 0; 0 \leq w_i \leq w_p$$

$$e_s(c) = \left[\sum_{i=1}^{l_s} W_s\,(w_i) \left||A(c, w_i)| - A_d(w_i)\right|^{2p}\right]^{1/2p} \tag{1.11}$$

$$for\ W_s(w_i) \geq 0;\ w_s \leq w_i \leq \pi$$

Where $e_p(c)$ and $e_s(c)$ are the error values in the passband and stopband respectively. $A(c, w_i)$ is the magnitude response of the obtained filter and $A_d(w_i)$ is the magnitude response of the ideal filter and i is the number of samples to calculate the error. The minimax optimization problem is to search for the optimal coefficient vector $\boldsymbol{c}$ that minimizes the objective function $\boldsymbol{e(c)}$**:**

$$\min_{\boldsymbol{c}} \boldsymbol{e}(\boldsymbol{c}) \tag{1.12}$$

$W_s(w_i)$ represents the weighing function which is given by:

$$W_s(w_i) = \begin{cases} 1 & passband\ and\ stopband \\ 0 & elsewhere \end{cases} \tag{1.13}$$

The weighing scores usually differ with the error values; so, if the errors of the passband and stopband differ by a margin the weighing values will increase or decrease accordingly.

The interval $0 - w_p$ is the passband and $w_c - 1$ is the stopband of the filter. The response usually varies from $1 - \delta_p$ to $1 + \delta_p$ in the passband and in the stopband, it varies from $-\delta_s$ to δ_s . The transition region which ranges from w_p to w_s can accept any value.δ_p denotes the passband ripple whereas δ_s denotes the stopband ripple. The diagrammatic representation is shown below:

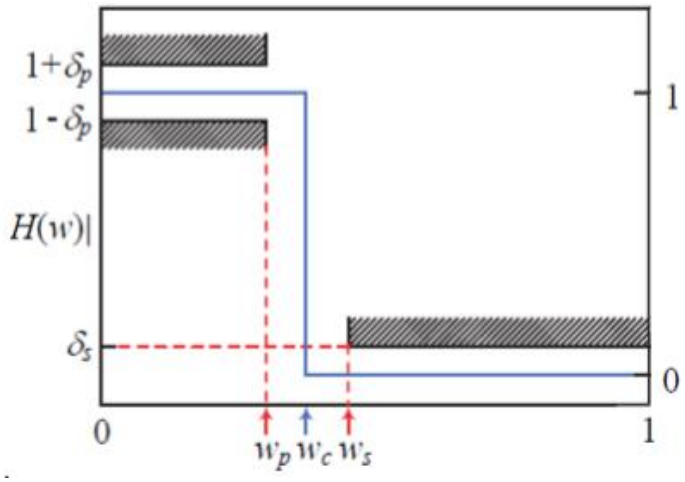

Figure 1.7 Lowpass filter design specifications [25]

The above design problem can be formulated as a linear problem shown below in the following equations:

Minimize δ

Such that: $1 - \delta \leq H(w) \leq 1 + \delta, for\ \omega \in [0, \omega_p]$ (1.14)

$$-(\delta_s \delta)/\delta_p \leq H(w) \leq (\delta_s \delta)/\delta_p\ , for\ \omega \in [w_s, 1]$$

Where $H(w)$ is the frequency response of the filter and is given by:

$$H(w) = \sum_{n=0}^{\frac{[N-1]}{2}} h(n) Trig(w, n) \quad (1.15)$$

Where $Trig$ is the trigonometric function depending on the type of the filter and whether the filter is odd or even and N is the filter length.

1.3.3 Constrained equiripple FIR filters

The design of such filters requires constrained equiripple approximation of an approximate function. The design is as shown below:

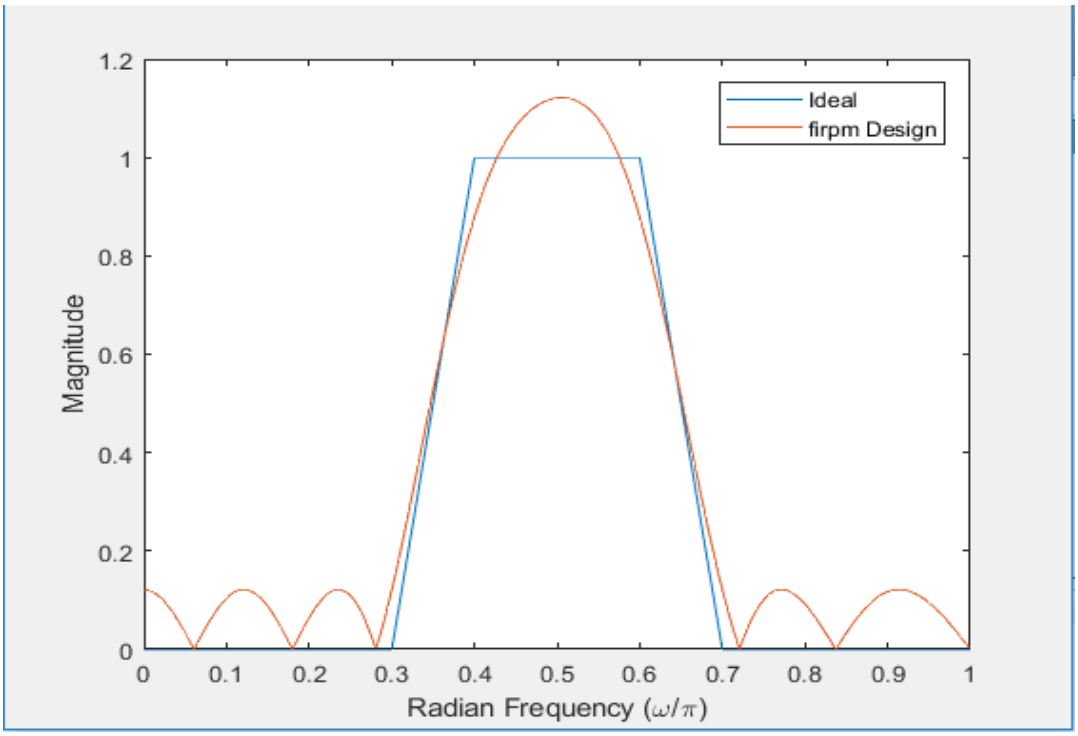

Figure 1.8 Design of equiripple FIR filters using FIRPM

As shown in the above figure, 0.4 is the cut off frequency for the design of the equiripple bandpass filter. The filter has been designed using Parks Mc Clellan Algorithm which is very efficient. This reduces the maximum error in each iteration as it is an iterative algorithm. The MATLAB function **firpm** is based on Parks McClellan method and is used to design linear phase FIR filters with a given length and specified passbands and stopbands. The syntax of the function is shown below:

b=firpm(n,f,a,w)

where n is the filter order, which is one less than the filter length. f and a define the passbands and stopbands whereas w is the weight vector of length equal to the number of bands.

1.3.4 Difference between equiripple design and least squared FIR filter design

There are two methods available broadly for the design of an efficient and optimal filter design, Equiripple filter design and the Least Squares Filter design. The Equiripple filter design has

equal ripples in the passband and stopband, so the signal distortion which happens at the edge of the passband is avoided in case of Equiripple Filter Design but it has a large transition width.

On the other hand, the Least Squares Filter design has a smaller bandwidth as compared to Equiripple filter design, hence the passband width is larger. The passband ripple exhibits a spike at the passband edge due to Gibb's phenomenon which causes signal distortion at the edge.

1.3.5 Why are FIR filters preferred over IIR filters?

Table 1.2 Comparison between FIR and IIR filters

Property	FIR filters	IIR filters
Phase or group delay	Linear phase is always possible	It is hard to design
Stability	They are always stable	They can exhibit unstable behavior and limit cycles
Order required	Large	Small
Implementation	Can have multirate or polyphase implementations	No multirate or polyphase implementations

1.3.6 General phase FIR filters

GFIR filters also known as General Finite Impulse response filters requires constant group delay in the system. These filters are asymmetric filters as they do not have same coefficients on the left and right-hand side, therefore each coefficient is different from the other. The complexity of the design makes it difficult to optimize and get a good result, so because of this only the passband delay is taken into account.

A Nth order general FIR filter [25] consists of $(N+1)$ asymmetric impulse responses and can be represented by a distinct coefficient vector c as

$$c = [c_0, c_1, c_2, c_3, \dots \dots . c_N]^T \tag{1.16}$$

The frequency response of the general FIR filter can be expressed as:

$$H(w) = \sum_{n=0}^{N} c_n z^{-n} | z = e^{jwT} = |H(w)|\, e^{j\theta(w)} \tag{1.17}$$

The group delay $\tau(w)$ of a digital filter can be computed from the derivative of phase $\theta(w)$ with respect to frequency w as:

$$\tau(w) = -\frac{\partial\theta(w)}{\partial w} \tag{1.18}$$

The objective function of the weighted least-squares magnitude response error in a passband is defined by:

$$e_{mp}(c) = \sum_{i}^{I_{mp}} W_{mp}(w_i)\left||H(c,w_i)| - H_{dp}(w_i)\right|^2 \tag{1.19}$$

$$for\ \forall w_i \in \sigma_{mp}$$

where $H_{dp}(w_i) = 1$ in the passband, I_{mp} denotes the corresponding number of discrete frequency points; $W_{mp}(w_i)$ denotes the corresponding frequency error weights, and σ_{mp} denotes the corresponding union of frequency points of interest.

$$e_{ms}(c) = \sum_{i}^{I_{ms}} W_{ms}(w_i)\left||H(c,w_i)| - H_{ds}(w_i)\right|^2 \tag{1.20}$$

$$for\ \forall w_i \in \sigma_{ms}$$

where $H_{ds}(w_i) = 0$ in the stopband, I_{ms} denotes the corresponding number of discrete frequency points; $W_{ms}(w_i)$ denotes the corresponding frequency error weights, and σ_{ms} denotes the corresponding union of frequency points of interest.

Similarly, the objective function of the weighted least square errors group delay response error in the passband is defined by:

$$e_g(c) = \sum_{i=0}^{I_g} W_g(W_i)|\tau(c,w_i) - \tau_d(w_i)|^2 \tag{1.21}$$

$$for\ \forall w_i \in \sigma_g$$

where $\tau_d(w)$ denotes the desired group delay in the passband; I_g denotes the number of discrete frequency points, $W_g(w_i)$ denotes the corresponding frequency error weights and σ_g denotes the corresponding union of frequency points of interest.

The design optimization problem for a general FIR digital filter is to search for an optimal coefficient vector c that simultaneously minimizes its magnitude and group delay errors. For the case of a lowpass digital filter consisting of a passband and a stopband, a joint objective function defined by [24] is adopted such that:

$$e(c) = \left[max\left(e_{mp}(\boldsymbol{c}), e_{ms}(\boldsymbol{c})\right) + \alpha\, e_{gp}(\boldsymbol{c})\right]^{\frac{1}{2}} \qquad (1.22)$$

The parameter α denotes a group delay error weighting factor. In this paper, the filter design problem is formulated as a joint objective function with constraints such that

$$\begin{aligned} &\min_{c} e(c) \\ &e_{mpp}(c) \leq \delta_{mp} \quad for\ \forall w_i \in \sigma_{mp} \\ &e_{msp}(c) \leq \delta_{ms} \quad for\ \forall w_i \in \sigma_{ms} \\ &e_{gp}(j) \leq \delta_g \quad for\ \forall w_i \in \sigma_g \end{aligned} \qquad (1.23)$$

The parameters $e_{mpp}(c), e_{msp}(c)\ and\ e_{gp}$ denote respectively passband peak magnitude error, stopband peak magnitude error, and passband peak group delay error; and the parameters $\delta_{mp}, \delta_{ms}\ and\ \delta_g$denote respectively small positive passband magnitude, stopband magnitude, and group delay tolerance limits. The design problem formulation in (1.19)-(1.22) can be generalized to the case of a highpass filter or a bandpass filter or a bandstop filter or a multi-band filter.

1.4 Quantization of coefficients

During the approximation step, the coefficients of the digital filter are calculated with the high accuracy inherent to the computer employed in the design. When these coefficients are quantized for practical implementations then the time and frequency responses of the realized digital filter deviate from the ideal response. In some cases, the quantized filter may even fail to meet the prescribed specifications. The sensitivity of the filter response to errors in the coefficients is highly dependant on the type of the structure.

The finite numerical resolution of digital filter representations has an impact on the properties of filters. The quantization of coefficients, state variables, algebraic operations and signals plays an important role in the design of recursive filters. Compared to non-recursive filters, the impact of quantization is often more significant due to feedback process. Several degradations from the desired characteristics are the potential consequences of a finite word length in practical implementations.

A recursive filter of the order $N \geq 2$ can be decomposed into the second-order-sections(SOS). Due to the grouping of poles/zeros to the filter coefficients with the limited amplitude range, a realization has been made by the cascaded SOS. The transfer function of SOS is given below:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad (1.24)$$

This can, however, be split into the recursive and non-recursive part. The transfer function of the recursive part of the filter is given below:

$$H(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad (1.25)$$

1.4.1 Initial Coefficient Values for Populations

Let $c_k^{[u]}$ and $c_k^{[l]}$ be the upper and the lower bounds for the kth coefficient c_k of a LP or HP or BP or BS prototype filter such that:

$$c_k^{[l]} \leq c_k \leq c_k^{[u]} \; for \; 1 \leq k \leq \frac{N}{2} + 1 \quad (1.26)$$

The initial coefficient c_{pk} for the population member p is computed by:

$$c_{pk} = c_k^{[l]} + rand * \left(c_k^{[u]} - c_k^{[l]}\right) \; for \; p = 1{:}P, k = 1{:}K \quad (1.27)$$

Where $rand$ is the uniformly distributed value between 0 and 1.

1.5 Filter Design Problems

There are basically two classes of digital filters, namely Finite Impulse response (FIR) and Infinite Impulse response(IIR). Due to the absence of a denominator we find that the FIR filters are more stable. FIR Filters are guaranteed to be of linear phase with the use of symmetric or asymmetric coefficients. FIR filters include general phase FIR Filters and Linear Phase FIR filters in which each of their transfer functions, frequency responses and group delay have been described earlier in this chapter.

For symmetric filters, there are four types of $(M-1)th$ order linear phase FIR digital Filter of length M depending on the number of points M of the impulse response and the type of symmetry.

Impulse response h, distinct coefficient vector c, the frequency responses $H(c, w)$ of Type I linear phase FIR digital Filter [25] is given below:

Type I	***M odd and even symmetry***
$\boldsymbol{h}$	$\boldsymbol{h} = [h(0), h(1), h(2), \ldots . . h(n), h(M-2), h(M-1)]^T$ $h(n) = h(M-1-n)\ for\ n = 0,1,2,3 \ldots . ., \left(\frac{M-3}{2}\right)$
$\boldsymbol{c}$	$\boldsymbol{c} = \left[c_0, c_1, c_2 \ldots \ldots c_{\left(\frac{M-1}{2}\right)}\right]^T$ $= \left[h\left(\frac{M-1}{2}\right), 2h\left(\frac{M-1}{2}-1\right), \ldots \ldots . ., 2h(2), 2h(1), 2h(0)\right]^T$
$H(\boldsymbol{c}, w)$	$e^{-j\left(\frac{M-1}{2}\right)wT}\left\{h\left(\frac{M-1}{2}\right) + \sum_{n=0}^{\frac{M-3}{2}} 2h(n)cos\left[\left(\frac{M-1}{2}-n\right)wT\right]\right\}$ $= e^{-j\left(\frac{M-1}{2}\right)wT} A(c, w)$
$A(\boldsymbol{c}, w)$	$A(\boldsymbol{c}, w) = c^T \cos w$ $\left[1 \quad cos(wT) \quad cos(2wT) \ldots \ldots . cos\left(\frac{M-1}{2}wT\right)\right]^T$

The number of impulse responses M is related to the filter order N by $M = N + 1$.

Problem formulation

An Nth order non-recursive digital filter can be represented by the transfer function

$$H(z) = \sum_{n=0}^{N} h_n z^{-n} = \boldsymbol{c^T z}(z) \tag{1.28}$$

Where $\boldsymbol{c^T}$ is the real coefficients vector. N is the total number of filter coefficients.$N-1$ is the order. For optimization problem, the coefficient vector is:

$$c^T = [c_1, c_2 \ldots \ldots . c_n] \tag{1.29}$$

The frequency response can be gained by substituting $z = e^{jTw}$ where T is the sampling period in seconds and w is the frequency.

For the design of the linear phase FIR digital filter, we assume $\sigma = w_{i,}\ 1 \leq i \leq M$, be the group of frequencies to evaluate the frequency response. Therefore the error at each sample point in w_i is given as:

$$e_i = H_d(w_i) - H(w_i) \tag{1.30}$$

For the symmetric digital filter the group delay is constant which is mentioned as:

$$\tau = \frac{N}{2} \tag{1.31}$$

The main aim of the Filter design problem is to find the optimal coefficient vector c that minimizes the magnitude and group delay errors. For linear phase, the group delay error is constant; so it minimizes only the passband and stopband magnitude errors whereas for the general phase FIR filter the coefficient vector c simultaneously minimizes the magnitude and group delay errors.

$$c = \min_{c} e(c) \tag{1.32}$$

For linear phase FIR filter, the minimax objective function $e(c)$ can be decomposed into passband magnitude error function $e_p(c)$ and stopband magnitude error function $e_s(c)$ as

$$e(c) = \left[e_p(c) + e_s(c)\right]^{1/p} \tag{1.33}$$

For the general phase FIR filter the joint objective function can be decomposed into passband magnitude error function $e_{mp}(c)$, stopband magnitude error function $e_{ms}(c)$ and group delay error $e_{gp}(c)$ defined by [24] as:

$$e(c) = \left[max\left(e_{mp}(c), e_{ms}(c)\right) + \alpha e_{gp}(c)\right]^{\frac{1}{2}} \tag{1.34}$$

In this study I have designed four types of linear phase FIR filters of orders 24 and 48. For order 48, the coefficient vector $\boldsymbol{c}$ will have 25 coefficient values whereas for order 24, the coefficient vector will have 13 values i.e. one more than the order of the filter. This is due to the symmetric nature of the filter. I have also designed two types of general phase FIR filter of order 24 where the coefficient vector will have 25 values due to its antisymmetric nature. All the coefficient values are in the range of -1 to 1.

In the section below, some important deterministic algorithms, heuristic, metaheuristic and evolutionary algorithms generally used for designing such FIR filter problems have been described. The objective of each of the algorithms is to minimize the objective error function for all the filter design problems.

1.5.1 Deterministic algorithms

Deterministic digital signal processing is procedure used to display the information in a measured data. The procedure utilizes different mathematical formulas and implements them with the help of digital techniques to get appropriate deterministic statistics. Finite impulse response filter is used in deterministic digital signal processing as a filter with impulse response to all finite length inputs. It is computed to settle at zero at its finite time.

These algorithms used specific rules for moving from one solution to another. These algorithms have been successfully applied to many engineering design problems. They always give the same output, with the underlying machine passing through the same sequence of states.

1.5.2 Heuristic algorithms

The Heuristic search method enhances the capability to explore and exploit locally as well as globally to obtain optimal design FIR Filter parameters. Heuristic algorithms are superior or atleast comparable to other algorithms and can be efficiently used for higher order filter designs.

Heuristic algorithms are the algorithms which are designed to solve the problems faster and in an efficient manner than traditional methods by sacrificing optimality, accuracy, precision or the completeness for speed. Heuristic algorithms are often employed with the approximate solutions that are sufficient and the exact solutions that are computationally expensive. The heuristic algorithms find solutions among all possible ones but they do not guarantee that the best solution will be found, and therefore they are considered as not-accurate algorithms. Approximate algorithms entail the interesting issue of quality estimations of the solution they find. These problems can be a real challenge in solving strong mathematical problems. The main goal of the heuristic algorithms is to find as good solution as possible to all instances of the problem.

Usually, heuristic algorithms are used for problems that cannot be solved [1]. Classes of time complexity are defined to distinguish the problems according to their hardness. Turing machines are an abstraction that is used to formulate the notion of the algorithm and also its computational complexity. Class P consists of those problems that are solved on a deterministic turing machine in polynomial time. Class NP consist of all those problems whose solution can be found in polynomial time on a non-deterministic Turing machine. A subclass of NP, Class NP-complete includes problems such as a polynomial algorithm for solving one of them can be transformed into polynomial algorithms for solving all other NP problems. The class NP-hard

can be understood as the class of problems that are NP-complete or harder. Some of the heuristic algorithms are:

Swarm intelligence which employs a large number of agents interacting locally with one another and the environment.

Tabu Search which uses dynamically generated tabus to guide the solution search to optimum solutions. It examines the potential solution to the problem and checks the local intermediate neighbours to find the improved solution.

Simulated Annealing is used in global optimization to give a reasonable approximation of a global optimum in a function for the search space.

1.5.3 Metaheuristic algorithms

Metaheuristic algorithms are basically higher level heuristic algorithms which are used for IIR filter designs. The term 'meta' means higher-level or beyond , so metaheuristic means literally to find the solution using higher-level techniques. They are considered as higher-level techniques or strategies which intend to combine with lower level techniques for exploration and exploitation of the huge space for parameter search when used in filter design problems.

Metaheuristic algorithms are a combination of heuristic and randomization. It is formally defined as an iterative generation process which guides a subordinate heuristic by combining the different concepts intelligently for exploring and exploiting the search space. The main goal of metaheuristics is to efficiently explore the search space in order to find the optimal solutions. The techniques which constitute the metaheuristic algorithms range from the simple local search procedures to complex learning processes. Metaheuristic algorithms are non-deterministic algorithms. They incorporate mechanisms to avoid getting trapped in the confined areas of the search space.

Metaheuristics are not problem specific. They usually make use of the domain-specific knowledge in the form of heuristics that are controlled by the upper level-strategy. Today's more advanced metaheuristics make use of the search experience to guide the search. The metaheuristic is a general algorithm framework for addressing the interactable problems. Metaheuristics are approximation algorithms that cannot always produce provably optimal solutions but they do have the potential to produce good solutions in a reasonable amount of time.

1.5.4 Evolutionary algorithms

Evolutionary Algorithms are used in finding the solution for problems where there is no explicit solution and that is what is exactly required for digital filter design problem. Their particular strength is that they can efficiently search for a solution in a very large space.

Evolutionary Algorithms(EA) consist of several heuristics, which are able to solve optimization tasks by initiating some aspects of natural evolution. They may use different levels of abstraction, but they are always working on the populations of possible solutions for a given task. Evolutionary methods are used in hard optimization problems rather than pattern recognition.

1.5.4.1Genetic algorithms

In nature, every living organism has a set of rules, a blueprint so to speak and describing how the organism is created. The genes of an organism represent these rules and are connected together into long strings called chromosomes. Each gene represents the specific property of an organism and the collective set of gene settings are referred to as organism's genotype. The physical expression of the genotype is called the phenotype. Yet in rare cases, it will be expressed in the organism as a completely new trait. It is a local search technique used to find approximate solutions to optimization and search problems. Genetic algorithms are a particular class of Evolutionary algorithms that use techniques inspired by evolutionary biologies such as inheritance, mutation, selection, and crossover. They are typically implemented as a computer solution in which a population of abstract representations of candidate solutions to an optimization problem evolves towards better solutions. The evolution normally starts from a population of completely random individuals and occur in generations. In each generation the fitness of the whole population is evaluated, multiple individuals are stochastically selected from the current population and modified to form a new population. The new population is then used in the next iteration of the algorithm. Genetic Algorithms uses *crossover* and *mutation* as a search mechanism. Some applications of genetic algorithms are as follows:

1) Automotive design

2) Engineering design

3) Robotics

4) Evolvable hardware

5) Optimized telecommunications routing

1.5.4.1 Differential evolution

In evolutionary computation, differential evolution is a method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Here each variable's value is represented by a real number. Differential Evolution is a design tool of great utility that is accessible for practical applications. DE has been used in several scientific and engineering applications to discover effective solutions to nearly intractable problems without appealing to expert knowledge or complex design problems. If a system is amenable to be rationally evaluated, DE can provide the means for extracting the best possible performance from it. The Differential Evolution uses mutation as a search mechanism and selection to direct the search towards the prospective regions in the feasible region. DE is a population based search technique which utilizes NP variables as a population of D dimension parameter vectors for each generation. The initial population is chosen randomly. In the case of the available preliminary solution, the initial population is generated by adding normally distributed random deviations to the preliminary solutions. The basic idea behind DE is a new scheme for generating trial parameter vectors. If the resulting vector yields a lower objective function value than the predetermined population member, the newly generated vector replaces the vector with which it was compared. The best parameter vector is evaluated for every generation in order to keep track of the progress that is made during the optimization process. DE maintains two arrays each of which holds a population size NP and D dimensional real-valued vectors. The primary array holds the current vector population, while the secondary array accumulates vectors that are selected for the next generation.

Table 1.3 Comparing the performance of three metaheuristic algorithms using Test functions [26]

No	Function MS	DE		PSO		HSA	
		Reached optimal	Average	Reached optimal	Average	Reached optimal	Average
1	Booth	0	0	0	0	9.49694e-010	2.21531e-009
2	Rastrigin	0.00589048	2.84916	15.9194	22.6993	4.02884e-005	0.000101882
3	Schwefel	-12569.5	-12569.5	-11740.4	-11740.4	-12569.5	-12569.5
4	Michalewicz10	-9.66015	-9.66015	-9.65277	-9.48277	-9.66015	-9.66015

The above table shows that for the Multimodal Separable functions, the Differential Evolution (DE) and Harmony Search (HS) algorithms are a very efficient method for finding the optimal solution and its convergence speed is much faster than the Particle Swarm optimization.

1.6 Neural Networks

A neural network is a massively parallel structure which is composed of many nonlinear processing elements connected to each other through weights. It is a trainable nonlinear dynamic system which stores various patterns with distributed coding. When compared with sequential digital computers, we find that neural networks have a faster response due to parallel processing and a higher performance due to nonlinear processing.

Neural networks basically have a multilayer structure consisting of a sigmoidal type of nonlinear operation at the output of each hidden neuron and the output of each output neuron. Neural network classifiers are free model estimators. They usually do not provide assumptions on how the outputs depend on the inputs. Instead, they decide the boundaries of the classes and adjust themselves to the training set by the learning algorithm.

1.7 Contributions

The main contribution of the work done here is the implementation of HS Algorithm and adapting the algorithm to utilise it in designing FIR filters and neural networks. The algorithm has been initially used for several other applications mentioned in Chapter 2. The disadvantage of the other algorithms like GA and PSO is that they require fine tuning of parameters in order to obtain a feasible solution. Also, diversification and intensification are the two major components whose balanced combination is very important for the success of any metaheuristic algorithm. Harmony search successfully balances these two major components by pitch adjustment and harmony considering rate and therefore it ensures a certain level of efficiency and that the evolving system will not get trapped in the local minima.

1.8 Motivation and Outline of Thesis

The main objective of this project is the requirement of an efficient optimization algorithm that will be able to optimize complex designs problems such as for the advanced digital filters and neural networks making the filter processes much more efficient and noise free. Since the algorithm produced some effective results in the initial runs, hence it motivated me to explore more with the algorithm by applying it to different design problems and comparing the results

with those already present algorithms. The coefficient values have to be optimized using an optimization algorithm for reducing the noise to the minimum and designing a good filter for real time applications. Likewise, for designing complex neural networks optimization algorithm plays a vital role in determining the weights and bias of the network. The weights and bias are the important optimization aspects which helps to reduce the error between the actual output and the desired output in such a way that the neural network will give the same output even by introducing some amount of noise at the input. There are some algorithms based on evolutionary methods which can design digital filters with comparatively less time. Various types of complex neural network designs have been adopted from [18] in my thesis which includes two layers neural network using XOR, Advanced feedforward neural networks (0-9) digit and also the two layer neural network design results for fuzzy inference networks.

The thesis clearly presents Harmony Search algorithm and its applications in designing the advanced digital filter designs and neural network problems.

The first chapter illustrates the theory about the digital filters, the various types of digital filters, the comparison between the two types and details about the types of filters have also been described. The conventional methods used for designing filters and common strategies used for the same have also been highlighted in Chapter 1. It basically contains the main goals of the thesis.

In the second chapter, some of the state of art of methods for designing the FIR digital filter designs are discussed and the literature has been reviewed. The Harmony Search Algorithm, its pros and cons, strengths and weakness along with the improvements and the basic introduction to neural networks have been discussed in Chapter 2.

The third Chapter describes the methodology or the creation of work required to obtain each of the Neural Networks and filter design results mentioned in the Chapter 4.

Chapter 4 shows the evidences that the harmony search Algorithm can be used as a good alternative to the Parks McClellan algorithm through the results which have been shown.

Chapter 5 states the conclusion of the thesis.

Chapter 2:

Review of Literature

Digital filtering is a ubiquitous operation in digital signal processing applications and is realized using infinite impulse response(IIR) or Finite impulse response(FIR) filters. Although FIR Filter requires a large number of coefficients when compared to the IIR Filters, it is compared to IIR Filters due to stability and phase linearity properties.

An optimization algorithm is a procedure which is executed iteratively by comparing various solutions to an optimum solution is found. The main objective of an optimization algorithm could be simply to minimize the cost of production and to maximize the efficiency of the of the production. In an optimal problem formulation, the optimal design is achieved by comparing few alternative solutions. In this particular method, the feasibility of each design solution is first investigated. Therefore, an estimate of underlying objective of each solution is compared with the others and the best solution is recorded. The design parameters, however, can vary from product to product. The purpose of the formulation is to create the mathematical model of the optimal design problem, which can be solved using optimization algorithm.

The formulation of optimization algorithm begins with identifying the design variables which are primarily varied during the optimization process. The design problem involves many design parameters, of which some are highly sensitive to the proper working of the design. These parameters are known as the design variables. The constraints represent some functional relationships among the design variables and parameters satisfying certain physical phenomenon and certain resource limitations. There are mainly two types of constraints: Inequality constraints and Equality constraints.

The learning problem in neural networks is formulated in terms of minimization of a loss function. The function is composed of error and regularization terms. The error term mainly evaluates how the neural network problem fits the data set. The regularization is used to prevent overfitting, by controlling the effective complexity of the neural network design. The loss function depends on adaptive parameters (bias and synaptic weights) of the neural network. We can group them conveniently into a single n-dimensional weight vector w. There are many training algorithms that can be used in the training process of the neural network.

In this thesis study, I have made use of the Harmony Search Algorithm to solve the complex neural network design problems. This Algorithm has also been used in designing advanced

digital filter designs which can be used for many real-time applications. Digital filters are an essential part and one of the most important features of modern day circuit designs and plays a vital role in the improvement of the overall system by producing better results. Harmony Search Algorithm is just a successful example which transforms the qualitative improvisation process into some quantitative rules by idealization, thus turning the beauty and harmony of music into an optimization procedure through a search for a perfect harmony.

2.1 Survey of Harmony Search Applications

In the real world, modern science and industry are rich in the problems of optimization. HS was originally proposed by Geem [2] and applied to solve the optimization problem of water distribution networks in 2000, the applications of the HS have covered many areas including industry, optimization benchmarks, power systems, medical science, control systems, construction design, and information technology [3].

The Industry is a prominent area full of various practical optimization issues subject to multi-modal, constrained, nonlinear, and dynamical. The HS algorithm proposed by Saka [4] determines the optimal steel section designations from the available British steel section table and implements the design constraints from BS5950. Recently, an enhanced harmony search (EHS) in [5] is developed enabling the HS algorithm to quickly escape from local optima. The proposed EHS algorithm is utilized to solve four classical weight minimization problems of steel frames including two-bay, three-storey planar frame subject to a single-load case, onebay, ten-storey planar frame consisting of 30 members, three-bay, twenty-four storey planar frames, and Spatial 744 member steel frame. In [6], the HS is used to select the optimal parameters in the tuned mass dampers [6]. Fesanghary et al. [7] propose a hybrid optimization method based on the global sensitivity analysis and HS for the optimal design of shell and tube heat exchangers. There is a lot of work focused on the optimization issues concerning power systems, such as cost minimization. A modified HS algorithm is proposed to handle non-convex economic load dispatch of real-world power systems. The economic load dispatch and combined economic and emission load dispatch problems can be converted into the minimization of the cost function [8]. Sinsuphan et al. [9] combine the HS with sequential quadratic programming and GA to solve the optimal power flow problems. The objective function to be optimized is the total generator fuel costs in the entire system. The chaotic self-adaptive differential HS algorithm, proposed by Arul et al. [10], is employed to deal with the dynamic economic dispatch problem. Li and Duan [11] modify the HS by adding a Gaussian

factor to adjust the bandwidth (bw). With this modified HS, they develop a pre-training process to select the weights used in the combining of feature maps to make the target more conspicuity in the saliency map. In their method based on the HS, Fourie et al. [12] design a harmony filter using the improved HS algorithm for a robust visual tracking system.

2.2 What is Harmony Search Algorithm?

Harmony Search Algorithm is an emerging metaheuristic algorithm that was inspired by the observation that the aim of music is to search for a perfect state of harmony. There is a parallel idea between HS and how the Jazz musicians create harmony when they play music.

HS algorithm is based on a few parameters: *hmcr*, *par*, and *bw*. The parameter *hmcr* is called the Harmony memory considering rate and it denotes the rate of choosing candidates from the Harmony Memory(HM) and generally ranges from 0.7 to 1. The parameter *par* is called the pitch adjusting rate and indicates the rate of choosing the neighboring value and can be selected from 0 to 1. The parameter *bw* is called the bandwidth denotes the amount of maximum rate of change of change in the pitch adjustment.

2.3 Design of Harmony Search Algorithm

This section discusses the projected effective harmony Search. Initially a brief outline about HS is given and lastly, the alteration procedures of the proposed effective Harmony Search are discussed.

Harmony Search Algorithm is one of the efficient optimization algorithm developed by Geem et al. [13]. It is inspired by the music improvisation process. The analogy between music improvisation and optimization can be established by creating a correspondence between music player to the decision variable. In order to execute the technique in real time optimization each decision variable chooses and possible range together to make a solution. This solution is then improved by creating harmony memory, and pitch adjusting. Over the year various optimization algorithms [14-16] are proposed but HSA remains one of the best choices for function optimization. In order to introduce the HS algorithm for engineering optimization. [17].

The various involved in Harmony Search algorithm are discussed as follows [13]:

Step 1 Initialize harmony memory.

Step 2 Improvising new harmony vectors.

Step 3 Update Harmony Memory.

Step 4 Check Stopping criterion.

During optimization, harmony search algorithm attempts to find the harmony vector x which minimizes (or maximizes) a specified objective function $f(x)$. The algorithm consists of four steps which are described below:

The above four steps are discussed in the following subsections [1]-[4] [18]:

1. **Initialize the harmony memory:** In this step, each parameter m for $m = 1\ to\ M$ of each of the P initial harmony vectors $x_p = [c_{p1}, c_{p2}, c_{p3}, \ldots., c_{pm}]$ for $p = 1\ to\ P$ is generated randomly within the upper limit u and the lower limit v of the mth parameter as:

$$c_{pm} = v + rand * (u - v)\ for\ p = 1\ to\ P\ and\ m = 1\ to\ M \quad (2.1)$$

The harmony memory consisting of x_p for $p = 1\ to\ P$ are arranged in the ascending order of increasing objective function $f(x_p)$ for $p = 1\ to\ P$ such that $f(x_1) \leq f(x_2) \leq f(x_3) \ldots. \leq f(x_p)$ as:

$$\begin{bmatrix} c_{11} & \cdots & c_{1p} \\ \vdots & \ddots & \vdots \\ c_{p1} & \cdots & c_{pm} \end{bmatrix} \quad (2.2)$$

2. **Improvising a new solution**- In this step, HS improvises Q harmony vectors $x'_m = [c'_{q1}, c'_{q2}, c'_{q3}, \ldots\ldots, c'_{qm}]\ for\ q = 1\ to\ Q$ based on two considerations, namely memory considerations and pitch adjustment.

   For the memory consideration, each parameter value c'_{qm} for $m = 1\ to\ M$ of the new harmony vector is randomly selected among the corresponding P parameter values $\{c_{1m},\ c_{2m},\ c_{3m,\ldots\ldots\ldots},\ c_{pm}\}$ with a probability $C \in [0,1]$.

$$c'_{qm} = \begin{cases} c'_{qm} \in \{c_{1m}, c_{2m}, c_{3m}, \ldots, c_{pm}\}\ C \in [0,1] & (2.3a) \\ c'_{qm} = c'_{qm}(old) & (2.3b) \end{cases}$$

For the pitch consideration, an additional search for good harmony in the search space is achieved by adjusting each continuous parameter c'_{qm} in a new solution vector at a pitch adjusting rate (A) for $A \in [0,1]$ as

$$c'_{qm} = \begin{cases} c'_{qm} + randn * bw & (2.4a) \\ c'_{qm} = c'_{qm} & (2.4b) \end{cases}$$

where $randn$ is the normally distributed pseudorandom numbers and bw is an arbitrary distance bandwidth which determines the maximum change to each parameter of the new harmony vector.

The upper and lower bound are applied to each parameter of each of the Q new harmony vectors such that:

Replace any new parameter $c'_{qm} < v$ $(2.5a)$

Replace any new parameter $c'_{qm} > u$ $(2.5b)$

3. **Harmony Memory Update**: In order to update the population with Q new harmony vectors $x'_q = [c'_{q1}, c'_{q2}, c'_{q3}, \dots, c'_{qM}]$ for $q = 1\ to\ Q$, the objective function $f(x'_q)$ of each of the harmony vectors is calculated. Among the P existing harmony vectors and Q new harmony vectors, the top P harmony vectors based on the ranking of their objective functions are selected for next improvisation.
4. **Stopping criterion**: The improvisation process in steps 3 and 4 is terminated when the maximum number of improvisations is reached. Finally, the best harmony memory vector x^{best} is selected among all the harmony vectors to be the solution for the optimization problem.

$$\boldsymbol{x^{best} \in (x_1, x_2, x_3 \dots \dots x_p)} \qquad (2.6)$$

2.4 Improvements in Harmony Search Algorithm

Because the effects of optimization are mostly depended on the initialization of HM, the selection of parameters such as HMCR and the new ways of solution. So, if the harmony search algorithm faces with bad optimization or unsuitable parameters selection or complex optimization objective, some shortage such as weak local searching ability and convergence precision will appear. Since the harmony search algorithm was invited, some experts put forward many effective suggestions to improve the algorithm, which include two aspects, one is improving algorithm itself (including HM parameters optimization) and other is combining with other algorithms.

Basic Harmony Search Algorithm uses fix value for PAR and BW and initializes these parameters in step 1, and these parameters cannot be updated in the iterative process. Because the small PAR and big BW results in bad algorithm effect, so it needs to add definite iterative items to find the best value.

In 2007, Mahdevi put forward the Improved Harmony Search Algorithm(IHS), it brought forward the dynamic adaptive PAR and BW strategy, and the two parameters can vary dynamically between maximum and minimum, while the PAR was increasing linearly with the number of iterations and the BW index decreasing with the number of iterations.

$$PAR(gn) = PAR_{min} + \frac{(PAR_{max} - PAR_{min})}{NI} \times gn \quad (2.7)$$

Where,

$PAR(gn)$ = Pitch Adjusting Rate for each generation

PAR_{min}= Minimum Pitch Adjusting Rate

PAR_{max} = Maximum Pitch Adjusting Rate

NI = Number of Solution vector generation

gn = Generation Number

$$bw(gn) = bw_{max} exp(c.gn) \quad (2.8)$$

$$c = \frac{ln\left(\frac{bw_{min}}{bw_{max}}\right)}{NI} \quad (2.9)$$

2.5 Why is Harmony Search successful?

Presently in the event, when we compare HS with other metaheuristic algorithms, we can recognize its methods for taking care of intensification and diversification in the HS system, and presumably, comprehend why it is an exceptionally fruitful metaheuristics calculation. In the HS calculation, diversification is basically measured by the pitch adjustment and randomization [19]. In this case, there are two subcomponents for diversification, which may be an essential component for high proficiency of the HS strategy. The principal subcomponents of making new music or creating new arrangements by means of randomization would be in any event at the same level of effectiveness as different calculations by randomization. In any case, an extra subcomponent for Harmony search diversification is the pitch adjustment. Pitch adjusting is carried out by adjusting the pitch in the given bandwidth by a small random amount relative to the existing pitch or solution from the harmony memory. Pitch adjustment is the refinement process of local solutions. Both the memory considering and the pitch altering

guarantee that the great local arrangements are held while the randomization and the harmony memory considering will investigate the worldwide search space adequately. The randomization explores the search space more efficiently and effectively; while the pitch adjustment ensures that the newly generated solutions are good enough, and are not too far from the existing good solutions. The intensification factor in HS is mainly represented by Harmony Memory Accepting Rate r_{accept}. A high Harmony acceptance rate means that the good solutions from the history/memory are likely to be selected or inherited. Otherwise, if the acceptance rate is too low then the solutions will converge more slowly. This intensification is enhanced by the controlled pitch adjustment. Such interactions between various components could be another important factor for the success of HS algorithm over other algorithms.

2.6 Introduction to Neural Networks

Neural networks have the potential for very complicated behavior and their ability to learn is one of their major advantages over the traditional non-linear system. The massive interconnections for inter-processing units in multilayer networks provide the tool for neural network models. Neural networks are currently used for pattern recognition and fuzzy logic as well as in control.

Computers are great at solving algorithmic and math problems, but often the world can't easily be defined with a mathematical algorithm. The key to Artificial Neural Networks(ANNs) is that their design enables them to process information in a similar way to our own biological brains, by drawing inspiration from how our own nervous system functions.

One of the most impressive features of Artificial Neural Networks is their ability to learn. The artificial neural networks are inspired by the biological nervous system, especially the brain. ANNs can model the learning process by adjusting the weighted connections found between the neurons in the network. This effectively emulates the strengthening and weakening of synaptic connections found in our brains. The strengthening and weakening of connections is what enables the network to learn. Learning algorithms are extremely useful when it comes to certain problems that either can't be practically written by a programmer or can be done more efficiently by a learning algorithm.

There are different algorithms that can be used for training Artificial Neural Networks, each with their own separate advantages and disadvantages. The learning process within artificial

neural networks is the result of altering the network's weights with some kind of learning algorithm. The objective is to find the set of solution matrix of the weights.

There are three major learning paradigms:

a) **Supervised learning:** The learning algorithm would fall under this category that if the desired output for the network is provided with the input while training the network. By providing the neural network with both the input and output pair, it is possible to calculate an error based on its target output and the actual output. It can then use that error to make connections to the network by updating its weights. Supervised learning is performed off-line.

b) **Unsupervised learning:** This learning algorithm uses no external teacher and is based on the only local information. It is also referred to as self-organization, in the case that it self-organizes the data presented to the network and detects their emergent collective properties. Paradigms of unsupervised learning are Hebbian learning and competitive learning. From human neurons to artificial neurons, the aspect of learning concerns the distinction or not of separate phase, during which the network is trained and a subsequent operation phase. A neural network leans-on-line if it learns and operates at the same time. Unsupervised learning is performed on-line.

2.7 Why Neural Networks?

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers take an algorithmic approach; i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer is not able to solve the problem. This restricts the problem-solving ability of conventional computers to problems that we already understand and know how to solve. Neural networks process information in the same manner that the human brain does. The network is composed of a large number of highly interconnected processing elements(neurons) working in parallel to solve a particular problem. Neural networks learn by example. They cannot be programmed to solve a particular task.

Neural networks are widely used in pattern recognition because of their ability to generalize and to respond to unexpected inputs/patterns. During training, neurons are taught to recognize various specific patterns and whether to fire or not when that pattern is received. If a pattern is received during the execution stage that is not associated with the output, the neuron selects the

output that corresponds to the pattern from the set of patterns that it has been taught of, that is least different from the input. This concept is called generalization.

2.8 Types of Neural Networks

2.8.1 Feedforward neural networks

Feedforward neural networks [18] are artificial neural networks where the connections between units do not form a cycle. Feedforward neural networks were the first type of artificial neural networks invented and are simpler than their counterpart. They are called feedforward because information only travels forward in the network, first through the input nodes, then through the hidden nodes and finally through the output nodes.

Feedforward neural networks are primarily used for supervised learning in cases where the data needs to be learned is neither sequential nor time dependent. That is, feedforward neural networks compute a function f on fixed size input x such that $f(x) \approx y$ for training pairs (x, y).

Feedforward neural networks are the ideal candidates for performing classification (e.g. categorical) tasks, and the activation of hidden units can be used to analyze their internal, categorical classifications.

2.8.2 Fuzzy neural networks

A fuzzy neural network [18], [20]-[21] combines the features of fuzzy systems (with an ability to process fuzzy information using fuzzy algorithms) and the features of neural networks (with a learning ability and a high speed parallel structure) to form a network which can learn from the given data and environments.

There are mainly three types of fuzzy neural networks [18], [20]-[21], namely Min-Competitive Fuzzy Neural Network (MCFNN), Min-Max Fuzzy Neural Network (MMFNN) and Min-Sum Fuzzy Neural Network (MSFNN) which can be designed for pattern classification, recognition, interpolation and other applications.

The MCFNN can be combined with the Maximum Fuzzy Neurons (Max-FNs) and the Input Fuzzy Neurons (Input-FN's) to form a fuzzy neural network for pattern recognition [18], [20].

The MCFNN, MMFNN and the MSFNN can be used for non-fuzzy and fuzzy pattern classification [18], [21].

Efficient self-organizing learning algorithms can be used for training the networks. After being trained by the labeled data, each of the fuzzy neural networks can find the fuzzy and hard partition between the classes. Each fuzzy neural network can build the decision boundaries by creating subsets of the pattern space. These are free model estimators and do not assume how the outputs depend on the inputs. Instead, each of them adjusts itself to a given training set by learning algorithms and decide the boundaries of classes. When given an unknown pattern, each fuzzy network uses the used the learned knowledge to estimate the membership value of the pattern in each class and classify the input pattern according to the membership values. Each of the fuzzy neural networks is represented by a set of fuzzy inference rules and these networks have been used for various applications.

Chapter 3:

Methodology and Analysis

3.1 FIR Filter Design

In this thesis, I have designed two types of FIR Filters using Harmony Search Algorithm Linear Phase FIR and General Phase FIR [25].

The objective function and problem formulation for a specific type of filter and neural network; the parameters of the optimization methods have been employed in this section. The objective is to minimize the magnitude and group delay error of FIR filters and the search continues until the objective function converges.

The transfer function $H(c,z)$ of a digital filter with the coefficient vector $\boldsymbol{c}$ of dimension $\boldsymbol{K \times 1}$ is given, its frequency response $H(c,w)$ can be expressed in terms of magnitude response $|H(c,w)|$ and phase response $\theta(w)$ as:

$$H(c,w) = |H(c,w)|e^{j\theta(c,w)} \tag{3.1}$$

The group delay $\tau(c,w)$ of a digital filter can be computed by taking the negative partial derivative of the phase response $\theta(c,w)$ with respect to the frequency w as:

$$\tau(c,w) = -\frac{\partial\theta(c,w)}{\partial w} \tag{3.2}$$

Then the objective function $\boldsymbol{e(c)}$ of the least pth frequency response error is defined by:

$$\boldsymbol{e(c)} = \sum_{0}^{\pi} W(w)\left|H\left(c, w - H_d(w)\right)\right|^p \tag{3.3}$$

The discrete frequency weighing function $W(w)$ can be normalized such that

$$\sum_{i=1}^{I} W(w_i) = 1 \tag{3.4}$$

Where I= Number of frequency points

The optimization problem is to search for an optimal coefficient vector $\boldsymbol{c}$ that minimizes the objective function $e(c)$ as:

$$\min_{c} e(c) \qquad (3.5)$$

For each of the linear phase FIR Filters the passband magnitude error, stopband magnitude error is calculated and then the total error is calculated which needs to be optimized whereas for each of the general phase FIR Filters the magnitude as well as the group delay error is calculated.

3.2 Neural Network Design

3.2.1 Design using XOR neural network

One of the common backpropagation problems that can be solved using neural networks is the Exclusive-OR problem [18] which requires the network to be trained in such a manner that it is able to produce the similar inputs and distinguished input results separately, with similar inputs producing 0 at the output and different inputs with 1 at the output. A network has been designed for this particular problem with 2 hidden neurons as shown in Figure 3.1. The network contains 9 parameter values (6 weighing coefficients, 3 bias values) that need to be optimized in order for the network to produce successful exclusive-OR results which are later tested and verified in order to prove the neural network parameters produce the same results for all sort of input noise and values.

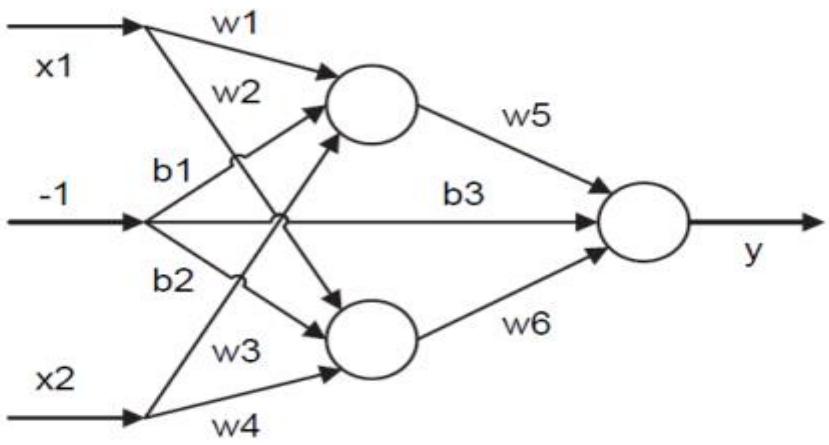

Figure 3.1 Neural network design for two input XOR problem [18]

The neural network consists of two inputs (denoted by x1 and x2), one bias value of -1 denoted by b1 and b2 to the input of each of the two hidden neurons; two hidden neurons with outputs denoted by ($y_1^{[1]}$ and $y_2^{[1]}$).

The sum of the scaled inputs of each of the three neurons $z_i^{[h]}$ for $i = 1,2; \ldots . h = 1,2$ is passed through binary sigmoid activation function y with the slope parameter defined by:

$$y_i^{[h]} = \frac{1}{1 + e^{-rnz_i^{[h]}}} \qquad (3.6)$$

The slope parameter determines the slope of the transition region within the range $-L \leq z \leq L$ as well as determining the value of $\pm L$. The function saturates to 1 or 0. The slope parameter here is assumed as 2.

The objective function to be minimized is equal to the sum of absolute output differences over the four XOR patterns:

$$e(c) = \sum_{i=1}^{4} |y_1^2 - y_i| \qquad (3.7)$$

Stepwise Procedure for execution:

a. *The inputs to each of the two hidden neurons have been defined as* $x1$ *and* $x2$ *along with the output y. All these values were defined inside a for loop which is created for the four XOR patterns.*
b. $z1$ *and* $z2$ *is calculated based on the two scaled inputs and its respective weighing coefficients* $w1, w2, w3$ *and* $w4$.
c. *The* $sigmoid$ *activation function y for each value of i is calculated based on the slope of the transition range* r_n *and the respective values of* $z1$ *and* $z2$.
d. *Then the net output y of the XOR neural network is calculated using equation (3.6).*
e. *The last step is to calculate the value of e(c) which is the sum of the absolute output differences over the four XOR patterns and serves as the objective function required to be minimized by the Harmony Search Algorithm.*
f. *The search continues until the objective function is optimized and the output obtained is equal to the pattern of the ideal output.*

3.2.2 Design using feedforward neural networks

This is a simple design for a feedforward neural network problem using the simplified sigmoid function [18]. The idea is to simplify a more complex neural network problem with a large number of input and weighing function values in order to produce predefined output results. In

this particular problem, the first layer contains 100 bits as input which may be -1 or +1. There are 10 hidden neurons in the second layer and 4 output neurons in the third layer. The output of a neuron is the sigmoid activation function produces either negative or positive 1. In this multilayer Feedforward neural network system, the output of a neuron j at a layer h due to a kth input pattern X_K can be expressed as:

$$y_{jk}^{[h]} = F\left(\sum_{i=1}^{N_h-1} w_{ij}^{[h]} y_{ik}^{[h-1]} + b_j^{[h]}\right) \tag{3.8}$$

for $j = 1\ to\ N_h$, $h = 1\ to\ L$, $k = 1\ to\ K$

In equation (3.8), $y_{ik}^{[h-1]}$ is the output of the neuron i at layer $h-1$, $w_{ij}^{[h]}$ is the weight between a neuron i at layer $h-1$ and a neuron j at layer h, $b_j^{[h]}$ is the bias of the neuron j at the layer h; N_h is the number of neurons at layer h.F represents sigmoid activation function.

With 1 hidden layer, 10 hidden neurons and 4 output neurons, the hundred bits of input require 1000 weighing functions needing to be trained in the first layer. For the second stage, the outputs of each of the hidden neurons calculated by using eqn. 3.8 are then again multiplied with a weight thus requiring 40 more weights to be optimized for the 4 output neurons. Each neuron requires a bias that also needs to be optimized to a certain value in order to produce the desired output results therefore 14 neurons will require 14 bias coefficients to be trained during the optimization. Therefore, a total of 1054 parameters comprising of 1040 weight values and 14 bias values are required to be optimized in order to achieve a predefined set of output values. These parameters are optimized using Harmony Search algorithm and are then compared to the predefined output function values.

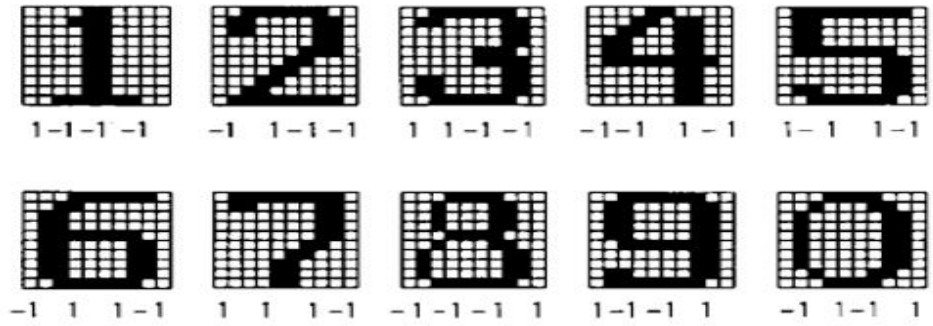

Figure 3.2 Training pattern pairs [22]

Stepwise procedure for execution:

a. *The first hundred bits of input is defined using a matrix for each of the 0-9 pixels which may be -1 or +1.*
b. *The output required to be obtained is defined.*
c. *The switch case concept has been applied to the program in order to increase the efficiency of the program. For each of the parameters the number of hidden neurons required to be used is defined in this section. For instance, for 1054 parameters the number of hidden neurons required to be used is 10. Similarly for 844 parameters the number of hidden neurons required is 8.*
d. *The first layer is defined with the number of weighing functions required to be trained for 100 bits of inputs and its sum is calculated. A for loop is used for defining the number of patterns and number of hidden neurons. Each input pattern contains hundred digits which is also defined using a for loop. The number of weighing functions required to be trained depends on the number of hidden neurons multiplied by the number of input bits.*
e. *In the second layer the output for each of the hidden neurons is calculated using equation 3.8 and the number of weights required to be optimized is calculated by multiplying the number of hidden neurons by four output neurons. The number of k patterns and output neurons is defined using a for loop in layer 2.*
f. *In the next step the bias coefficients are trained during the optimization procedure. The number of bias coefficients is calculated based on the summation of the number of hidden neurons and the four output neurons. This is how the number of parameters is calculated in order to achieve a predefined set of output values.*

g. *The parameter values and the output function values are calculated along with the errors for each of the four output neurons. The error is calculated as the difference between the absolute values of actual output and obtained output. Then the Mean square error of each of the four output neurons have been calculated by taking the sum of square of the errors and dividing the sum of squares by forty.*

h. *The parameters are optimized using Harmony Search Algorithm and then compared to the predefined output function values. The search continues until the obtained output is equal to the ideal output (without noise) and the Mean square errors obtained are almost equal to zero.*

3.2.3 XOR design using min-sum fuzzy inference network

Min Sum Fuzzy Inference Network MSFIN) [18], [21] is a three-layer feedforward network. TRAN-FN's are used in the first layer and MIN-FN's are used in the second layer for the MSFIN. The weight functions from the first to the second layer of the MSFIN are different from those of MMFIN (Minmax Fuzzy Inference Networks). The algorithms for the jth MIN-FN in the second layer of the MSFIN classifier are as follows:

$$s_j^2 = \min_{i=1}\left(w_{ij}^{[1]}(x_i)\right) \qquad j = 1,2 \dots \dots \dots M \qquad (3.9)$$

$$w_{ij}^{[1]}(x_i) = \begin{cases} 1 + a_{ij}^{[1]}(x_i - \theta_{ij}) & if\ 1 \geq 1 + a_{ij}^{[1]}(x_i - \theta_{ij}) \geq 0 \\ 1 - a_{ij}^{[1]}(x_i - \theta_{ij}) & if\ 1 \geq 1 - a_{ij}^{[1]}(x_i - \theta_{ij}) \geq 0 \\ 0, & otherwise \end{cases} \qquad (3.10)$$

where $a_{ij}^{[1]}, a_{ij}^{[2]}\ and\ \theta_{ij}$ are the parameters of the triangular membership function which are to be determined by the learning algorithm.

SUM-FN is used in the third layer of MSFIN. The algorithm is:

$$s_p^3 = \begin{cases} \dfrac{\sum_{j=1}^{M} w_{jp}^2 s_j^2}{\sum_{j=1}^{M} s_j^2} & if \sum_{j=1}^{M} s_j^2 \neq 0 \\ 0 & if \sum_{j=1}^{M} s_j^2 = 0 \end{cases} \qquad (3.11)$$

w_{jp}^2 is the connection weight between the jth MIN-FN in the second layer and the pth SUM-FN in the third layer. The training data sets for the fuzzy exclusive XOR Problem is recorded in TABLE 3.1.

Stepwise procedure for execution:

a. *In the first section the number of training data sets(Table 3.1) and the number of hidden neurons is defined in a for loop and the triangular membership function is calculated using equation 3.10. The minimum of each neuron is calculated. The sum of the first layer is calculated.*
b. *In the second layer s_j i.e. the sum from the first layer is multiplied by the corresponding connection weight using equation (3.11).The algorithm for the jth MIN-FN is calculated here. Then the sum of the second layer is calculated.*
c. *The third layer calculates the pth SUM-FN. The errors for each of the neurons is calculated by taking the difference of the absolute values of actual output and obtained output. The Mean Square error is calculated by dividing the sum of square of the errors by nine since the number of training patterns are nine.The error rate is also calculated.*
d. *The errors for each of the neurons are required to be minimized using Harmony Search Algorithm and the search continues until the desired results are obtained.*

Table 3.1 Training data sets: Nine training Samples for Fuzzy Exclusive XOR Problem

X_2	0.0	0.5	1.0
X_1	y_1, y_2	y_1, y_2	y_1, y_2
0.0	0,1.00	0,0	1.00,0
0.5	0,0	0,0	0,0
1.0	1.00,0	0,0	0,1.00

For each of the objective functions that are designed for the respective problem, the Harmony search algorithm has been coded in the main function using all its relevant control parameters

and has been implemented to design both the FIR Filters and the Neural Networks. The algorithm was found to be successful due to its extensive exploration and exploitation property in the search space. Though the algorithm has been found to be extremely parameter sensitive the best strategy that has been observed is to set the Harmony Consideration rate (HMCR) very high and the Pitch adjustment rate (PAR) low for better results. The value is usually set between $(0.9 - 1)$ for HMCR and $(0.3 - 0.5)$ for PAR.

Chapter 4:

Experiments and Specifications

According to [25], I have designed four types of Linear phase FIR Filters i.e. Lowpass, Highpass, Bandpass and Bandstop filters of order 24 ; two types of filters for order 48 i.e. Lowpass and Bandpass and two types of General phase FIR filters i.e. Lowpass and Bandpass filters of order 24. For the general phase FIR filters of 24 order the number of iterations required were 2000 for lowpass and 2000 for bandpass. For linear phase FIR filters of order 48 the number of iterations required were 5000 and 6000. For linear phase FIR filters of order 24 the number of iterations required were 3000 for lowpass,2000 for bandpass, 3000 for highpass and 2000 for bandstop. Each of the filter designs using Harmony Search algorithm were compared with the state of art of design i.e. Parks McClellan (FIRPM) algorithm. The specifications of the ACER laptop used for execution of the results are AMP Quad Core Processor with TURBO Core Technology upto 3.40 GHz, AMD Redeon R7 Graphics,16GB DDR4 Memory and 1000 GB HDD. In each of the tables given below, $h(n)$ corresponds to the elements of each of the coefficient vector c.

4.1 Results

For Type I, LP-FIR filter of order 24, Filter designs using HS are given below:

4.1.1 Linear phase order 24 FIR filter design obtained Using HS

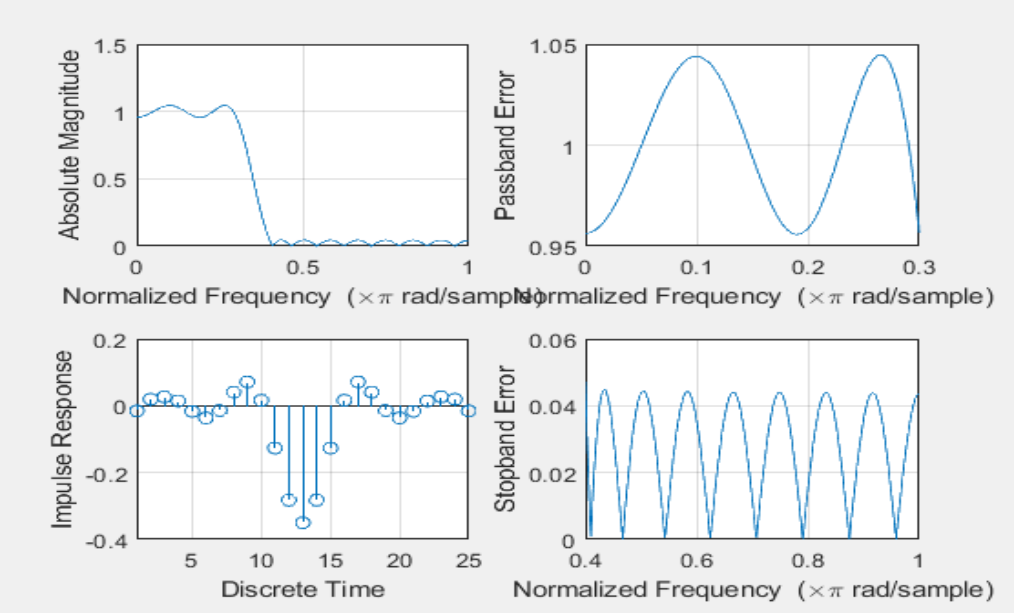

Figure 4.1 Order 24 linear phase lowpass FIR digital filter using HS

Table 4.1 Coefficients of order 24 Type I Lowpass LP-FIR filter by HS

$h(n)$	Coefficients	$h(n)$	Coefficients
h (1) = h (25)	-0.350237552828043	h (8) = h (18)	-0.072649905467191
h (2) = h (24)	-0.565252101711701	h (9) = h (17)	-0.033197972446944
h (3) = h (23)	-0.253185656265041	h (10)=h (16)	0.028746381580727
h (4) = h (22)	0.032547044606794	h (11) =h(15)	0.053433481135664
h (5) = h (21)	0.142361273702732	h (12) =h(14)	0.038835320237712
h (6) = h (20)	0.081416921142883	h (13) =h(13)	-0.030929528348338
h (7) = h (19)	-0.028252610807596		

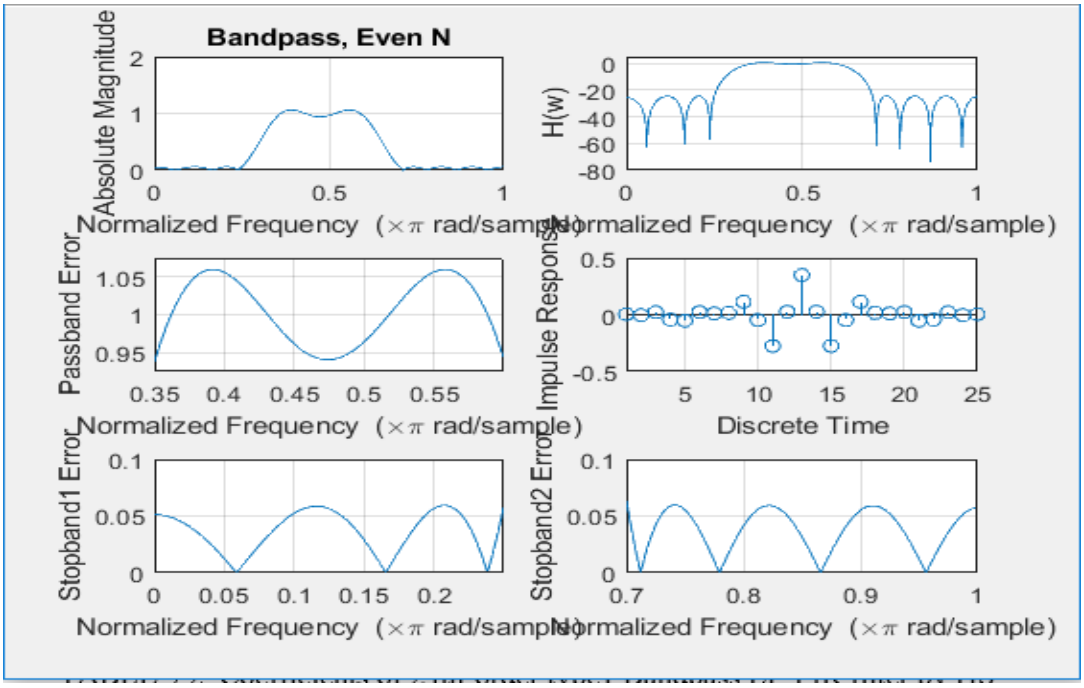

Figure 4.2 Order 24 linear phase bandpass FIR digital filter using HS

Table 4.2 Coefficients of order 24 Type1 Bandpass LP-FIR filter by HS

$h(n)$	Coefficients	$h(n)$	Coefficients
h (1) = h (25)	0.350303134708427	h (8) = h (18)	0.047079368719908
h (2) = h (24)	0.052209310071158	h (9) = h (17)	-0.107236712478247
h (3) = h (23)	-0.551036508540813	h (10)=h (16)	-0.086029699446602
h (4) = h (22)	-0.093899507511783	h (11) =h(15)	0.046593406215097
h (5) = h (21)	0.228883288820410	h (12) =h(14)	-0.005652692935345
h (6) = h (20)	0.032025438939093	h (13) =h(13)	0.011843243684736
h (7) = h (19)	0.023604472736934		

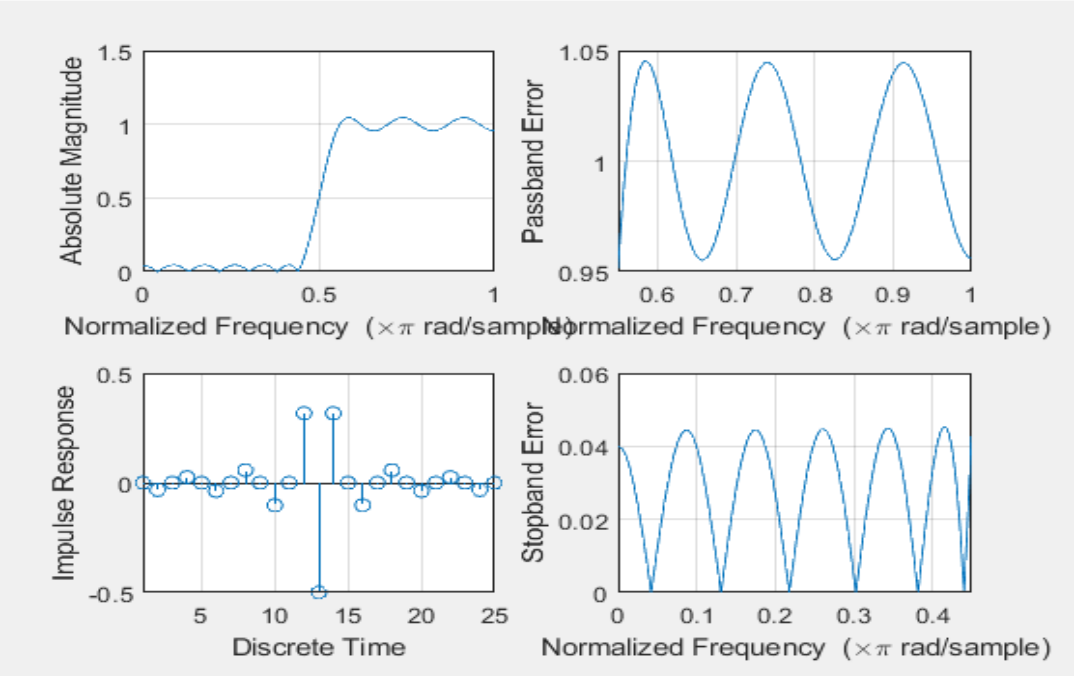

Figure 4.3 Order 24 linear phase highpass FIR digital filter using HS

Table 4.3 Coefficients of order 24 Type1 Highpass LP-FIR filter by HS

$h(n)$	Coefficients	$h(n)$	Coefficients
h (1) = h (25)	-0.499820574150641	h (8) = h (18)	-0.074222182119480
h (2) = h (24)	0.634523380847690	h (9) = h (17)	0.000375203770619
h (3) = h (23)	0.000357974532578	h (10)=h (16)	0.051272559778609
h (4) = h (22)	-0.204376733181727	h (11) =h(15)	0.000304996944570
h (5) = h (21)	0.000369942776821	h (12) =h(14)	-0.064917786714292
h (6) = h (20)	0.115655528658297	h (13) =h(13)	0.000130710993405
h (7) = h (19)	0.000367750795238		

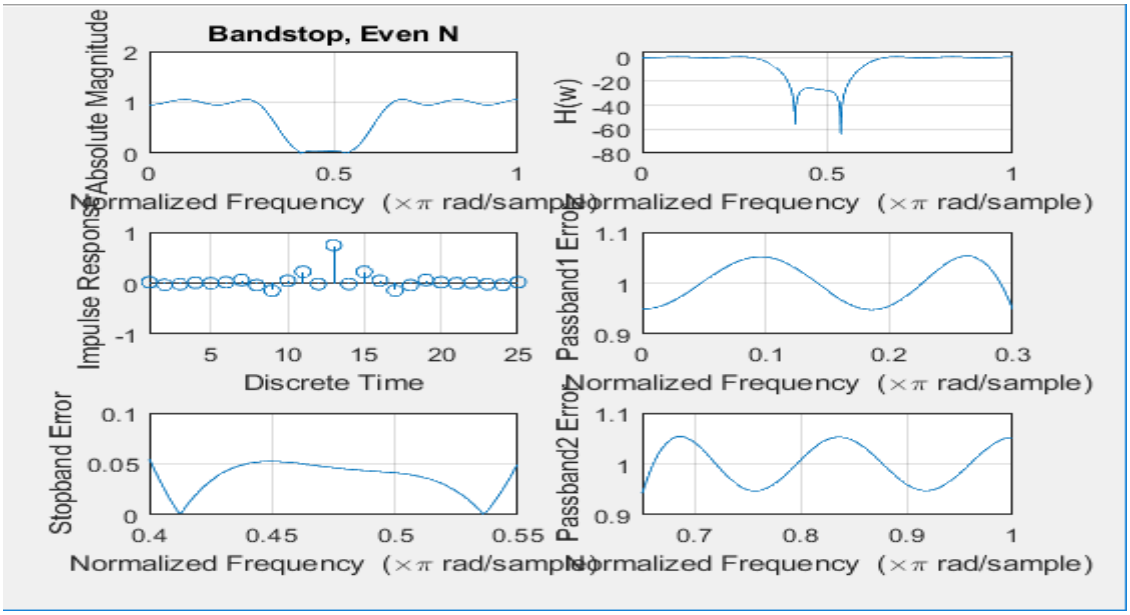

Figure 4.4 Order 24 linear phase bandstop FIR digital filter using HS

Table 4.4 Coefficients of order 24 Type1 Bandstop LP-FIR filter by HS

$h(n)$	Coefficients	$h(n)$	Coefficients
h (1) = h (25)	0.744112519996535	h (8) = h (18)	0.031448299802474
h (2) = h (24)	-0.038178211008451	h (9) = h (17)	-0.006067218708003
h (3) = h (23)	0.447923987568812	h (10)=h (16)	0.014756011526648
h (4) = h (22)	0.088740783386581	h (11) =h(15)	-0.046961029411774
h (5) = h (21)	-0.291105613775601	h (12) =h(14)	-0.066086576947304
h (6) = h (20)	-0.082217719472732	h (13) =h(13)	0.032709290917292
h (7) = h (19)	0.119815317593663		

4.1.2 Linear Phase Results compared with FIRPM for order 24

For Type I order 24, Linear Phase filters designed using HS were compared with the state of art of designs using PM algorithm.

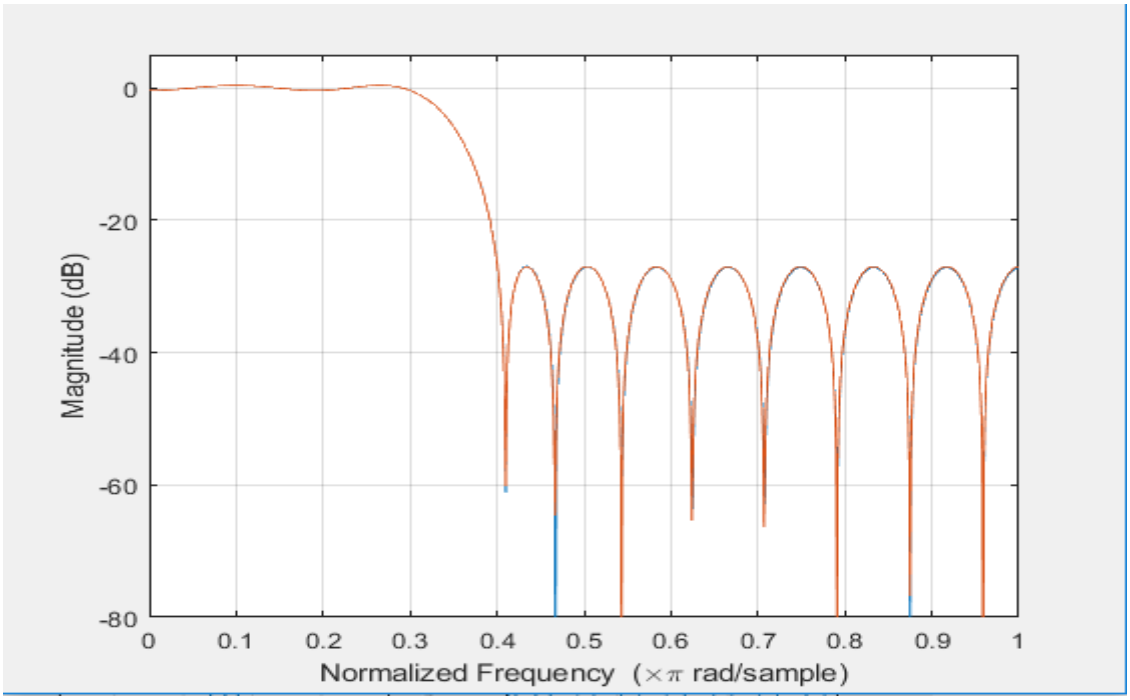

Figure 4.5 Lowpass FIR filter comparing HS and FIRPM

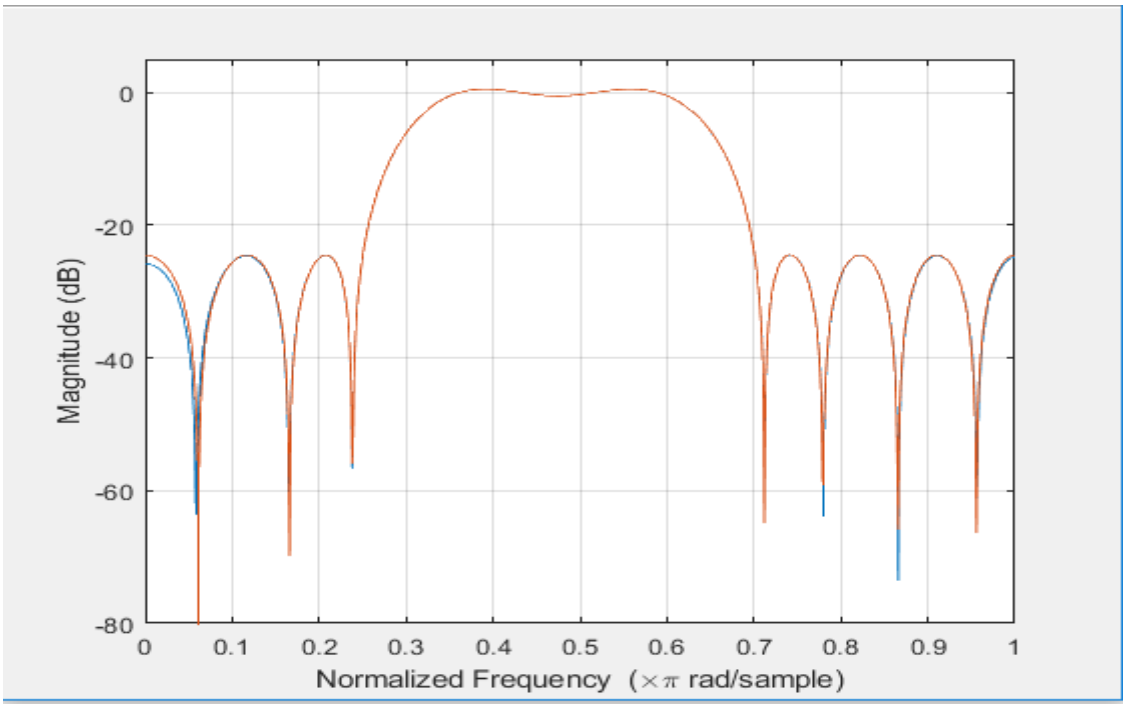

Figure 4.6 Bandpass FIR filter comparing HS and FIRPM

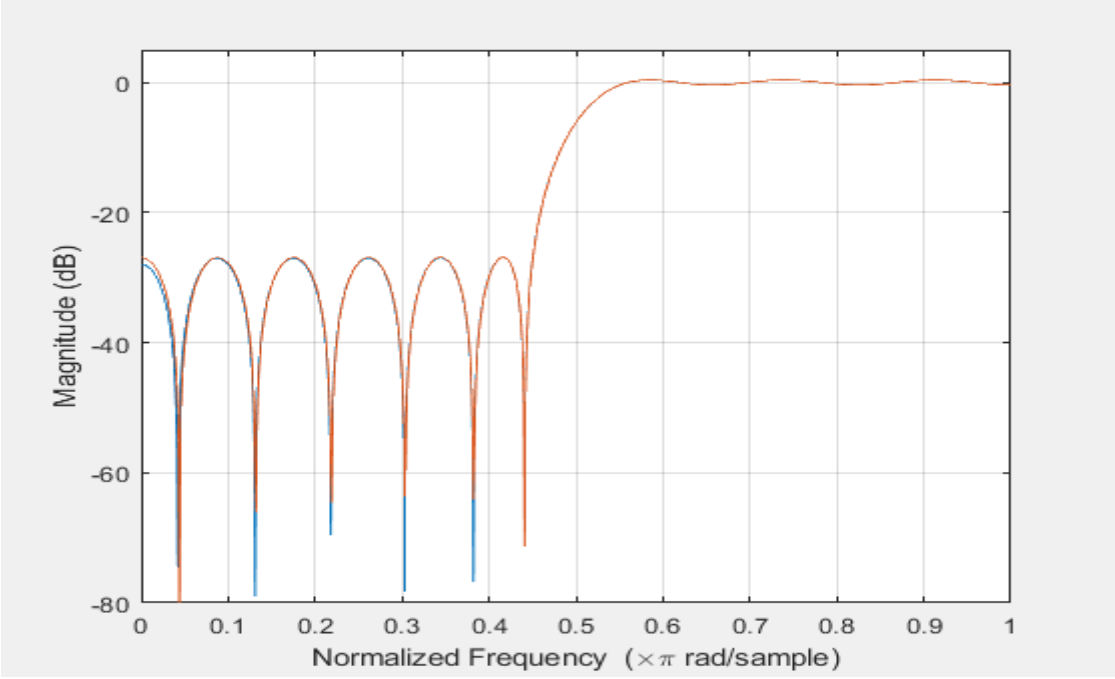

Figure 4.7 Highpass FIR filter comparing HS and FIRPM

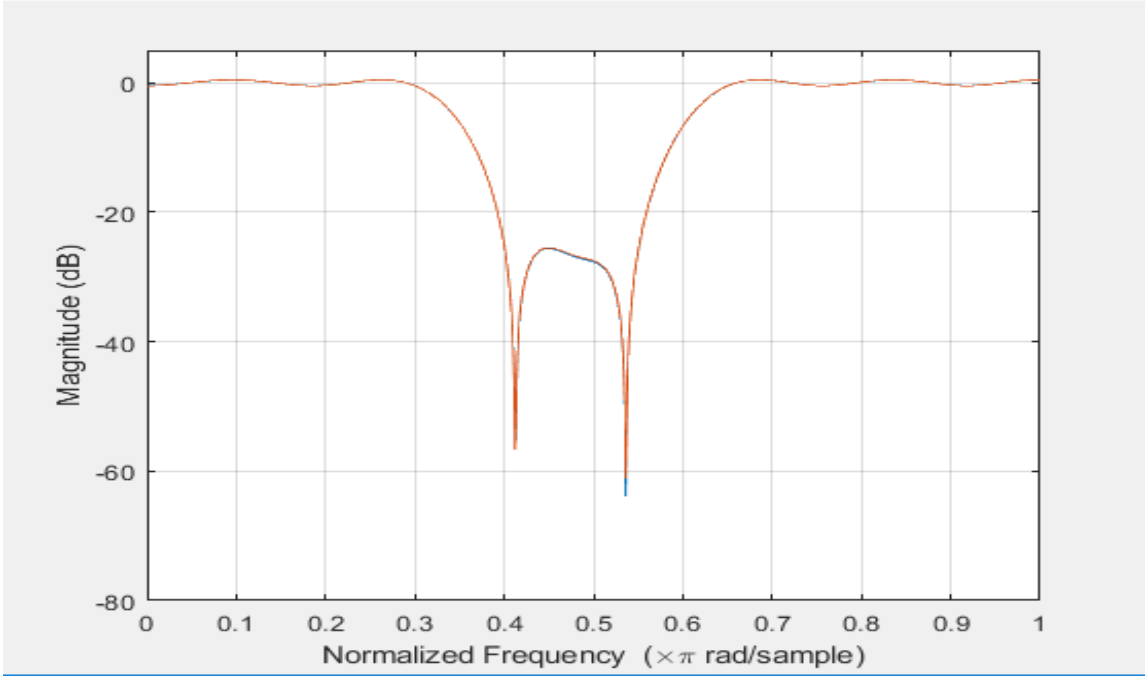

Figure 4.8 Bandstop FIR filter comparing HS and FIRPM

Table 4.5 Order 24 FIR type 1 filter design results comparison (PM: Parks McClellan; HS: Harmony Search)

Filter	**Alg**	**Peak(Stopband1) error**	**Peak(Passband) error**	**Peak(Stopband2) error**	**Time elapsed(sec)**	**Iterations**
Lowpass	HS	0.046628260027534	0.044666427086463	-	77.348587	3000
	PM	0.046804417806560	0.044663349407852	-	0.049528	-
Bandpass	HS	0.059338371436313	0.061796132957827	0.064082926953406	74.295309	2000
	PM	0.059579921275612	0.061630490482057	0.063817460882562	0.182275	
Highpass	HS	0.045290834602924	0.048374884643127	-	129.882131	3000
	PM	0.045273829394664	0.048082904678732	-	0.183603	-
Filter	**Alg**	**Peak(Passband1) error**	**Peak(Stopband) error**	**Peak(Passband2) error**	**Time elapsed(sec)**	**Iterations**
Bandstop	HS	0.053218190592352	0.055317396941249	0.057826084207768	75.851723	2000
	PM	0.053132020244947	0.055140706326915	0.057141822013960	0.190047	

4.1.3 Linear Phase order 48 results obtained Using HS

For Type I, LP-FIR filter of order 48, Filter designs using HS are given below:

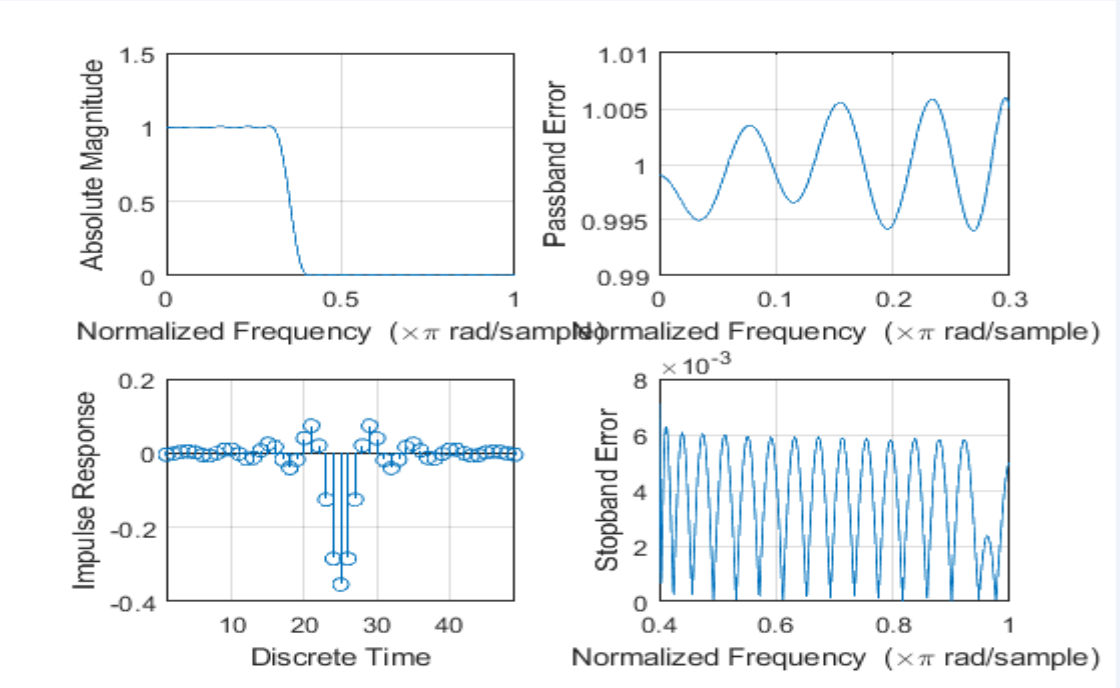

Figure 4.9 Order 48 linear phase lowpass FIR digital filter using HS

Table 4.6 Coefficients of order 48 Type1 Lowpass LP-FIR filter by HS

$h(n)$	Coefficients	$h(n)$	Coefficients
h (1) = h (49)	-0.352251700188307	h (14) =h (36)	-0.030936996191454
h (2) = h (48)	-0.568281184170916	h (15) =h (35)	-0.005923780693295
h (3) = h (47)	-0.252443433102352	h (16) =h (34)	0.018577221563863
h (4) = h (46)	0.036519496366527	h (17) =h (33)	0.018584013068112
h (5) = h (45)	0.146955245209471	h (18) =h (32)	0.000510603408968
h (6) = h (44)	0.081553019599817	h (19) =h (31)	-0.013180094027982
h (7) = h (43)	-0.033958722373520	h (20) =h (30)	-0.010513552330539
h (8) = h (42)	-0.080402902213771	h (21) =h (29)	0.001753422586218
h (9) = h (41)	-0.036880265716685	h (22) =h (28)	0.008607651735627
h (10) = h (40)	0.029613946733669	h (23) =h (27)	0.005963507675716
h (11) = h (39)	0.049671031599446	h (24) =h (26)	-0.001109695183876
h (12) = h (38)	0.016559466462641	h (25)	-0.006997367396814
h (13) = h (37)	-0.024255040884726		

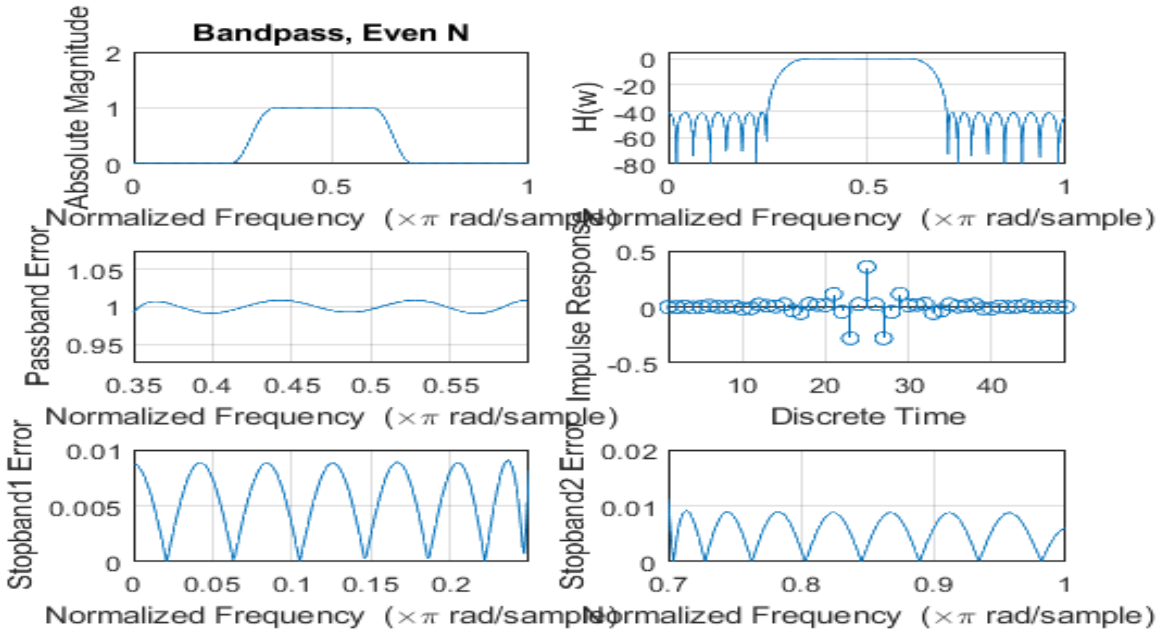

Figure 4.10 Order 48 linear phase bandpass FIR digital filter using HS

Table 4.7 Coefficients of order 48 Type1 Bandpass LP-FIR filter by HS

$h(n)$	Coefficients	$h(n)$	Coefficients
h (1) = h (49)	0.356276607381141	h (14) =h (36)	-0.018822185172538
h (2) = h (48)	0.058406454737794	h (15) =h (35)	-0.037862669876357
h (3) = h (47)	-0.558783198285375	h (16) =h (34)	0.000078752517054
h (4) = h (46)	-0.106140643669705	h (17) =h (33)	-0.002928708021000
h (5) = h (45)	0.224822827668657	h (18) =h (32)	0.002485914158717
h (6) = h (44)	0.035674626523712	h (19) =h (31)	0.022726216510685
h (7) = h (43)	0.036829784372809	h (20) =h (30)	0.003856380313290
h (8) = h (42)	0.063163275710184	h (21) =h (29)	-0.012244614366438
h (9) = h (41)	-0.101527156684745	h (22) =h (28)	-0.004991469559354
h (10) = h (40)	-0.076430796730980	h (23) =h (27)	-0.006681096771392
h (11) = h (39)	0.010845194001327	h (24) =h (26)	0.000634917771595
h (12) = h (38)	0.017763420355727	h (25)	-0.006997367396814
h (13) = h (37)	0.046444592625196		

4.1.4 Linear Phase Results compared with FIRPM for order 48

For Type I order 48, Linear Phase filters designed using HS were compared with the state of art of designs using PM algorithm.

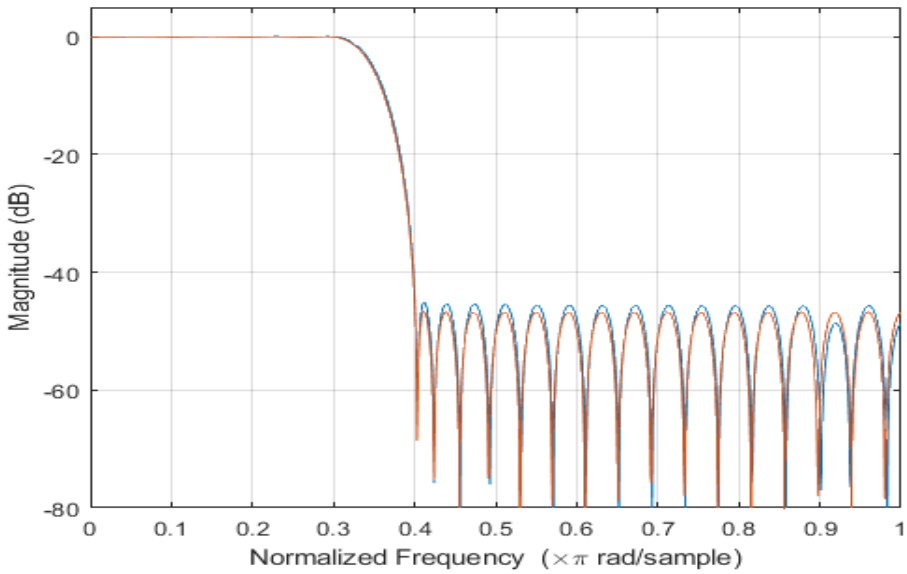

Figure 4.11 Lowpass FIR Filter comparing HS and FIRPM

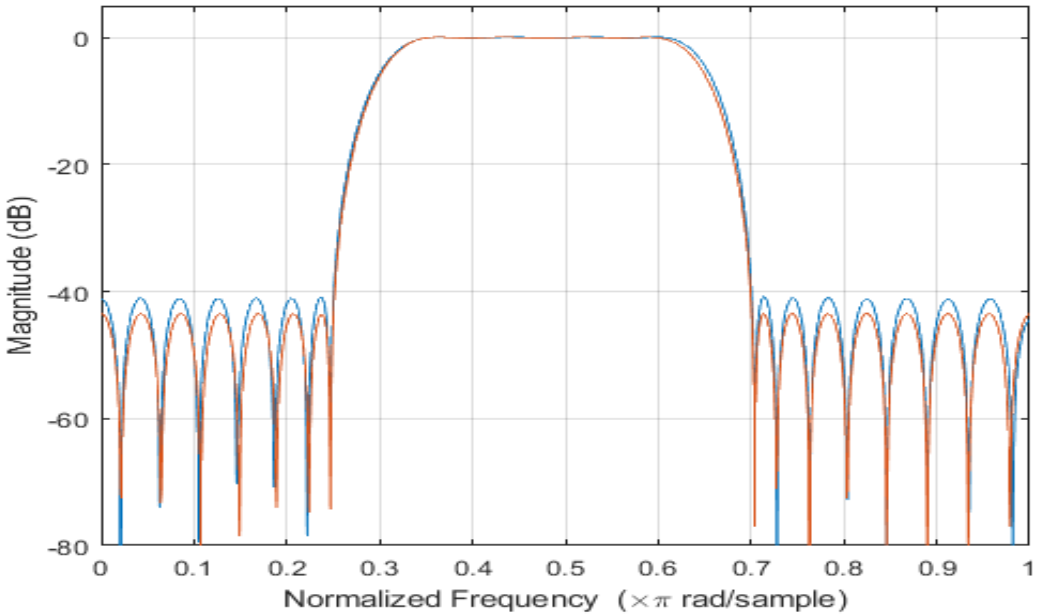

Figure 4.12 Bandpass FIR Filter comparing HS and FIRPM

Table 4.8 Order 48 FIR type 1 filter design results comparison (PM: Parks McClellan; HS: Harmony Search)

Filter	Alg	Peak(Stopband1) error	Peak(Passband) error	Peak(Stopband2) error	Time elapsed(sec)	Iterations
Lowpass	HS	0.006322975019476	0.005518869333202	-	218.325517	5000
	PM	0.004620232061237	0.004630930079829	-	0.417	-
Bandpass	HS	0.009021703930327	0.008856677587438	0.011149104523132	177.263734	6000
	PM	0.006700546942702	0.006700259956931	0.006699594292274	0.134	

Table 4.9 Lowpass, Highpass, Bandpass, and Bandstop digital filter cutoff frequencies

	w_{s1}	w_{p1}	w_{p2}	w_{s2}
LP	-	-	0.30π	0.40π
HP	0.45π	0.55π	-	-
BP	0.25π	0.35π	0.6π	0.7π
BS	0.40π	0.30π	0.65π	0.55π

Table 4.10 Linear Phase FIR Filter Coefficients (Order 24)

Symbol	Description	LP	BP	HP	BS
$c_k^{[u]}$	Upper bound of filter coefficients	0.4	0.4	1	-1
$c_k^{[l]}$	Lower bound of filter coefficients	-0.06	-0.85	-1	1
N	Filter order	24	24	24	24
N_c	Number of distinct filter coefficients	13	13	13	13
τ	Group delay	12	12	12	12
p	Least pth order	128	128	128	128
K	Number of frequency points	1001	1001	1001	1001
K_{s1}	Number of SB1 frequency points	601	251	451	
K_p	Number of PB frequency points	301	251	451	
K_{s2}	Number of SB2 frequency points	-	301	-	
K_{p1}	Number of PB1 frequency points				301
K_s	Number of SB frequency points				151
K_{p2}	Number of PB2 frequency points				351
P	HS population size	13	13	13	13
HMS	Harmony memory Size	20	40	40	40
$HMCR$	Harmony Memory Considering Rate	1	1	1	1
PAR	Pitch Adjusting Rate	1	1	1	1

Table 4.11 Linear Phase FIR Filter Coefficients (Order 48)

Symbol	Description	LP	BP	HP	BS
$c_k^{[u]}$	Upper bound of filter coefficients	0.4	0.4	1	-1
$c_k^{[l]}$	Lower bound of filter coefficients	-0.06	-0.85	-1	1
N	Filter order	48	48	48	48
N_c	Number of distinct filter coefficients	25	25	25	25
τ	Group delay	24	24	24	24
p	Least pth order	128	128	128	128
K	Number of frequency points	1001	1001	1001	1001
K_{s1}	Number of SB1 frequency points	601	251	451	
K_p	Number of PB frequency points	301	251	451	
K_{s2}	Number of SB2 frequency points	-	301	-	
K_{p1}	Number of PB1 frequency points				301
K_s	Number of SB frequency points				151
K_{p2}	Number of PB2 frequency points				351
P	HS population size	25	25	25	25
HMS	Harmony memory Size	20	40	40	40
$HMCR$	Harmony Memory Considering Rate	1	1	1	1
PAR	Pitch Adjusting Rate	1	1	1	1

4.1.5 General FIR Results obtained Using HS for Order 24

The design results for order 24 General FIR are given below:

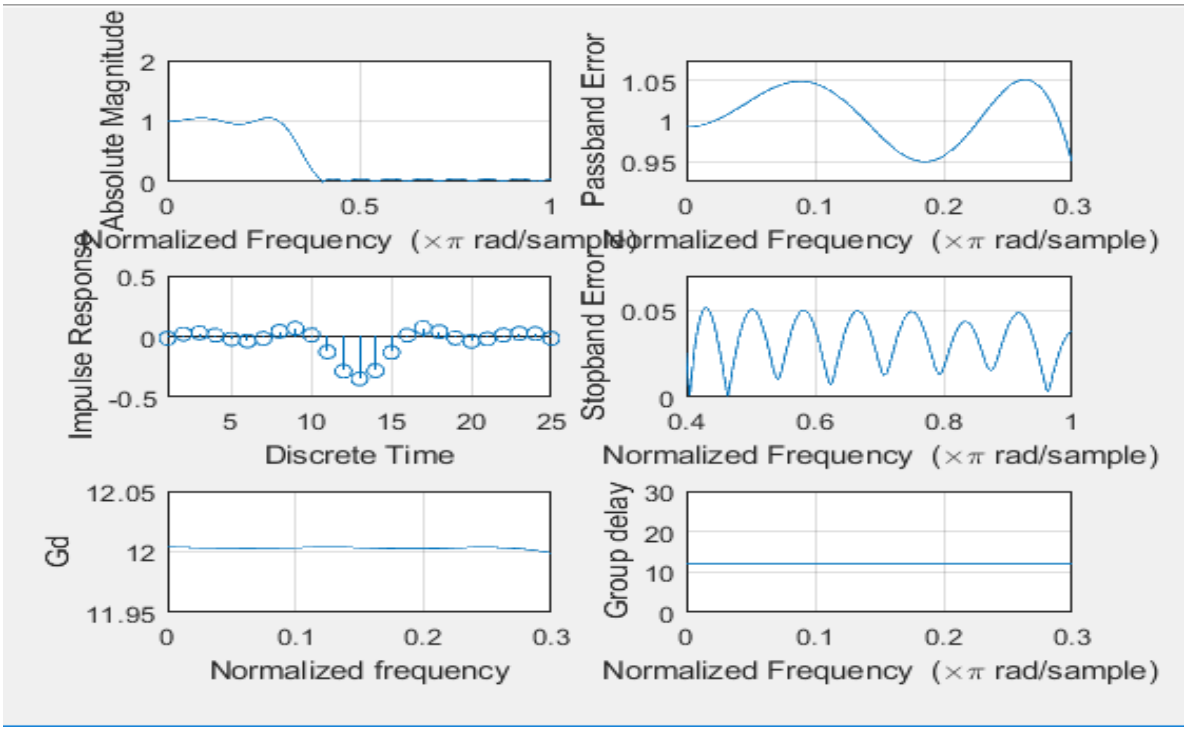

Figure 4.13 Order 24 general phase lowpass FIR digital filter using HS

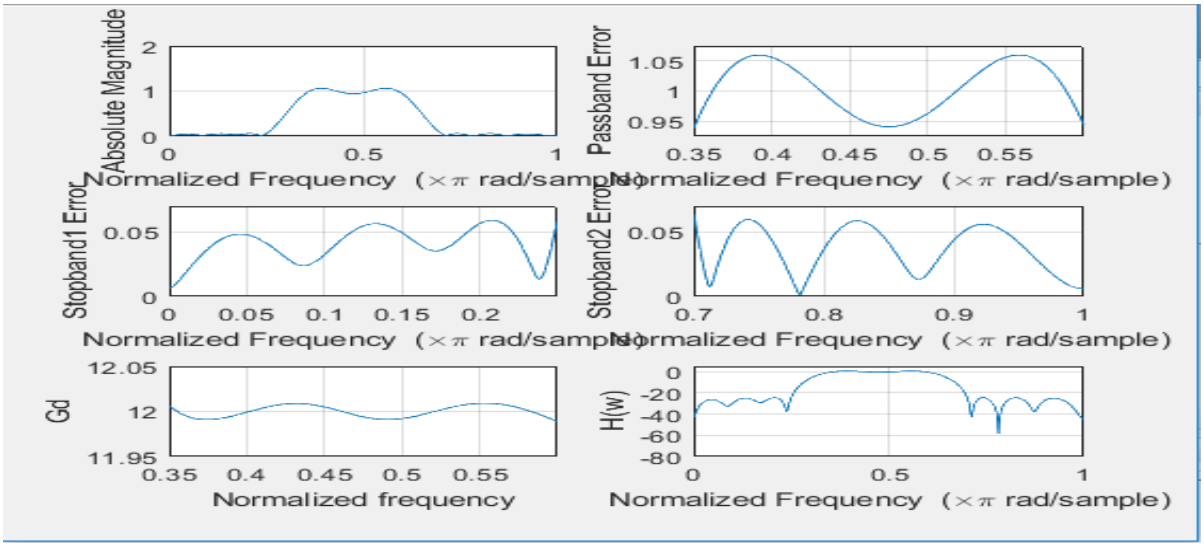

Figure 4.14 Order 24 general phase bandpass FIR digital filter using HS

Table 4.12 Coefficients of order 24 type1 Lowpass LP-GFIR filter by HS

$h(n)$	Coefficients	$h(n)$	Coefficients
h (1)	0.003694355886800	h (14)	-0.349274760606803
h (2)	-0.014763144385305	h (15)	-0.281777065191755
h (3)	0.019138733095744	h (16)	-0.131713463510407
h (4)	0.029696469927572	h (17)	0.010160648724700
h (5)	0.010324290013704	h (18)	0.071265833011746
h (6)	-0.021176484456728	h (19)	0.038805264958585
h (7)	-0.037457015110282	h (20)	-0.012903371150532
h (8)	-0.015822054394316	h (21)	-0.039696912918417
h (9)	0.042017179968997	h (22)	-0.020267304233717
h (10)	0.067378097042448	h (23)	0.012058478029507
h (11)	0.012260570149869	h (24)	0.025281495999975
h (12)	-0.126984670588759	h (25)	0.022924670774612
h (13)	-0.286188318756668		

Table 4.13 Coefficients of order 24 type1 Bandpass LP-GFIR filter by HS

$h(n)$	Coefficients	$h(n)$	Coefficients
h (1)	0.008955930739267	h (14)	0.214694042527110
h (2)	0.001114354403548	h (15)	-0.197727540988785
h (3)	0.034974459821717	h (16)	-0.153166617948601
h (4)	-0.000092077325015	h (17)	0.074427505357468
h (5)	-0.064014282529663	h (18)	0.040258614507933
h (6)	0.003260108992233	h (19)	0.008805782460207
h (7)	0.011201551182595	h (20)	0.047473011653316
h (8)	-0.005278327714543	h (21)	-0.022077632319768
h (9)	0.111053586481819	h (22)	-0.060993629166203
h (10)	0.079942705918041	h (23)	0.002321752182706
h (11)	-0.250953318560656	h (24)	-0.003273050473436
h (12)	-0.170263059050198	h (25)	-0.000600764453942
h (13)	0.283193531374874		

Table 4.14 Order 24 General FIR Type 1 filter design results using HS

Filter	**Alg**	**Peak(Stopband) error**	**Peak(Passband) error**	**Group delay error**	**Peak(Stopband2)**
Lowpass	HS	0.051502483774843	0.051227181069569	0.003694355886800	-
Lowpass	Paper	0.051242944505630	0.050130894010477	0.038164522218185	
Filter	**Alg**	**Peak(Passband) error**	**Peak(Stopband1) error**	**Peak(Stopband2) error**	**Group delay**
Bandpass	HS	0.062123033806461	0.059023845560883	0.064299164927664	0.010449261319422
Bandpass	Paper	0.060228252610734	0.060148302523172	0.059976240658255	0.043592307120207

4.1.6 Design using XOR neural network

Table 4.15 2-input one output Neural network design parameters

Parameters	Obtained Values
W1	5.000000000000000
W2	5.000000000000000
W3	-5.000000000000000
W4	-4.999712367554367
W5	5.000000000000000
W6	-4.999122246789436
B1	2.676347853329807
B2	-2.676165488769043
B3	-2.498565477908653

Table 4.16 2-input one output Neural network design

Iterations	Results	Outputs	Error Values
2000	Desired	0	1.0e-04 * 0.540526889129954
	Obtained	0.007352053380723	
2000	Desired	1	1.0e-04 * 0.540554301567679
	Obtained	0.992647760194555	
2000	Desired	1	1.0e-04 * 0.540543852259729
	Obtained	0.992647831256971	
2000	Desired	0	1.0e-04 * 0.540527266975788
	Obtained	0.007352055950384	

To verify the results obtained produce a XOR output, an input grid of 100 by 100 is selected of the two inputs X1 and X2 and the output results are plotted at every instant of input. The result is shown in Figure 5.6.3.

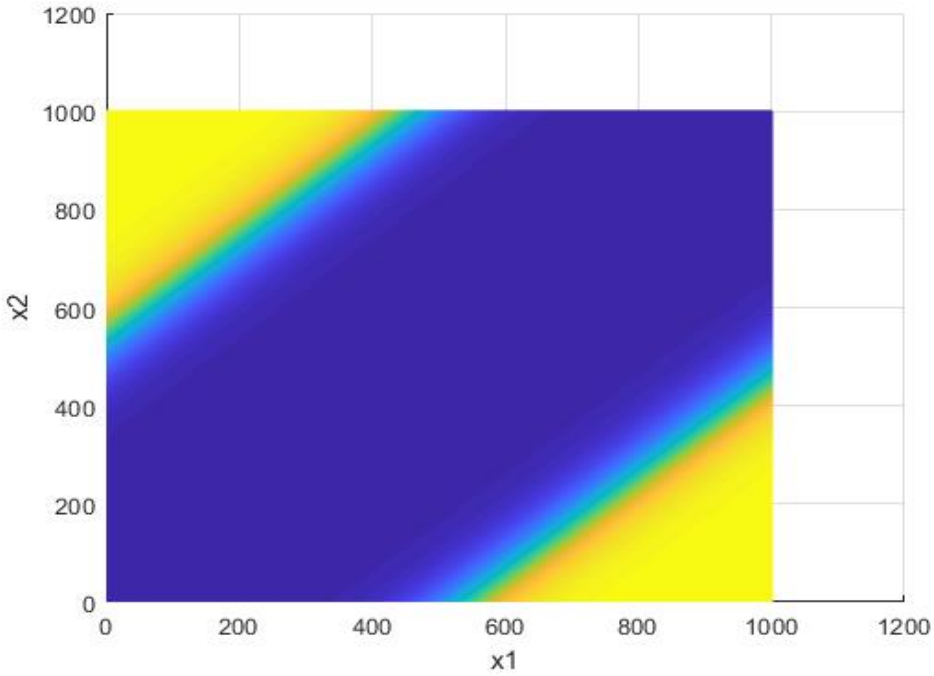

Figure 4.15 Two dimensional view of XOR neural network

4.1.7 Digit recognition using feedforward neural network

Table 4.17 Computational results of the feedforward neural network design for ten hidden neurons (without noise)

Number Of Iterations`	Time elapsed(sec)	Best Cost Value
4000	694.592215	3.0309e-15

Table 4.18 Comparison of results with the ideal output and the Mean Square errors for each of the four output neurons

Digits	**Results**	**4- Output Neurons (Number of hidden neurons=10)**				**Mean Square errors**			
Zero	Desired	**-1**	**1**	**-1**	**1**	0	0	0	0
	Obtained	-1	1	-1	1	0	0	0	0
One	Desired	**1**	**-1**	**-1**	**-1**	0	0	0	0
	Obtained	1	-0.99	-0.99	-1	0	0	0	0
Two	Desired	**-1**	**1**	**-1**	**-1**	0	0	0	0
	Obtained	-1	1	-1	-1	0	0	0	0
Three	Desired	**1**	**1**	**-1**	**-1**	0	0	0	0
	Obtained	1	1	-1	-1	0	0	0	0
Four	Desired	**-1**	**-1**	**1**	**-1**	0	0	0	0
	Obtained	-1	-1	1	-1	0	0	0	0
Five	Desired	**1**	**-1**	**1**	**-1**	0	0	0	0
	Obtained	1	-0.99	1	-1	0	0	0	0
Six	Desired	**-1**	**1**	**1**	**-1**	0	0	0	0
	Obtained	-0.99	1	1	-1	0	0	0	0
Seven	Desired	**1**	**1**	**1**	**-1**	0	0	0	0
	Obtained	1	1	1	-1	0	0	0	0
Eight	Desired	**-1**	**-1**	**-1**	**1**	0	0	0	0
	Obtained	-1	-1	-1	1	0	0	0	0
Nine	Desired	**1**	**-1**	**-1**	**1**	0	0	0	0
	Obtained	1	-1	-1	0.99	0	0	0	0

Table 4.19 Computational results of the feedforward neural network design for ten hidden neurons (40% noise)

Number of Iterations`	Time elapsed(sec)	Best Cost Value
2000	316.970887	3.2738e-10

Table 4.20 Comparison of results with the ideal output and the Mean Square errors for each of the four output neurons

Digits	**Results**	**4- Output Neurons (Number of hidden neurons=10)**				**Mean Square errors**			
Zero	Desired	**-1**	**1**	**-1**	**1**	0	0	0	0
	Obtained	-1	1	-1	1	0	0	0	0
One	Desired	**1**	**-1**	**-1**	**-1**	0	0	0	0
	Obtained	1	-0.99	-0.99	-1	0	0	0	0
Two	Desired	**-1**	**1**	**-1**	**-1**	0	0	0	0
	Obtained	-1	1	-1	-1	0	0	0	0
Three	Desired	**1**	**1**	**-1**	**-1**	0	0	0	0
	Obtained	1	1	-1	-1	0	0	0	0
Four	Desired	**-1**	**-1**	**1**	**-1**	0	0	0	0
	Obtained	-1	-1	1	-1	0	0	0	0
Five	Desired	**1**	**-1**	**1**	**-1**	0	0	0	0
	Obtained	1	-0.99	1	-1	0	0	0	0
Six	Desired	**-1**	**1**	**1**	**-1**	0	0	0	0
	Obtained	-0.83	1	1	-1	0	0	0	0
Seven	Desired	**1**	**1**	**1**	**-1**	0	0	0	0
	Obtained	1	1	1	-1	0	0	0	0
Eight	Desired	**-1**	**-1**	**-1**	**1**	0	0	0	0
	Obtained	-1	-1	-1	1	0	0	0	0
Nine	Desired	**1**	**-1**	**-1**	**1**	0	0	0	0
	Obtained	1	-1	-1	0.82	0	0	0	0

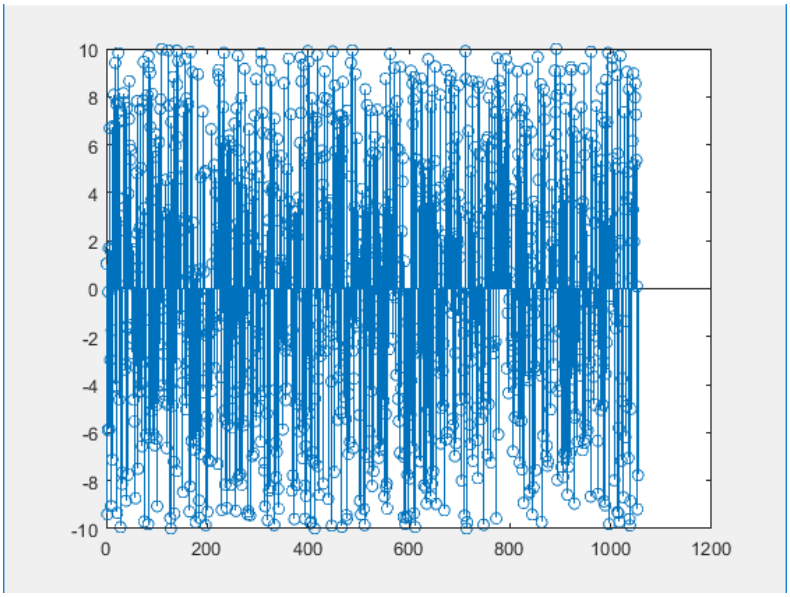

Figure 4.16 Plot of the 1054 weight vectors for ten hidden neurons

Table 4.21 Computational results of the feedforward neural network design for eight hidden neurons (without noise)

Number Of Iterations`	Time elapsed(sec)	Best Cost Value
4000	521.707630	4.4439e-06

Table 4.22 Computational results of the feedforward neural network design for eight hidden neurons (without noise)

Digits	**Results**	**4- Output Neurons (Number of hidden neurons=8)**				**Mean Square errors**			
Zero	Desired	**-1**	**1**	**-1**	**1**				
	Obtained	-0.99	0.99	-0.99	1	0	0	0	0
One	Desired	**1**	**-1**	**-1**	**-1**				
	Obtained	0.99	-1	-0.99	-0.99	0	0	0	0
Two	Desired	**-1**	**1**	**-1**	**-1**				
	Obtained	-0.99	1	-0.99	-0.99	0	0	0	0
Three	Desired	**1**	**1**	**-1**	**-1**				
	Obtained	0.99	1	-1	-0.99	0	0	0	0
Four	Desired	**-1**	**-1**	**1**	**-1**				
	Obtained	-1	-1	0.99	-0.99	0	0	0	0
Five	Desired	**1**	**-1**	**1**	**-1**				
	Obtained	0.99	-0.99	0.99	-0.99	0	0	0	0
Six	Desired	**-1**	**1**	**1**	**-1**				
	Obtained	-0.99	0.99	0.99	-1	0	0	0	0
Seven	Desired	**1**	**1**	**1**	**-1**				
	Obtained	0.99	1	0.99	-0.99	0	0	0	0
Eight	Desired	**-1**	**-1**	**-1**	**1**				
	Obtained	-1	-0.99	-0.99	0.99	0	0	0	0
Nine	Desired	**1**	**-1**	**-1**	**1**				
	Obtained	0.99	-0.99	-0.99	0.99	0	0	0	0

Table 4.23 Computational results of the feedforward neural network design for eight hidden neurons (40% noise)

Number Of Iterations`	Time elapsed(sec)	Best Cost Value
2000	285.544329	1.2818e-07

Table 4.24 Computational results of the feedforward neural network design for eight hidden neurons (40% noise)

Digits	**Results**	**4- Output Neurons (Number of hidden neurons=8)**				**Mean Square errors**			
Zero	Desired	**-1**	**1**	**-1**	**1**				
	Obtained	-0.99	0.65	-0.99	1	0	0	0	0
One	Desired	**1**	**-1**	**-1**	**-1**				
	Obtained	0.99	-1	-0.55	-0.99	0	0	0	0
Two	Desired	**-1**	**1**	**-1**	**-1**				
	Obtained	-0.99	1	-0.99	-0.99	0	0	0	0
Three	Desired	**1**	**1**	**-1**	**-1**				
	Obtained	0.99	1	-1	-0.99	0	0	0	0
Four	Desired	**-1**	**-1**	**1**	**-1**				
	Obtained	-1	-1	0.79	-0.99	0	0	0	0
Five	Desired	**1**	**-1**	**1**	**-1**				
	Obtained	0.99	-0.99	0.99	-0.99	0	0	0	0
Six	Desired	**-1**	**1**	**1**	**-1**				
	Obtained	-0.99	0.99	0.99	-1	0	0	0	0
Seven	Desired	**1**	**1**	**1**	**-1**				
	Obtained	0.99	1	0.77	-0.99	0	0	0	0
Eight	Desired	**-1**	**-1**	**-1**	**1**				
	Obtained	-1	-0.99	-0.99	0.99	0	0	0	0
Nine	Desired	**1**	**-1**	**-1**	**1**				
	Obtained	0.99	-0.99	-0.99	0.99	0	0	0	0

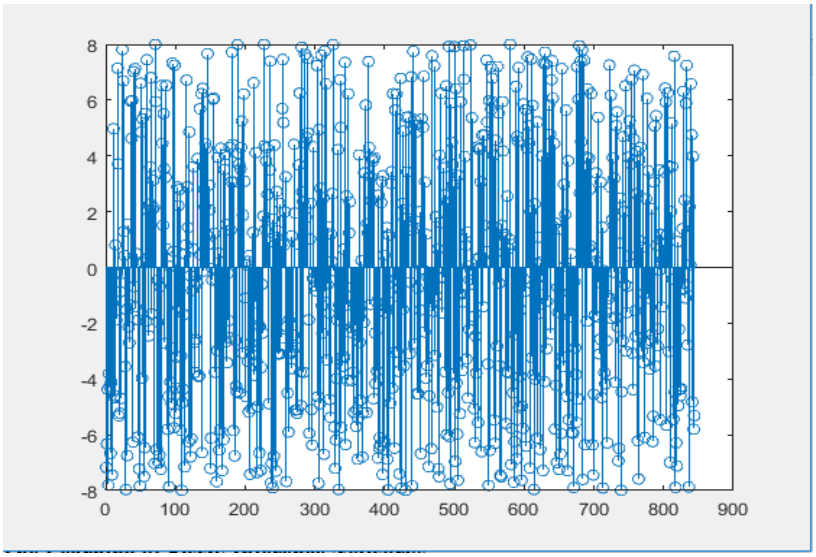

Figure 4.17 Plot of the 844 weight vectors for eight hidden neurons

4.1.8 XOR design using min-sum fuzzy inference network

Table 4.25 Computational results of the min sum fuzzy neural network design (without noise)

No of Iterations	Number of Training data sets	Number of hidden neurons	Time Elapsed(sec)	Mean Square error	Error rate (in %)
3000	9	8	48.782043	3.650675033056877e-08	7.922046973973806e-06

Table 4.26 Comparison of results of the min-sum fuzzy inference network

Number of training sets	Outputs	Results		Obtained Error values corresponding to the obtained output (1.0e-06 *)
9	Desired	0	1	0.004863132444635
	Obtained	0	1	0.024103069540260
	Desired	0	0	0
	Obtained	0	0	0
	Desired	1	0	0.001852336950492
	Obtained	1	0	0.053895729460855
	Desired	0	0	0
	Obtained	0	0	0
	Desired	0	0	0
	Obtained	0	0	0
	Desired	0	0	0
	Obtained	0	0	0
	Desired	1	0	0.042581973502820
	Obtained	1	0	0.112744241219114
	Desired	0	0	0
	Obtained	0	0	0
	Desired	0	1	0.009299800117205
	Obtained	0	1	0.079220469739738

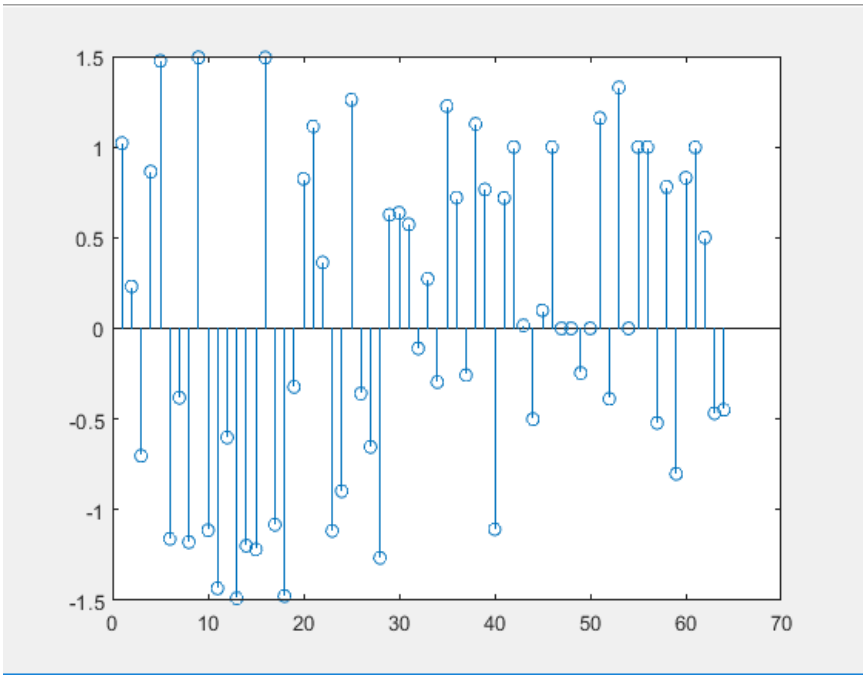

Figure 4.18 Plot of the 64 best solution vectors for eight hidden neurons

Chapter 5:

Conclusions

The harmony search algorithm has been used to design Type 1 linear phase FIR digital filter coefficients and the complex neural network parameters. The results have been compared with the state of art methods. The HS algorithm focusses on the optimum selection of control parameters values and formulations which requires more mathematical and logical requirements to be able to modify the algorithm of this optimization method. HS has proven to be a suitable alternative since it gave results almost near to PM in almost all designs. The HS algorithm generates new vector after considering all the existing vectors whereas the genetic algorithm (GA) only considers the two parent vectors. This increases the flexibility of HS algorithm and produces better solutions.

HS imposes fewer mathematical requirements and does not require initial value settings of the decision variables. There are few important parameters HMCR, HMS, PAR, and bw, but PAR and bw are very important parameters in fine tuning of optimal solution vectors. The process of searching for the best harmony can be considered as analogous to finding a solution to the optimization problem [8] since both the processes are intended to produce the best or the optimal result under the given conditions.

The value of HMCR in the basic HS algorithm is fixed at its initialization and does not change until the search ends. As the value of HMCR decreases, harmony memory is used less efficiently, the algorithm converges much more slowly, and more time is consumed. As the value of HMCR increases, harmony memory is used more efficiently, the method converges more rapidly, less time is consumed, and the HS is easily trapped in a local optimum. To enhance the power of the HS algorithm, HMCR is usually valued in the interval [0.9,1]. The meta-heuristic algorithm handles intensification and diversification. Its intensification and diversification are represented by HMCR and PAR, respectively, and its superiority is confirmed by the existence of a large number of successful applications to diverse scientific and engineering problems.

The power and efficiency of the HS algorithm seem obvious after comparison with other metaheuristics; however, there are some unanswered questions concerning the whole class of HS algorithms. At the moment, the HS algorithm like almost all other metaheuristics is a higher-level optimization strategy which works well under appropriate conditions, but sometimes it is

not understandable to us that how they work so well. For example, when choosing the harmony accepting rate, we usually use a higher value, say, 0.7 to 0.95. This is obtained by experimenting with the simulations, or using a similar inspiration from genetic algorithms when the mutation rate should be low, and thus the accepting rate of the existing gene components are high. However, it is very difficult to say what range of values and which combinations are surely better than others. In general, there lacks a theoretical framework for metaheuristics to provide some analytical guidance to the following important issues: How to improve the efficiency for a given problem and what conditions are required for a good rate of convergence? Also, how to prove the global optima for a given metaheuristic problem is a concern. These are still open questions that need further research. The encouraging thing is that many researchers are interested in tackling these difficult challenges, and important progress has been made concerning the convergence of algorithms such as simulated annealing. Any progress concerning the convergence of HS and other algorithms would be influentially profound.

Even without a solid framework, the scientists are encouraged to develop more hybrid algorithms. In fact, the algorithm development itself is a metaheuristic process similar to the manner to the key components of HS algorithms: to use the existing successful algorithms; to develop slightly different variants based on the existing algorithms, and to formulate heuristically completely new metaheuristic algorithms. By using the existing algorithms, the right algorithms can be found out for the right problem. Often, we have to change and reformulate the problem slightly or to improve the algorithms slightly so as to find the solutions more efficiently. Sometimes, we have to develop new algorithms from scratch to solve some tough optimization problems. There are many ways to develop new algorithms, and from the metaheuristic point of view, the most heuristic way is probably to develop new algorithms by hybridization. That is to say, new algorithms are often based on the right combination of the existing metaheuristic algorithms. For example, combining a trajectory type simulated annealing with multiple agents, the parallel simulated annealing can be developed. In the context of HS algorithms, the combination of HS with PSO, the global-best harmony search has been developed [23]. As in the case of any efficient metaheuristic algorithms, the most difficult thing is probably to find the right or optimal balance between diversity and intensity of the found solutions; here the most challenging task in developing new hybrid algorithms is probably to find the right combination of which feature/components of existing algorithms.

The design examples of the digital filters show that the results in most of the cases are either equal or very close to that of FIRPM. Also 40% input noise is introduced to the neural network

design problem, the obtained output in most of the cases i equal to the desired output except for one or two in which the output is not accurate due to noise. All over, as per observation, HS has proven to produce quality results with a satisfactory performance.

REFERENCES

[1] M. Sayadi, R. Ramezanian, and N. Ghaffari-Nasab, “A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems,” Int. J. of Industrial Engineering Computations, 1, 1–10, 2010.

[2] Z. W. Geem (ed.), Music-Inspired Harmony Search Algorithm, (Springer, Berlin, 2001).

[3] D. Manjarresa, I. Landa-Torresa, S. Gil-Lopeza, et al., “A survey on applications of the harmony search algorithm,” Eng. Appl. Artif. Intel. 26(8), 1818–1831, 2013.

[4] M. K. Saka, Optimum design of steel skeleton structures, in music inspired harmony search algorithm, ed. by Z.W. Geem (Springer, Berlin, 2009), pp. 87–112.

[5] R. Mahmoud, M. Maheri, and M. Narimani, “An enhanced harmony search algorithm for optimum design of side sway steel frames,” Comput. Struct. 136, 78–89, 2014.

[6] G. Bekdas and S. M. Nigdeli, “Estimating optimum parameters of tuned mass dampers using harmony search,” Eng. Struct. 33(9), 2716–2723, 2011.

[7] M. Fesanghary, E. Damangir, and I. Soleimani, “Design optimization of shell and tube heat exchangers using global sensitivity analysis and harmony search algorithm,” Appl. Therm. Eng. 29(5–6), 1026–1031, 2009.

[8] B. Jeddi and V. Vahidinasab, “A modified harmony search method for environmental/economic load dispatch of real-world power systems,” Energy Convers. Manag. 78, 661–675, 2014.

[9] N. Sinsuphan, U. Leeton, and T. Kulworawanichpong, “Optimal power flow solution using improved harmony search method,” Appl. Soft Comput. 13(5), 2364–2374, 2013.

[10] R. Arul, G. Ravi, and S. Velusami, “Chaotic self-adaptive differential harmony search algorithm based dynamic economic dispatch,” Int. J. Electr. Power Energy Syst. 50, 85–96, 2013.

[11] J. Li and H. Duan, “Novel biological visual attention mechanism via Gaussian harmony search,” Optik-Int. J. Light Electron Opt. 125(10), 2313–2319, 2014.

[12] J. Fourie, S. Mills, and R. Green, “Harmony filter: a robust visual tracking system using the improved harmony search algorithm,” Image Vis. Comput. 28(12), 1702–1716, 2010.

[13] R. E. Barlow and F. Proschan, Mathematical Theory of Reliability, John Willy & Sons, Inc., New York, 2000.

[14] Sudhansu S. Maithi, Sudhir Murmu, and G. Chattopadhyay, “Estimation of reliability in the two-parameter geometric distribution,” arXiv:1501.05072v1 [stat.AP, 21 Jan 2015.

[15] M. Yakub and A. H. Khan, “Geometric failure law in life testing,” Pure and Applied Mathematika Science 14, pp. 69-76, 2001.

[16] S. K. Bhattacharya and S. Kumar, “Discrete life testing,” IAPQR Transactions 13, pp. 71-76.16, 2004.

[17] H. Krishna and N. Jain, “Classical and Bayes Estimation of Reliability Characteristics of some Basic System Configurations with Geometric Lifetimes of Components,” IAPQR Transactions 27, pp. 35-49, 2002.

[18] H. K. Kwan, Intelligent Computing and System Design, Edition 1.3, dfisp.org, 28 October 2017.

[19] V. Ravi, N. J. Chauhan, and N. Raj Kiran, “Software reliability prediction using intelligent techniques: Application to operational risk prediction in firms,” International Journal of Computational Intelligence and Applications, vol.8, No. 2, pp. 181-194, 2009.

[20] H. K. Kwan and Y. Cai, A fuzzy neural network and its application to pattern recognition, IEEE Transactions on Fuzzy Systems, Volume 2, Number 3, pages 185-193, August 1994.

[21] L. Y. Cai and H. K. Kwan, “Fuzzy classification using fuzzy inference networks,” IEEE Transaction on systems, Man, Cybernetics, Part B: Cybernetics, Volume 28, Number 3, pages 334-347, June 1998.

[22] H. K. Kwan and C. Z. Tang, “Designing multilayer feedforward neural networks using simplified sigmoid activation functions,” Electronics Letters, Volume 28, pages 2343-2345, 3 December 1992.

[23] M. G. H. Omran and M. Mahdavi, Global-best harmony search. Applied Math. Computation, 198:643-656, 2008.

[24] H. K. Kwan, “Asymmetric FIR filter design using evolutionary optimization,” in Proceedings of IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE 2017), Windsor, Ontario, Canada, 30 April-3 May 2017, 4 pages.

[25] H. K. Kwan, Global Optimization Algorithms and Design Applications, Edition 1.2, dfisp.org, 13 March 2018.

[26] Ulker Ezgi Deniz and Ali Haydar, “Comparison of the performances of Differential evolution”, Particle Swarm Optimization and Harmony Search Algorithms on Benchmark Functions, Volume 3, Number 2, September 2012, 8 pages.

VITA AUCTORIS

NAME:	Abira Paul
PLACE OF BIRTH:	Windsor, ON
YEAR OF BIRTH:	1991
EDUCATION:	West Bengal University of Technology, B.Tech in Electronics and Communication Engineering, Kolkata, India, 2014 University of Windsor, MAS Electrical, Windsor, ON, 2018