2017

Designs of Digital Filters and Neural Networks using Firefly Algorithm

Ghazanfer Ali
University of Windsor

Designs of Digital Filters and Neural Networks using Firefly Algorithm

By

Ghazanfer Ali

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

2017

Designs of Digital Filters and Neural Networks using Firefly Algorithm

by

Ghazanfer Ali

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

November 21, 2017

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

Firefly algorithm is an evolutionary algorithm that can be used to solve complex multi-parameter problems in less time. The algorithm was applied to design digital filters of different orders as well as to determine the parameters of complex neural network designs. Digital filters have several applications in the fields of control systems, aerospace, telecommunication, medical equipment and applications, digital appliances, audio recognition processes etc. An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, processes information and can be simulated using a computer to perform certain specific tasks like clustering, classification, and pattern recognition etc. The results of the designs using Firefly algorithm was compared to the state of the art algorithms and found that the digital filter designs produce results close to the Parks McClellan method which shows the algorithm's capability of handling complex problems. Also, for the neural network designs, Firefly algorithm was able to efficiently optimize a number of parameter values. The performance of the algorithm was tested by introducing various input noise levels to the training inputs of the neural network designs and it produced the desired output with negligible error in a time-efficient manner. Overall, Firefly algorithm was found to be competitive in solving the complex design optimization problems like other popular optimization algorithms such as Differential Evolution, Particle Swarm Optimization and Genetic Algorithm. It provides a number of adjustable parameters which can be tuned according to the specified problem so that it can be applied to a number of optimization problems and is capable of producing quality results in a reasonable amount of time.

ACKNOWLEDGEMENTS

I would like to thank Prof. H. K. Kwan for introducing me the subject of digital filter and neural network design, evolutionary algorithms and in particular the firefly algorithm for optimization; and for his guidance and support throughout my M.App.Sc. research. The design problems included in this thesis are suggested by Dr. Kwan which include linear phase FIR digital filters, general FIR digital filters, and multi-layer neural networks. I would also like to thank Dr. H. Wu and Dr. F. Baki for their valuable feedbacks and suggestions. And also, my fellow students, Rija Raju, Miao Zhang and Jiajun Liang for their valuable discussions during my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS/SYMBOLS

FA	Firefly Algorithm
PM	Parks-McClellan Method
DE	Differential Evolution
$\boldsymbol{D(\omega)}$	Desired frequency response
$\boldsymbol{\delta}$	Passband ripple
$\boldsymbol{\delta_p}$	Specified passband ripple
$\boldsymbol{\delta_s}$	Specified stopband ripple
$\boldsymbol{E(\omega)}$	Weighted error function
FIR	Finite Impulse Response
g	Passband gain
GA	Genetic Algorithm
$\boldsymbol{h_k}$	k^{th} filter coefficient
$\boldsymbol{H(\omega)}$	Frequency response of the filter
IIR	Infinite Impulse Response
LP	Linear Program
N	Filter length
NP	Non-deterministic Polynomial Time
Obj	Objective function value
P_{max}	Maximum absolute value of frequency response in passband
P_{min}	Minimum absolute value of frequency response in passband

PSO	Particle Swarm Optimization
SA	Simulated Annealing
S_{max}	Maximum absolute value of frequency response in stopband
$\boldsymbol{W(\omega)}$	Weighting function
$\boldsymbol{\omega}$	Normalized frequency
$\boldsymbol{\omega_p}$	Passband cut-off frequency
$\boldsymbol{\omega_s}$	Stopband cut-off frequency
GFIR	General Finite Impulse Response
Pg	Page

Chapter 1
Introduction

As the technology is reaching its heights, there's a lot of research work continuing in the field of digital filters in order to increase their performance, speed and reduce the size, power, and cost of the end products. Digital filters are discrete time devices used to perform operations on an input signal to obtain an output sequence according to a pre-designed difference equation. Inside a digital filter, every sample from input to output sequences with their coefficient values are quantized to a definite word length and are then presented in the form of a binary sequence.

The block diagram representing the working of a digital filter can be seen in fig 1.1.

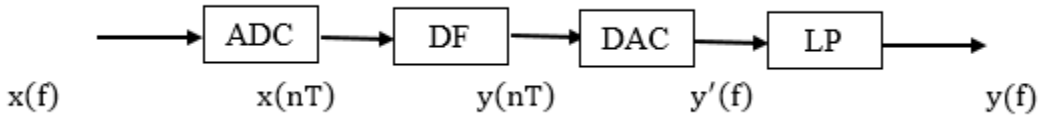


Figure 1.1 Digital filter working sequence (ADC: Analog to Digital Conversion; DAC: Digital to Analog Conversion; DF: Digital filter; LP: Analog lowpass filter). (Kwan [3], 2016, pg 6)

1.1 Why Digital filters?

Digital filters offer a number of applications with many advantages over analog filters. These advantages of digital filters affect the speed and performances of the overall system as compared to the same system being used with analog filters which require advanced mathematical knowledge and an understanding of the processes involved in the system affecting the filter. Few of the many good characteristics of digital filters as described in Kwan [3] are given below:

- Reliability: Digital filters are reliable that is there is no tolerance problem or aging in them.
- Accurate: By adjusting the digital word length, one can precisely control the accuracy of digital filters. It can be made approximately equal or extremely close to the ideal values by adjusting the characteristics of a digital filter.
- Flexible: By having another set of coefficient values, one can easily change the characteristics of a digital filter.
- Accommodating: It is possible to filter input sequences without any hardware replication.
- Efficient: Digital filters have a superior performance-to-cost ratio and do not drift with temperature or humidity or require precision components.
- Simplicity: Digital filters are software programmable, which makes them easy to build and test. Digital filters require only the arithmetic operations of addition, subtraction, and multiplication.

Digital filters are extremely stable due to their inherent mathematical construction. The frequency response does not change with time. On the other hand, due to the finite tolerances involved in the manufacture of electronic components, the frequency responses of similar analog filters are never exactly same. Also, the values of capacitors, resistors, etc. used in the analog filters may change with aging, resulting in the change in filter characteristics.

1.2 Types of filters

1.2.1 Based on the frequency response

Based on the frequency response, the filters can be categorized in the following 4 common types, as shown in the Fig. 1.1.

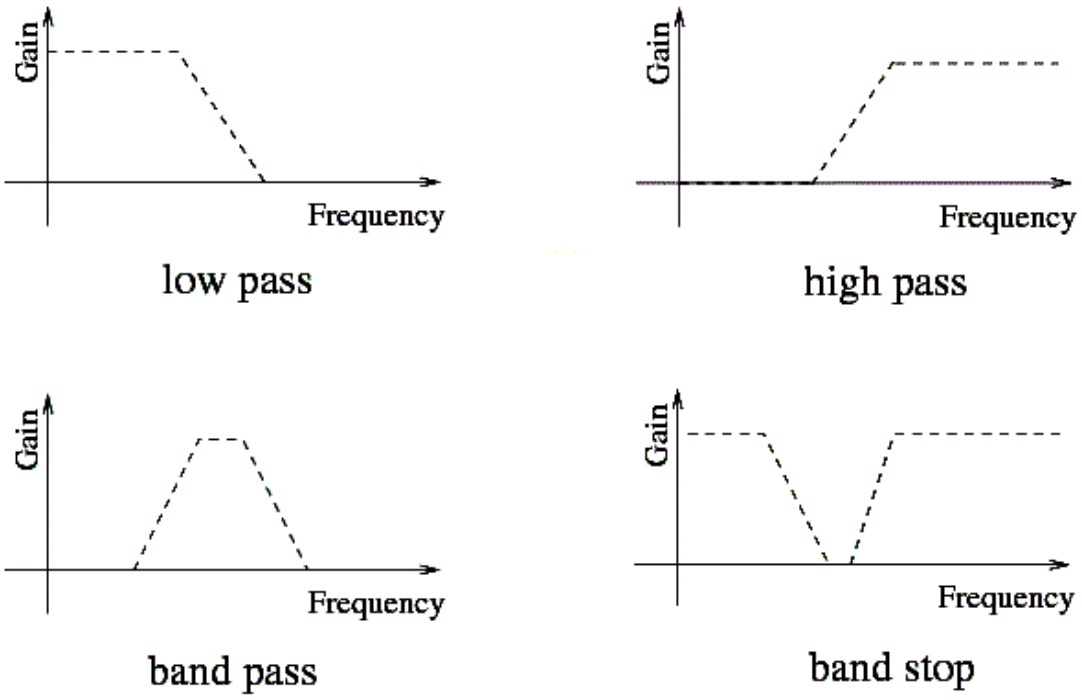


Figure 1.2 Types of filters based on the frequency response

These are described below:

1. Low pass filter: The low pass filter allows low frequencies to pass while removing the high frequencies.
2. High pass filter: The high pass filter allows high frequencies to pass while removing the low frequencies.
3. Bandpass filter: The bandpass filter allows only a certain band of frequencies to pass.
4. Bandstop filter: The bandstop filter allows all frequencies to pass except a band which it removes.

In addition to these basic types, there are other types such as notch filters, comb filters, etc. The slanted part of the frequency response of the filters is called the transition region, in which the frequency response transitions from the pass band to stop band or vice versa. A good filter must have a narrow transition band.

1.2.2 Based on the impulse response

Based on the impulse response, there are 2 categories of digital filters, namely, finite impulse response (FIR) and infinite impulse response (IIR) filters. As the names suggest, when an impulse input is given to the FIR filter, the output decays to 0 in a finite amount of time. On the other hand, the output takes an infinite amount of time to decay to 0 in the case of an IIR filter. This is due to the recursive nature of an IIR filter, where the output is fed back to the filter, resulting in an output even when the input has been stopped.

1.3 Digital FIR filter design

FIR is short for finite impulse response and is also called non-recursive filter. This kind of digital filter exhibits a finite duration impulse response. An FIR filter is designed by finding the coefficients and filter order that meets certain specifications, which can be in the time-domain (e.g. a matched filter) and/or the frequency domain (most common). Matched filters perform a cross-correlation between the input signal and a known pulse-shape. The FIR convolution is a cross-correlation between the input signal and a time-reversed copy of the impulse response. Therefore, the matched filters impulse response is "designed" by sampling the known pulse-shape and using those samples in reverse order as the coefficients of the filter For an FIR filter whose impulse response of length $N=R+1$, R being the order, is given by $\mathbf{h}=[h0 \quad h1 \quad h2 \quad \ldots hN-1]^{T}$. The transfer function or the z transform can be written as equation 1.1.

$$H(z) = \sum_{n=0}^{N} c(n) z^{-n} \tag{1.1}$$

Where H(z) denotes a polynomial written in ascending powers of $z-1$. The coefficients (n) for $n \geq 0$ represent the impulse response values of the FIR digital filter.

When a particular frequency response is desired, several different design methods are common:

- Window design method
- Frequency Sampling method
- Weighted least squares design
- Parks-McClellan method

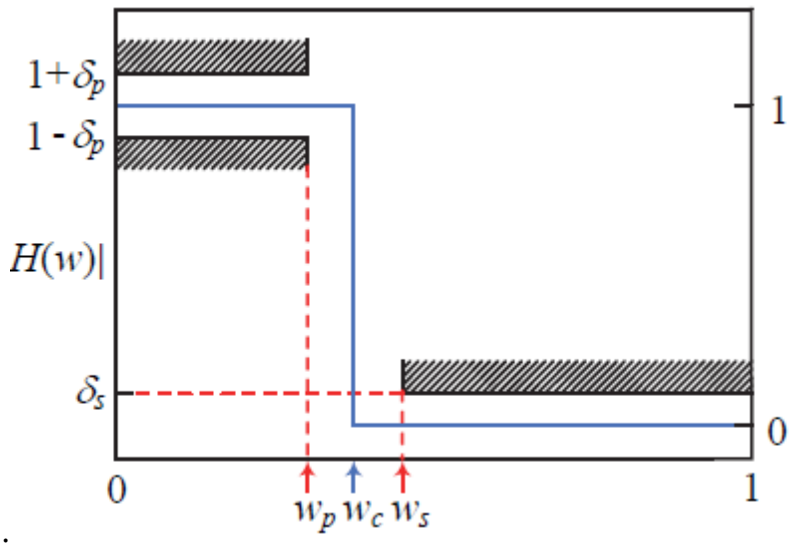


Figure 1.3 Ideal Lowpass digital FIR filter. (Kwan [3], 2016, pg 18)

All the four types of linear FIR filters can be achieved by using the properties of their coefficients symmetry. Usually, digital filters design involves the four main steps: approximation, realization, quantization consideration and implementation. By using

software simulating, the proper specifications such as amplitude response and phase properties can be completed.

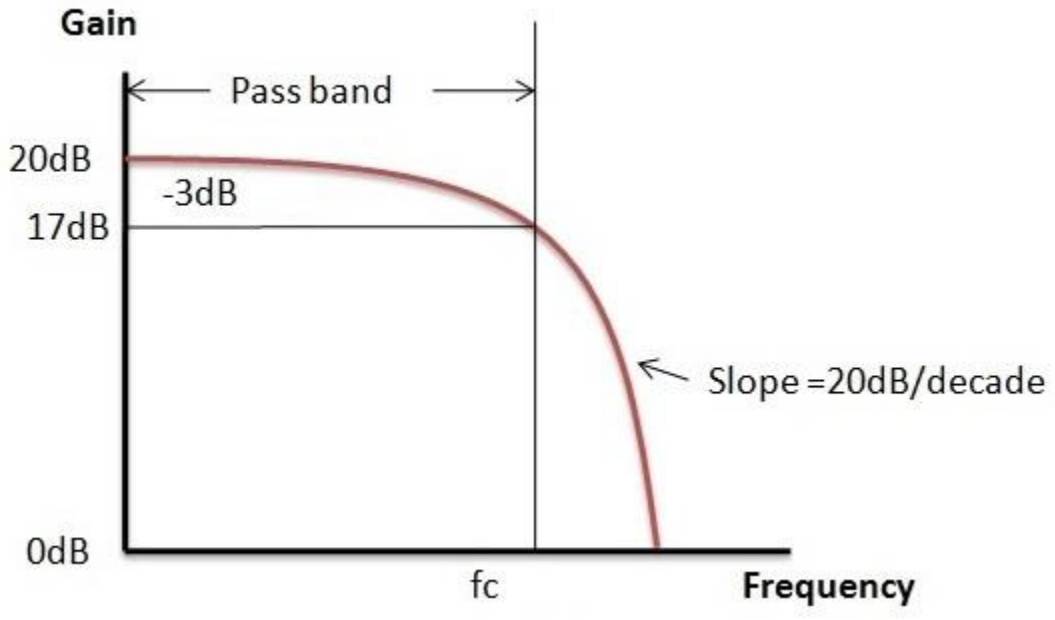


Figure 1.4 Practical lowpass digital filter

Fig. 1.5 shows the direct implementation of an FIR filter. T is delay and h_0, $h_1 \ldots h_{N-1}$ are filter coefficients. $x[k]$, $x[k-1]$, … $x[k-N+1]$ are the input and the delayed versions of the input. $y[k]$ is the output of the filter. It can be noted from the figure that there is no feedback from the output of the filter.

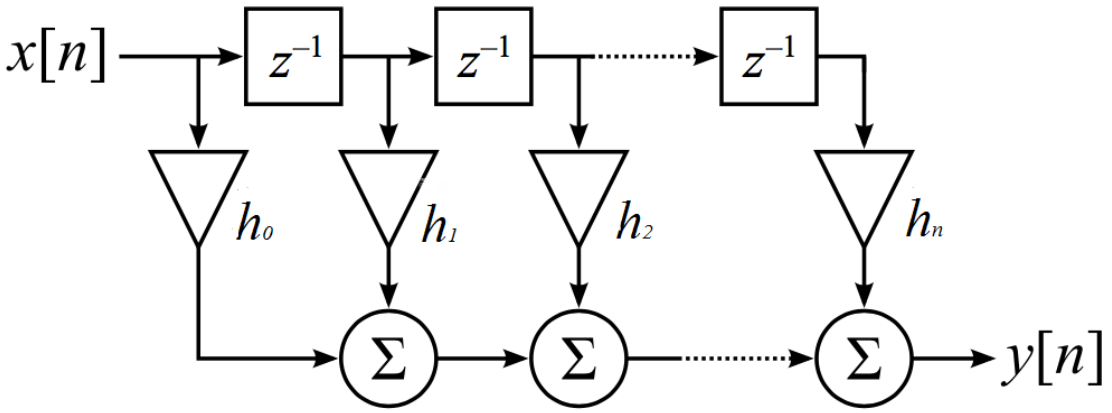


Figure 1.5 FIR direct form 1

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

$$y[k] = \ h_0 x[k] + h_1 x[k-1] + \cdots + h_{n-1} x[k-N+1] \tag{1.2}$$

The transfer function of the FIR filter in the z-domain can be written as

$$H(z) = \sum_{n=0}^{N-1} h_n z^{-n} \tag{1.3}$$

The frequency response of the filter can be found by substituting z with $e^{j\omega T}$ as shown below, where ω is the frequency of the input signal.

$$H\left(e^{j\omega T}\right) = \sum_{n=0}^{N-1} h_n e^{-j\omega nT} = \sum_{n=0}^{N-1} h_n \cos(\omega nT) - j \sum_{n=0}^{N-1} h_n \sin(\omega nT) \tag{1.4}$$

Generally, in practical implementations, the direct transposed form is used for the FIR filter. This has the advantage, as it does not require the extra shift register for the input.

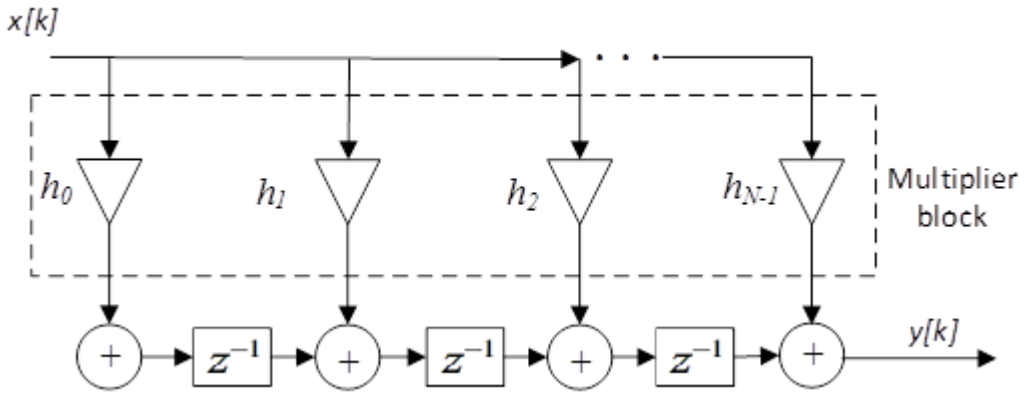


Figure 1.6 FIR filter in transposed direct form

1.3.1 Linear Phase Filters

The equations 1.2-1.4 represent general FIR filters with the arbitrary magnitude and phase response. It can be shown that it is possible to construct FIR filters with linear phase response. This is possible when the filter coefficients have an even or odd symmetry. Depending on the order of the filter and the symmetry of the filter coefficients, the linear phase filters can be of four types as shown in Table 1.1.

TABLE 1.1 Symmetric Filters

Type	Order	Symmetry	$\boldsymbol{H(\omega)}$at $\boldsymbol{\omega = 0}$	$\boldsymbol{H(\omega)}$ at $\boldsymbol{\omega = \pi/T}$
Type I	even	even	any	any
Type II	odd	even	any	0
Type III	even	odd	0	0
Type IV	odd	odd	0	any

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

$$\begin{aligned} &\text{Type I:} \quad h_k = h_{N-k+1}, \quad N \text{ is odd} \\ &\text{Type II:} \quad h_k = h_{N-k+1}, \quad N \text{ is even} \\ &\text{Type III:} \quad h_k = -h_{N-k+1}, N \text{ is even} \\ &\text{Type IV:} \quad h_k = -h_{N-k+1}, \quad N \text{ is odd} \end{aligned} \tag{1.5}$$

The amplitude responses of the four types of the filters can be expressed by the following equations:

Type I:

$$H(\omega) = h(M) + 2\sum_{n=0}^{M-1} h_n \cos((M-n)\omega) \tag{1.6}$$

Type II:

$$H(\omega) = 2\sum_{n=0}^{N/2-1} h_n \cos((M-n)\omega) \tag{1.7}$$

Type III:

$$H(\omega) = 2\sum_{n=0}^{M-1} h_n \sin((M-n)\omega) \tag{1.8}$$

Type IV:

$$H(\omega) = 2 \sum_{n=0}^{N/2-1} h_n \cos((M-n)\omega) \qquad (1.9)$$

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

It can be seen that Type I can be used to construct both low and high pass filters, Type II can be used to construct only low pass filters, Type IV can be used to construct only high pass filters and Type III can be used to construct only bandpass filters. Due to this, Type I filters are most common.

In the following figure, coefficient values of a Type I linear phase filter are shown:

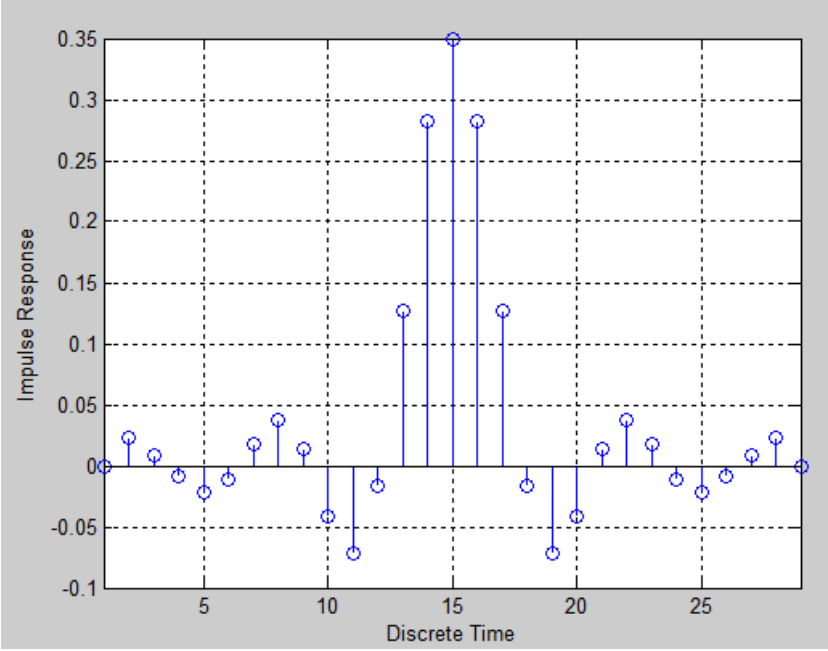


Figure 1.7 Type I FIR filter coefficients

The coefficients are symmetric around the central coefficient, as can be seen in the above figure. The phase response of a linear phase filter is shown in the following figure

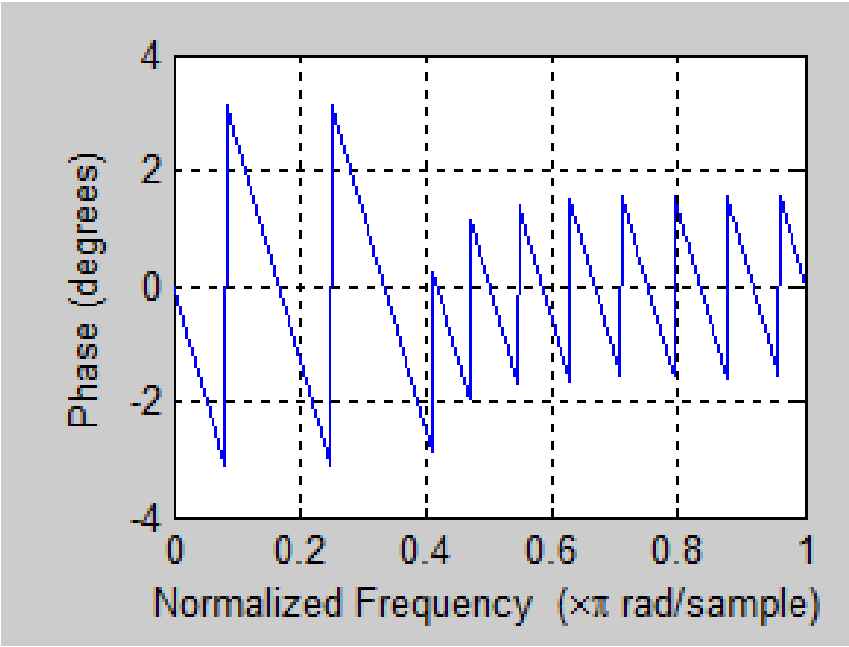


Figure 1.8 Phase response of a linear phase filter

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

1. $2\pi + \theta = \theta$, resulting in the phase being confined from $-\pi$ to π.
2. The sign reversal of the frequency response.

1.4 Design of FIR filters

FIR filters can be designed using various methods. The most common of these methods are:

1. Equi-ripple (minimax) design in which the maximum frequency response error from the specified frequency response is minimized. Parks-McClellan method can be used to design FIR filters based on minimax criterion.
2. Least mean square design, in which the mean square error is minimized from the desired frequency response.
3. Window-based methods based on inverse DFT.

Here, we will focus on the minimax design of FIR filters, as this is the criterion on which the work in this thesis is based.

1.4.1 Equi-ripple design of linear phase FIR filters

The transfer function of the Type 1 filter is shown in equation (2.1) as

$$\mathrm{H}(\mathbf{c},\mathrm{w}) = \mathrm{e}^{-\mathrm{j}\left(\frac{\mathrm{M}-1}{2}\right)\mathrm{wT}}\left\{\mathrm{h}\left(\frac{\mathrm{M}-1}{2}\right) + \sum_{\mathrm{n}=0}^{\frac{\mathrm{M}-3}{2}} 2\mathrm{h}(\mathrm{n})\cos\left[\left(\frac{\mathrm{M}-1}{2}-\mathrm{n}\right)\mathrm{wT}\right]\right\} \tag{1.10}$$

$$\mathrm{H}(\mathbf{c},\mathrm{w}) = \mathrm{e}^{-\mathrm{j}\left(\frac{\mathrm{M}-1}{2}\right)\mathrm{wT}}\mathrm{A}(\mathbf{c},\mathrm{w}) \tag{1.11}$$

Where,

$$\mathrm{A}(\mathbf{c},\mathrm{w}) = \mathbf{c}^{\mathrm{T}}\mathbf{cos}(\mathrm{w}) \tag{1.12}$$

And,

$$\mathbf{cos}(\mathrm{w}) = \left[1 \quad \cos(\mathrm{wT}) \quad \cos(2\mathrm{wT}) \quad \cdots \quad \cos\left(\frac{\mathrm{M}-1}{2}\mathrm{wT}\right)\right]^{\mathrm{T}} \tag{1.13}$$

The coefficient vector $\mathbf{c}^T$ is optimized initially starting with the random values and then following the structure of the algorithm used to reduce the objective function value at every iteration which I reduced by minimizing the error value at every iteration. To calculate the error, the minimax error approximation method is used. It takes the error between the frequency response of the ideal and designed filter. An ideal filter has a magnitude of 1 on the passband and 0 on the stopbands. So, the error of the iteration values is given by the absolute difference between the ideal magnitudes and the actual filter coefficient values in that iteration. The expression of the minimax function is given by

$$e_p(\mathbf{c}) = \left[\sum_{i=1}^{I_p} W_p(w_i)\left||A(\mathbf{c}, w_i)| - A_d(w_i)\right|^{2p}\right]^{1/2p} \quad \text{for } W_p(w_i) \geq 0;\ 0 \leq w_i \leq w_p \tag{1.14}$$

$$e_s(\mathbf{c}) = \left[\sum_{i=1}^{I_s} W_s(w_i)\left||A(\mathbf{c}, w_i)| - A_d(w_i)\right|^{2p}\right]^{1/2p} \quad \text{for } W_s(w_i) \geq 0; w_s \leq w_i \leq \pi \tag{1.15}$$

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

$$\min_{\mathbf{c}} e(\mathbf{c}) \tag{1.16}$$

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

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

Weighting scores may vary depending on the error values and if the error of passband and stopband values differs by quite a margin then the weight values can be increased or decreased accordingly.

Figure 1.3 represents the desired filter response, which corresponds to an ideal low-pass filter, with cut-off frequency ω_c. The response is exactly 1 in the passband and drops to 0 in the stopband sharply. In practical filters of finite length, the frequency response deviates from the ideal response as shown in the solid curve in the figure 1.4. Therefore, the specifications of practical filters are relaxed compared to the ideal filters.

The interval 0-ω_p is the pass-band and ω_c-1 is the stop-band of the filter. For the designed filter, in the pass-band, the response can vary from 1-δ_p to 1+δ_p and in the stopband from -δ_s to δ_s. In the transition region, which is from ω_p to ω_s the response can take any value. δ_p is called the pass-band ripple and δ_s the stop-band ripple. The tighter the filter specifications are, the higher the filter length, N is required to design the filter.

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

Minimize $\boldsymbol{\delta}$

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

$$-(\delta_s\delta)/\delta_p \leq H(\omega) \leq (\delta_s\delta)/\delta_p, \text{for } \omega \in [\ \omega_s, 1] \tag{1.17}$$

Where, $H(\omega)$ is the frequency response of the filter and is given by

$$H(\omega) = \sum_{n=0}^{\left\lfloor\frac{N-1}{2}\right\rfloor} h(n)\text{Trig}(\omega, n) \tag{1.18}$$

Where N is the filter length and Trig is a trigonometric function depending on the type of the filter and whether the filter length is odd or even (See equations 1.6-1.9).

Solving the linear program (LP), we can find the values of the filter coefficients $h(n)$ and the ripple δ.

The filter can be instead designed using Parks-McClellan method which is very efficient. This is an iterative algorithm, reducing the maximum error in each iteration. The MATLAB function **firpm** is based on the Parks-McClellan method and can be used to design linear-phase FIR filters with a given length and specified pass and stop bands. 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 pass and stop bands. For example, f = [0 0.3 0.5 1], and a = [1 1 0 0] represents a low pass filter with passband from 0 to 0.3π and stop band from 0.5π to π.

w is the weight vector of length equal to the number of bands. Each value in the vector represents the weight assigned to the corresponding band of the filter.

1.4.2 Advantages of FIR filters

An FIR filter has a number of useful properties which sometimes make it preferable to an infinite impulse response (IIR) filter. FIR filters:

- Require no feedback. This means that any rounding errors are not compounded by summed iterations. The same relative error occurs in each calculation. This also makes implementation simpler.
- Are inherently stable because the output is a sum of a finite number of finite multiples of the input values.

- Can easily be designed to be linear phase by making the coefficient sequence symmetric. This property is sometimes desired for phase-sensitive applications, for example, data communications, seismology, crossover filters, and mastering.

The main disadvantage of FIR filters is that considerably more computation power in a general purpose processor is required compared to an IIR filter with similar sharpness or selectivity, especially when low frequency (relative to the sample rate) cut offs are needed. However many digital signal processors provide specialized hardware features to make FIR filters approximately as efficient as IIR for many applications.

1.5 Quantization of coefficients

The continuous filter coefficients can be quantized using either uniform quantization or non-uniform quantization. The uniform quantization can be achieved by the following number representations:

1.5.1 Signed magnitude representation

In this representation, the magnitude of the number is represented by the bits excluding the MSB and the sign of the number is represented by the MSB.

For example: $(0101)_2 = +5_{10}$ and $(1010)_2 = -2_{10}$

The number 0 has 2 possible representations in this system, which are $(0000)_2$ and $(1000)_2$.

1.5.2 One's complement representation

In this representation, the negative of a number is equal to bitwise OR of the number.

For example: $(0110111)_2 = +55_{10}$ and $(1001000)_2 = -55_{10}$

In order to add two one's complement numbers, it is necessary to add the end-around carry to the result to obtain the correct answer. For example:

$(1110001)_2 + (0010000)_2 = (1\ 0000001)_2$

To obtain the correct answer the carry bit is added to the remaining number, which gives

$(1)_2 + (0000001)_2 = (0000010)_2$

1.5.3 Two's complement representation

To avoid the task of adding the carry bit after the addition of two one's complement numbers, in the two's complement representation the negative of a number is formed by taking bit-wise not of the number and then adding 1 to the result.

For example: -55_{10} is represented by $(1001000)_2 + (1)_2 = (1001001)_2$

The addition of two 2's complement numbers is straightforward and can be done using normal addition.

1.5.4 Signed digit format

In signed digit format each digit of the number has a sign associated with it. One example is balanced ternary, whose base is 3 and the digits can take the values from {-1, 0, 1}.

For example: $(1\ 0\ -1\ -1)_2 = 2^3 - 2^1 - 2^0 = 8 - 2 - 1 = 5$

The signed digit format is not unique.

1.5.5 Canonical signed digit representation

If in the signed digit representation no two consecutive digits are non-zero, then the resulting representation is called canonical signed digit representation (CSD). The CSD representation of a number is unique.

1.5.6 Non-uniform quantization

There are a number of quantization representations in which the difference between 2 consecutive values in the range of the representation does not remain uniform. An example is limiting the number of signed non-zero digits in the representation of the filter

coefficients. Limiting the number to 2, the filter coefficients are then represented by the following equation

$$h_n = c_{n1}2^{-b_{n1}} + c_{n2}2^{-b_{n2}} \quad (1.19)$$

Where,

$c_{n1}, c_{n2} \in \{-1,0,1\}$ and $b_{n1}, b_{n2} \in \{1,2, \dots b\}$

1.5.7 Integer representation of coefficients

The coefficients are quantized to a certain number of bits in an algorithm. The number of digits in binary format of a coefficient, proceeding initial zeros after quantization is called the effective word-length of the coefficient. For example, consider a filter with 3 coefficients with values 0.4569, -0.2438 and 0.1211. The binary values of these coefficients are

0.01110100111101110110…, -0.00111110011010011010… and
0.00011111000000000110…

If these are rounded to 9 digits after the binary point, then the values become

0.011101010, -0.001111101 and 0.000111110

The EWL of each coefficient is then equal to the number of digits of each coefficient after the initial zeros, which is 8, 7 and 6 respectively.

Instead of working on these values which are binary, these can be multiplied by a number which is a power of 2, i.e. 2^n, such that the resulting values become integers. In the above example, it can be easily seen that this number is 2^9. Multiplying the coefficients by 2^9, we get

11101010, -1111101 and 111110, which in decimal are 234, -125 and 62.

1.6 Initial Coefficients

The FIR filters and neural network designs selected for optimization has the Initial coefficients set such that it maximizes the randomization so that all the filter designs can be started from the scratch giving algorithm most of the work which ensures the quality of the design. The initial coefficients used for the designs in this thesis are given as follows:

Let $C_k^{[U]}$and $C_k^{[L]}$denote the upper bound and the lower bound of the kth coefficient C_k of a LP or HP or BP or BS used in the equations such that

$$C_k^{[L]} \leq C_k \leq C_k^{[U]} \; for \; 1 \leq k \leq \frac{N}{2} + 1 \tag{1.20}$$

The initial coefficient value of C_{pk} for a population member p is computed by

$$C_{pk} = C_k^{[L]} + rand * \left(C_k^{[U]} - C_k^{[L]}\right) for \; p = 1{:}P, k = 1{:}K \tag{1.21}$$

Where $rand$ is a uniformly distributed value between [0, 1].

1.7 General FIR Filters

GFIR filters known as General Finite Impulse Response filters are also a kind of filter design which is used where a constant group delay is required in the system. That is, a phase shift is not required between input and output of the filter. Such a filter is no longer a symmetric filter because it will not acquire same coefficients on the left and the right-hand sides, therefore each coefficient in a GFIR filter is different from the other. It makes it difficult to optimize and get a better result because there are nearly twice as many coefficients to be optimized and also a group delay factor is also there which needs to be taken care of. Because of the complex nature of the design, only passband group delay is taken into account and only the part in the frequency band where passband resides is optimized. The passband and stopband errors of the GFIR filters are calculated in the same

manner as the FIR filters given by the equations 1.14 and 1.15. The transfer function of an *N*-th order FIR digital filter consisting of N+1 coefficients can be written as

$$H(z) = \sum_{n=0}^{N} c(n) z^{-n} = \boldsymbol{c}^T \boldsymbol{z}\,(z) \tag{1.22}$$

From (1.20), it can be seen that **c** is the coefficient vector and *H*(z) denotes a polynomial written in ascending powers of z^{-1}. The magnitude response $|H(w)|$ of GFIR filter is equal to

$$|H(w)| = \left\{ \left[\sum_{n=0}^{N} c_n \text{cosnwT} \right]^2 + \left[\sum_{n=0}^{N} c_n \text{sinnwT} \right]^2 \right\}^{1/2} \tag{1.23}$$

The phase response, $\theta(w)$ is given by,

$$\theta(w) = -tan^{-1} \left[\frac{\sum_{n=0}^{N} c_n \text{sinnwT}}{\sum_{n=0}^{N} c_n \text{cosnwT}} \right] \tag{1.24}$$

From (1.22), the group delay $\tau(w)$ can be expressed as

$$\tau(w) = -\frac{\partial \theta(w)}{\partial wT} = \frac{1}{1 + c^2} \frac{\partial c}{\partial wT} \tag{1.25}$$

where

$$c = \frac{\sum_{n=0}^{N} c_n \text{sinnwT}}{\sum_{n=0}^{N} c_n \text{cosnwT}} \tag{1.26}$$

Taking partial derivative of (1.24), we have

$$\frac{\partial c}{\partial wT} = \frac{[\sum_{n=0}^{N} c_n \text{cosnwT}][\sum_{n=0}^{N} n c_n \text{cosnwT}]}{[\sum_{n=0}^{N} c_n \text{cosnwT}]^2} + \frac{[\sum_{n=0}^{N} c_n \text{sinnwT}][\sum_{n=0}^{N} n c_n \text{sinnwT}]}{[\sum_{n=0}^{N} c_n \text{cosnwT}]^2} \tag{1.27}$$

1.8 Optimization algorithms

A number of optimization algorithms have been discussed in [3] which integrates important FIR filter designs with optimization algorithms. In the following section, some important deterministic and heuristic algorithms from the literature are reviewed.

1.8.1 Deterministic algorithms

In computer science, a deterministic algorithm is an algorithm which, given a particular input, will always produce the same output, with the underlying machine always passing through the same sequence of states. Deterministic algorithms are by far the most studied and familiar kind of algorithm, as well as one of the most practical since they can be run on real machines efficiently.

Formally, a deterministic algorithm computes a mathematical function; a function has a unique value for any input in its domain, and the algorithm is a process that produces this particular value as output. If the system is deterministic, this means that from this point onwards, its current state determines what its next state will be; its course through the set of states is predetermined. Note that a system can be deterministic and still never stop or finish, and therefore fail to deliver a result.

If the feasible range of any coefficient is found to be empty at any stage, the algorithm backtracks to the previous quantized coefficient and quantizes it to the next nearest quantization value from the center of its feasible range.

First, the lower and upper bound of each coefficient is calculated [5]. This is done by solving the following linear program problem:

$$\begin{aligned} &\text{minimize: } f = h(k) \text{ and } f = -h(k) \\ &\text{such that: } b - \delta \le H(\omega) \le b + \delta, \text{for } \omega \in [0, \omega_p] \\ &-(\delta_s \delta)/\delta_p \le H(\omega) \le (\delta_s \delta)/\delta_p \text{ , for } \omega \in [\omega_{s,} \pi] \\ &b_l \le b \le b_h \end{aligned} \tag{2.1}$$

Where b is the passband gain and δ_p and δ_s are the maximum allowed pass and stop-band ripple. $H(\omega)$ is the magnitude of the frequency response of the filter.

After this, a depth-first search is done and filter coefficients are fixed to integers one by one. Once a coefficient is fixed the remaining un-quantized one are re-optimized.

These algorithms are guaranteed to return an optimum set of filter coefficients, but for filters with high word-lengths, the algorithm takes very long time to finish the search and becomes impractical. The run-time increases exponentially with the increase of the filter length.

A variety of factors can cause an algorithm to behave in a way which is not deterministic, or non-deterministic:

- If it uses external state other than the input, such as user input, a global variable, a hardware timer value, a random value, or stored disk data.
- If it operates in a way that is timing-sensitive, for example, if it has multiple processors writing to the same data at the same time. In this case, the precise order in which each processor writes its data will affect the result.
- If a hardware error causes its state to change in an unexpected way.

Although real programs are rarely purely deterministic, it is easier for humans as well as other programs to reason about programs that are. For this reason, most programming languages and especially functional programming languages make an effort to prevent the above events from happening except under controlled conditions.

Deterministic algorithms are capable of finding solutions which, in most of the cases, are optimal or near optimal.

1.8.2 Heuristic algorithms

The term heuristic is used for algorithms [11] which find solutions among all possible ones, but they do not guarantee that the best will be found, therefore they may be considered as approximately and not accurate algorithms. These algorithms, usually find a solution close to the best one and they find it fast and easily. Sometimes these algorithms can be accurate, that is they actually find the best solution, but the algorithm is still called heuristic until this best solution is proven to be the best. The method used from a heuristic algorithm is one of the known methods, such as greediness, but in order to be easy and fast the algorithm ignores or even suppresses some of the problem's demands.

In [5], GA is used to design FIR filter with coefficients which are constrained to the sums of two numbers, which are powers of two. In order to constrain the search space, a specific coefficient coding scheme is used. Instead of coding the values of the coefficients directly, the differences from some leading values are chosen and coded. The leading values are chosen as the coefficients obtained after quantizing the optimal continuous coefficients. In the case, when the sum of the power of two terms is used, the quantization is done such that the quantized value is the nearest value in the domain. As the optimal discrete filter coefficients are generally relatively not far away from the continuous filter coefficients, therefore the differences to be encoded are not very large and can be encoded using lesser number of bits, compared to encoding the full coefficient values.

Usually, heuristic algorithms are used for problems that cannot be easily solved [34]. Classes of time complexity are defined to distinguish problems according to their “hardness”. Class P consists of all those problems that can be solved on a deterministic Turing machine in polynomial time from the size of the input. Turing machines are an abstraction that is used to formalize the notion of algorithm and computational complexity. Class NP consists of all those problems whose solution can be found in polynomial time on a non-deterministic Turing machine. Since such a machine does not exist, practically it means that an exponential algorithm can be written for an NP-problem, nothing is asserted whether a polynomial algorithm exists or not. A subclass of NP, class NP-complete includes problems such that a polynomial algorithm for one of them could be transformed to polynomial algorithms for solving all other NP problems. Finally, the class NP-hard can

be understood as the class of problems that are NP-complete or harder. NP-hard problems have the same trait as NP-complete problems but they do not necessarily belong to class NP, that is class NP-hard includes also problems for which no algorithms at all can be provided. In order to justify application of some heuristic algorithm, we prove that the problem belongs to the classes NP-complete or NP-hard. Most likely there are no polynomial algorithms to solve such problems, therefore, for sufficiently great inputs heuristics are developed. Some of the very effective heuristic algorithms are described below:

Swarm intelligence was introduced in 1989. It is an artificial intelligence technique, based on the study of collective behavior in decentralized, Self-organized, systems [15, 23, 49]. Two of the most successful types of this approach are *Ant Colony Optimization (ACO)* and *Particle Swarm Optimization (PSO)* [3]. In ACO artificial ants build solutions by moving on the problem graph and changing it in such a way that future ants can build better solutions. PSO deals with problems in which the best solution can be represented as a point or surface in an n-dimensional space. The main advantage of swarm intelligence techniques is that they are impressively resistant to the local optima problem.

1.8.3 Evolutionary algorithms

Evolutionary algorithms [10, 54] are methods that exploit ideas of biological evolution, such as reproduction, mutation, and recombination, for searching the solution of an optimization problem. They apply the principle of survival on a set of potential solutions to produce gradual approximations to the optimum. A newest of approximations is created by the process of selecting individuals according to their objective function, which is called fitness for evolutionary algorithms and breeding them together using operators inspired from genetic processes [47]. This process leads to the evolution of populations of individuals that are better suited to their environment than their ancestors. The main loop of evolutionary algorithms includes the following steps:

1. Initialize and evaluate the initial population.

2. Perform competitive selection.

3. Apply genetic operators to generate new solutions.

4. Evaluate solutions in the population.

5. Start again from point 2 and repeat until some convergence criteria are satisfied.

Sharing the common idea, evolutionary techniques can differ in the details of implementation and the problems to which they are applied. Genetic programming searches for solutions in the form of computer programs. Their fitness is determined by the ability to solve a computational problem. The only difference from evolutionary programming is that the latter fixes the structure of the program and allows their numerical parameters to evolve. Evolution strategy works with vectors of real numbers as representations of solutions and uses self-adaptive mutation rates. The most successful among evolutionary algorithms are *Genetic Algorithms* (GAs) [5]. They have been investigated by John Holland in 1975 and demonstrate essential effectiveness. GAs are based on the fact that the role of mutation improves the individual quite seldom and, therefore, they rely mostly on applying recombination operators. They seek solutions to the problems in the form of strings of numbers, usually binary. There are a number of different Evolutionary algorithms that are being developed and are consistently being research on because of their efficiency and effectiveness to perform a complex task in a lesser amount of time. The evolutionary algorithm chosen for this particular thesis is *Firefly Algorithm* [1] which demonstrates the optimization of complex problems using the behavior of fireflies. Another algorithm which is further explained in 1.9 is the *Differential Evolution* (DE) algorithm [6] which is one of the simplest yet effective evolutionary algorithms and is globally used to solve various complex problems. Digital filters are designed using FA and DE and then compared for performances in the later chapters.

1.9 Differential Evolution:

In the initial research for this thesis, Differential Evolution method was applied in order to optimize the filter coefficients and obtain error values as minimum as possible. Differential Evolution (DE) [65] is a stochastic parallel search method that can efficiently optimize non-differentiable, nonlinear objective functions. DE requires only a few parameters to manage its operations. The reason why DE is called a parallel method is that it tries to optimize a number of parameter vectors at the same time, thus creating a higher chance of finding a global optimum.

DE creates new members in the solution space by adding and scaling members from the existing population. This makes it a self-referential method. Since scaling and addition are linear operators, the initial probability distribution of the members is kept the same. This property makes the DE scheme completely self-organizing.

The major characteristics of DE include robust; fast, simple and easy to use, effective with good global optimization abilities; inherently parallel; no predefined probability distribution is required; amendable to real, integer and mixed-parameter optimization; precision limited only by floating-point format; and applicable to noisy objective functions.

Differential evolution [65] requires only a few parameters to manage its operations. Using differential evolution does not require knowledge of its underlining principles and other evolutionary optimization techniques. In general, differential evolution is robust; fast; simple and easy to use.

DE algorithm uses mutation operation as a search mechanism and selection operation to direct the search toward the prospective location in the search space. It also uses crossover operation to increase the diversity of the perturbed coefficient vector, which can take child coefficient vector from one parent more often than it does from others. If the new vector yields a lower objective function than before, the new vector replaces the target vector in the next generation.

The main steps of the DE algorithm are given below:

- $Initialization$
- $Evaluation$
- $mutation$
- $crossover$
- selection

The crossover probability CR controls the fraction of the parameter values that are copied from the mutant. Mutation is applied in this way to each member of the population. If an element of the trial vector is found to violate the bounds after mutation and crossover, it is reset in such a way that the bounds are respected (with the specific protocol depending on the implementation). Then, the objective function values associated with the children are determined.

1.9.1 Digital Filters Using DE:

DE algorithm was implemented and results were calculated for a 24th order linear phase FIR type-1 filter design. More designs have been implemented and analyzed later in chapter 5 for comparison purposes. Here, to provide an understanding of the results produced by DE algorithm, some of the filter designs are selected and implemented to see how DE can provide good design results. The 24^{th} order Lowpass, Highpass, Bandpass and Bandstop filter designs using DE algorithm are shown in figure 1.9 – 1.12.

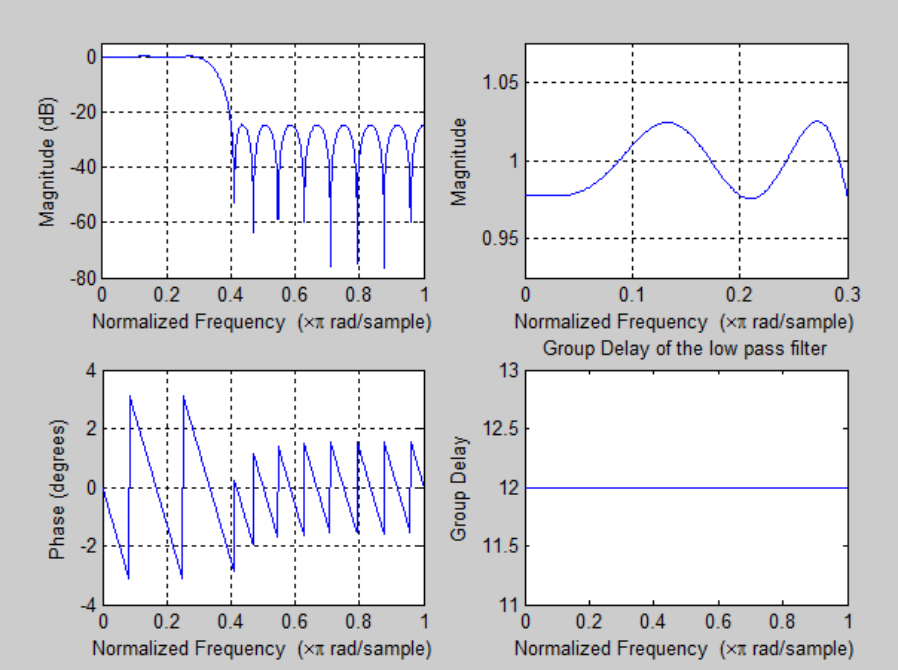


Figure 1.9 lowpass digital FIR filter using DE

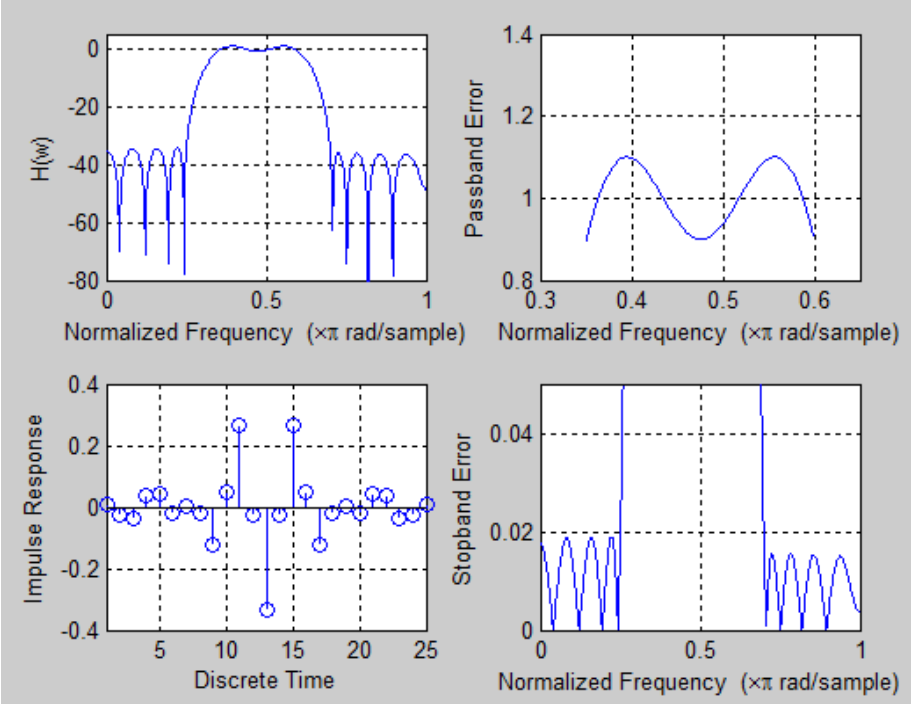


Figure 1.10 Bandpass digital FIR filter using DE

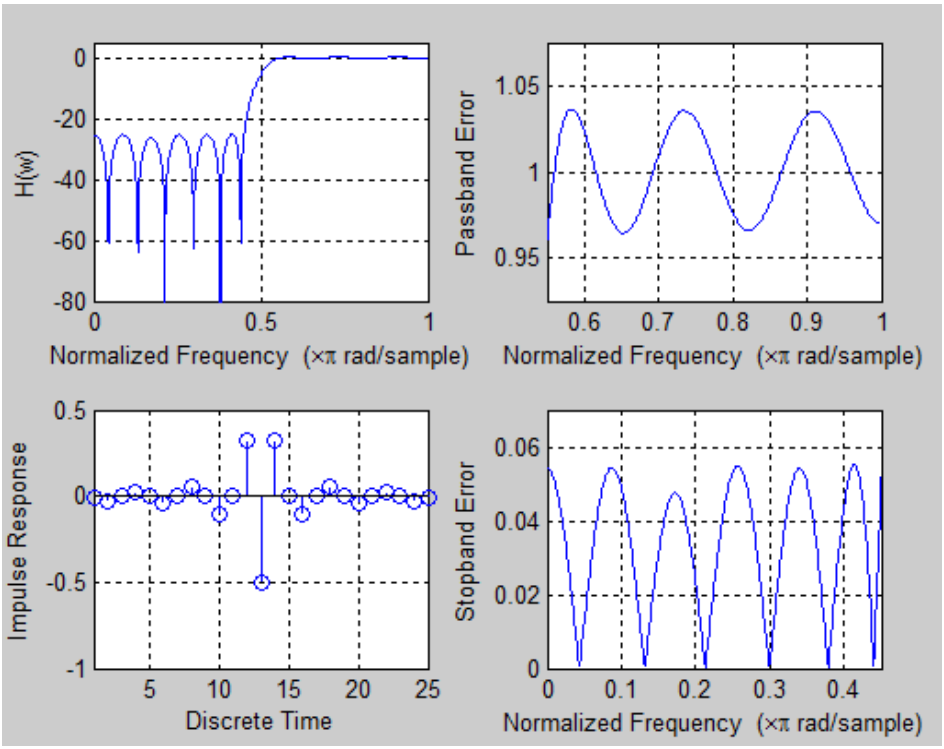


Figure 1.11 Highpass digital FIR filter using DE

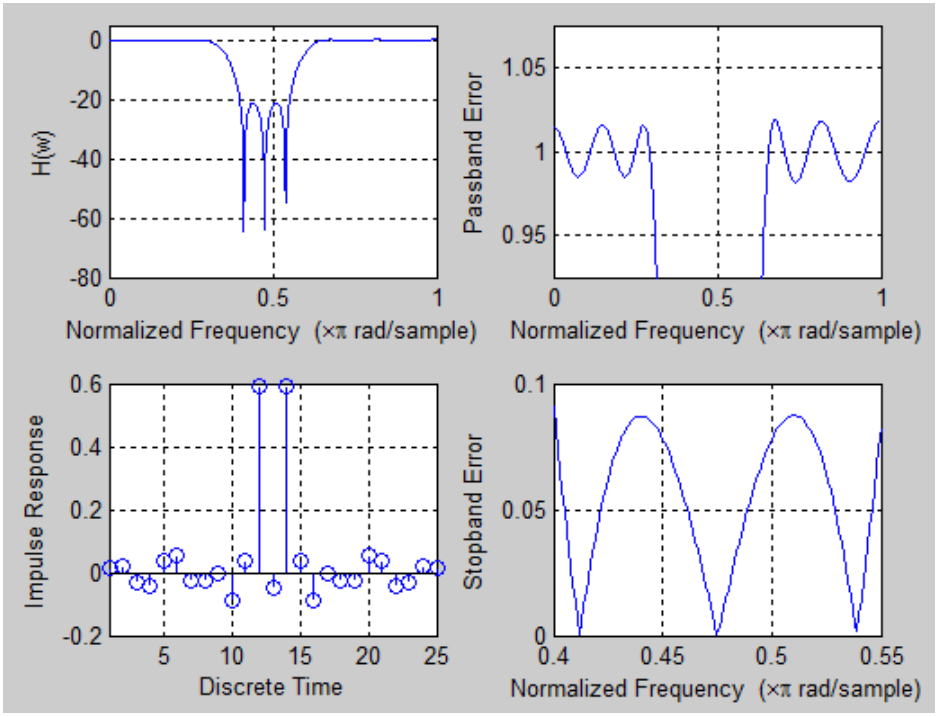


Figure 1.12 Bandstop digital FIR filter using DE

1.10 Motivation and goals

The goals of the proposed project are motivated by the need for a fast an effective optimization algorithm that is able to optimize complex problems such as designing optimized circuits for digital filters and neural networks making filtering processes more efficient and noise free. The algorithm analyzed has produced some promising results in the initial test runs (described in the later chapters) which motivated me to exploit more out of this algorithm applying it on different decent design problems and comparing results with those already present algorithms. Digital Filter design problems include digital filter coefficients values that need to be determined randomly in such a manner that it produces the least amount of noise and error values in the specified domain of coefficient values. For a successful digital filter design, the noise reduction is normally carried out by optimizing the coefficient values using an optimization algorithm. A neural network is a computer system modeled on the human brain and nervous system. Neural networks are made up of interconnected processing elements called units, which respond in parallel to a set of input signals given to each. The unit is the equivalent of its brain counterpart, the neuron. To design a specific network, determination of weights and bias values are the main optimization aspects that again, like digital filter coefficients, require an optimization algorithm that reduces the gap between the desired values and actual values in such a way that the neural network will itself give you the desired output by its own learning despite having a certain amount of noise at the input. Optimization algorithm plays an important role in determining the weights and biases of the network. Therefore the importance and significance of a good digital filter and neural network design operating on an effective optimization algorithm is the basic motivation behind the proposed thesis idea.

1.11 Thesis organization

This thesis compromises of a detailed study on Firefly Algorithm and its advantages, It has been used to design digital filters and neural networks and the organization of the compiled data is given as follows:

The first chapter has the introduction to the filters, advantages, and disadvantages of using digital filters to state a point why this design has been chosen to be implemented using FA. The conventional methods of filter designing and the common techniques used to design some of the filters. It also contains the motivation and goals of this thesis. The algorithms used in the designing of various filters including deterministic and heuristic methods are also discussed and explained briefly. In addition, a couple of algorithms such as Genetic algorithm and Differential evolution algorithms are also explained which are also used in the later chapters to design the same filters for comparison purposes with FA. In addition, some of the filter designs using DE are also given in the chapter.

In the second chapter, some of the state-of-the-art methods for designing linear phase FIR filters are discussed and the literature is reviewed. In the third chapter, Firefly algorithm is discussed in detail. Its orientation, similarities, advantages, disadvantages and all the information regarding the algorithm is discussed. Its applications before filter designs are also discussed briefly. Also, the mechanism and flow of algorithm are discussed along with its advantages and strong points over other well-known algorithms.

In the fourth chapter, Neural networks are discussed briefly, the need for implementation of neural nets along with its types and applications are discussed. A few described neural network problems are also presented in the chapter with their results using FA.

In the fifth chapter, the results of Firefly algorithm for digital filters and neural networks are analyzed. Some benchmark filters are designed using the algorithm and their frequency responses are presented. The run-time of the algorithm is also calculated and discussed and a comparison is made with the state of the art methods and other algorithms.

In the sixth chapter, the conclusion is given.

The main contribution of the work done here is the implementation of Firefly algorithm and adapting it to make it suitable for designing FIR filters and neural network designs. The algorithm has not been previously used for the design of digital FIR filters. The disadvantage of other algorithms such as GA and PSO is that they require fine-tuning of the parameters. Also, there is a problem of early convergence to a local minimum. FA, on the other hand, is very flexible with parameters, which are relatively easier to tune. Also, it is easy to tune FA such that it doesn't get trapped in local minima while being a very time effective method as well.

Chapter 2
Review of Literature

Filter optimization is not a topic new to us. There have been many advancements in this particular field and a lot of research work has been carried out. But more and more work is being done on daily basis to improve its performance, speed, adaptability and overall efficiency of the system [3]. As discussed in chapter 1, digital filters are an essential part and perhaps one of the most important features of the modern day circuit designs so its accuracy plays a vital role in the improvement of the overall system producing better results.

There have been many methods used for the optimization purposes but one of the most efficient ones have been accomplished using an optimization algorithm [3, 4]. Optimization algorithms are easier to execute, cost-effective and can give you results near to real ones. The improvement of the digital filter is hence dependent on a good performing optimization algorithm [6]. There are a number of high performing optimization algorithms that can be classified into different types with each having a certain advantage over the other. The main focus of this thesis is on Firefly optimization algorithm [1, 9] later explained in Chapter 3 which is a nature-inspired, meta-heuristic algorithm and can solve complex mathematical problems with close to ideal results depending on the parameter values set rightly according to the given problem.

There are a number of applications of Firefly algorithm already implemented and compiled in the literature. Important industrial applications of Firefly Algorithm which includes efficient and reliable power production which is necessary to meet both the profitability of power systems operations and the electricity demand, taking also into account the environmental concerns about the emissions produced by fossil-fueled power plants, the economic emission load dispatch problem has been defined and applied in order to deal with the optimization of these two conflicting objectives, that is, the minimization of both fuel cost and emission of generating units [11] introduces and describes a solution to this

famous problem using firefly algorithm which helps to optimize to the optimal results converging in an acceptable time, which for this test system was approximately 3 seconds.

Xin-She Yang used different complex test functions for FA in [12] to prove the efficiency and optimization capabilities of the algorithm. For the standard pressure vessel design optimization, the optimal solution found by FA is far better than the best solution obtained previously in the literature. In addition, a few new test functions with either singularity or stochastic components were also introduced but with known global optimality, and thus they can be used to validate new optimization algorithms. The optimization results imply that the Firefly Algorithm is potentially more powerful than other existing algorithms such as particle swarm optimization such as those described in [13]. X.S.Yang described Firefly algorithm along with other nature-inspired algorithm in [14]. He described the behavior of fireflies, working on the algorithm, its implementation and different modifications to the algorithms [1, 2].

X.Yang in [15] highlighted the importance of exploitation and exploration and their effect on the efficiency of an algorithm. It used the intermittent search strategy theory as a preliminary basis for analyzing these key components and ways to find the possibly optimal settings for algorithm-dependent parameters. It used the firefly algorithm to find this optimal balance and confirmed that firefly algorithm can indeed provide a good balance of exploitation and exploration. It also shows that firefly algorithm requires far fewer function evaluations.

Firefly algorithm has attracted much attention since its development and has been applied to many applications [20, 23, 27, 30, 37, 28,29]. Horng et al. demonstrated that Firefly-based algorithm used least computation time for digital image compression [28, 29], while Banati and Bajaj used firefly algorithm for feature selection and showed that firefly algorithm produced consistent and better performance in terms of time and optimality than other algorithms [21].

In the engineering design problems, Gandomi et al. [26] and Azad and Azad [19] confirmed that firefly algorithm can efficiently solve highly nonlinear, multimodal design problems. Basu and Mahanti [22] as well as Chatterjee et al. have applied FA in antenna design

optimization and showed that FA can outperform artificial bee colony (ABC) algorithm [23]. In addition, Zaman and Matin have also found that FA can outperform PSO and obtained global best results [41]. Sayadi et al. developed a discrete version of FA which can efficiently solve NP-hard scheduling problems [34], while a detailed analysis has demonstrated the efficiency of FA over a wide range of test problems, including multi-objective load dispatch problems [20, 36, 39]. Furthermore, FA can also solve scheduling and traveling salesman problem in a promising way [28, 30, 40].

Classifications and clustering are another important areas of applications of FA with excellent performance [25, 33]. For example, Senthilnath provided an extensive performance study by compared FA with 11 different algorithms and concluded that firefly algorithm can be efficiently used for clustering [35]. In most cases, firefly algorithm outperforms all other 11 algorithms. In addition, firefly algorithm has also been applied to train neural networks [31]. For optimization in dynamic environments, FA can also be very efficient as shown by Farahani et al. [24, 25] and Abshouri [18].

Nature Inspired algorithm is applied with a back-propagation method to train a feed-forward neural network in [25] and it is incorporated into the back-propagation algorithm to achieve fast and improved convergence rate in training feed-forward neural network. A different form of neural network designs is discussed in [43, 25, 44, 42] which shows fast convergence and better results. Even though the results of Firefly algorithm compared to other algorithms have been very efficient in most of the literature present, there is another huge gap between small-scale problems and large-scale problems. As most published studies have focused on small, toy problems, there is no guarantee that the methodology that works well for such toy problems will work for large-scale problems. All these issues still remain unresolved both in theory and in practice. As further research topics, most metaheuristic algorithms [51] require good modifications so as to solve combinatorial optimization properly. Though with great interest and many extensive studies, more studies are highly needed in the area of combinatorial optimization using metaheuristic algorithms [33, 39]. In addition, most current metaheuristic research has focused on small-scale problems, it will be extremely useful if further research can focus on large-scale real-world applications

Various studies show that PSO algorithms can outperform genetic algorithms (GA) and other conventional algorithms for solving many optimization problems. This is partially due to that fact that the broadcasting ability of the current best estimates gives better and quicker convergence towards the optimality. A comparison of the Firefly Algorithms with PSO and genetic algorithms for various standard test functions have been shown in [9]. After implementing these algorithms using MATLAB, extensive simulations have been carried out and each algorithm has been run at least 100 times so as to carry out meaningful statistical analysis. The algorithms stop when the variations of function values are less than a given tolerance $\leq 10^{-5}$. The results are summarized in the following table (see Table 2.1) where the global optima are reached. The numbers are in the format: an average number of evaluations (success rate), so 3752 ± 725(99%) means that the average number (mean) of function evaluations is 3752 with a standard deviation of 725. The success rate of finding the global optima for this algorithm is 99%. We can see that the FA is much more efficient in finding the global optima with higher success rates.

TABLE 2.1 Comparison of algorithm performance for different standard Functions

Functions/Algorithms	**GA**	**PSO**	**FA**
Michalewicz's (d=16)	89325 ± 7914(95%)	6922 ± 537(98%)	3752 ± 725(99%)
Rosenbrock's (d=16)	55723 ± 8901(90%)	32756 ± 5325(98%)	7792 ± 2923(99%)
De Jong's (d=256)	25412 ± 1237(100%)	17040 ± 1123(100%)	7217 ± 730(100%)
Schwefel's (d=128)	227329 ± 7572(95%)	14522 ± 1275(97%)	9902 ± 592(100%)
Ackley's (d=128)	32720 ± 3327(90%)	23407 ± 4325(92%)	5293 ± 4920(100%)
Rastrigin's	110523 ± 5199(77%)	79491 ± 3715(90%)	15573 ± 4399(100%)
Easom's	19239 ± 3307(92%)	17273 ± 2929(90%)	7925 ± 1799(100%)
Griewank's	70925 ± 7652(90%)	55970 ± 4223(92%)	12592 ± 3715(100%)
Shubert's (18 minima)	54077 ± 4997(89%)	23992 ± 3755(92%)	12577 ± 2356(100%)
Yang's (d = 16)	27923 ± 3025(83%)	14116 ± 2949(90%)	7390 ± 2189(100%)

Chapter 3

Firefly Algorithm

3.1 Mathematical model of Firefly algorithm:

Firefly algorithm is organized in a way that it requires the following steps to be set up properly. Not all of these steps are a necessary requirement but helps in implementing the algorithm more efficiently.

1. Set initial parameters in the parameter vector [n iterations α β γ]. Set Upper bound and Lower bound values. (For Example: For a FIR-1 filter, initial values may be set as N=35, iterations=1500, α = 0.25, β = 0.2 and γ=1, Ub=1.5, Lb= -1.5)

2. Generate an initial coefficient vector (say u0). Set the number of coefficients (d) to be optimized.

$$u0 = Lb + (Ub - Lb) * rand(1, d)$$

where $d = \frac{n}{2} + 1$ for FIR type 1 filter (n=24)

3. Inside the algorithm, calculate the number of evaluations and set up a population matrix P with its size equal to

   [number of fireflies, number of filter coefficients to be optimized]

   with random values generated around initial coefficient vector u0.

$$\text{Number of Evaluations} = N * \text{iterations}$$

$$P_{N,d} = u0 + rand(1, d)$$

4. From the population matrix obtained, obtain a fitness value for every coefficient vector. There will be (N x 1) number of fitness values obtained. Sort the fitness values of that vector in the ascending order and store the output in an X matrix.

$$f_i = \text{cost}(P_i)$$

$$X = \text{sort}\,(f_i)$$

5. Obtain f_{best} value from the X vector (which will always be the topmost value after sorting). Compare every value of X with itself (comparing X_i and $X_{j,}$). Calculate the distance 'r_{ij}' for each of the two compared value.

$$f_{best} = X_{N,1}$$

6. Change every coefficient vector from the population P that satisfies $X_i > X_j$ using

$$P_i = P_i + \beta(P_j - P_i) + \alpha * [\text{rand}(1, d) - \tfrac{1}{2}] \quad (3.3)$$

where $\beta = \beta_o e^{-\gamma r^2}$

and $r_{ij} = \|P_i - P_j\| = \sqrt{\sum_{k=1}^{d}(P_{i,k} - P_{j,k})^2}$

7. Check whether the values obtained in the new population are within the range and repeat step 4 until the maximum value of iterations is reached. The coefficient vector providing the f_{best} result is the desired result.

$$\text{If } P_{i,k} > Ub\,,\ P_{i,k} = Ub$$

$$\text{If } P_{i,k} < Lb\,, P_{i,k} = Lb$$

3.2 Background of Firefly algorithm

Nature-inspired metaheuristic algorithms are becoming powerful in solving modern global optimization problems, especially for the NP-hard optimization such as the traveling

salesman problem. For example, particle swarm optimization (PSO) was developed by Kennedy and Eberhard in 1995, based on the swarm behavior such as fish and bird schooling in nature. It has now been applied to find solutions for many optimization applications. Another example is the Firefly Algorithm developed by Xin-She Yang [1] which has demonstrated promising superiority over many other algorithms. The search strategies in these multi-agent algorithms are controlled randomization, efficient local search, and selection of the best solutions. However, the randomization typically uses uniform distribution or Gaussian distribution.

The flashing light of fireflies is an amazing sight in the summer sky in the tropical and temperate regions. There are about two thousand firefly species, and most fireflies produce short and rhythmic flashes. The pattern of flashes is often unique for a particular species. The flashing light is produced by a process of bioluminescence, and the true functions of such signaling systems are still debating. However, two fundamental functions of such flashes are to attract mating partners (communication) and to attract potential prey. In addition, flashing may also serve as a protective warning mechanism. The rhythmic flash, the rate of flashing and the amount of time form part of the signal system that brings both sexes together. Females respond to a male's unique pattern of flashing in the same species, while in some species such as Photuris, female fireflies can mimic the mating flashing pattern of other species so as to lure and eat the male fireflies who may mistake the flashes as a potential suitable mate.

The flashing light can be formulated in such a way that it is associated with the objective function to be optimized, which makes it possible to formulate new optimization algorithms. In the rest of this paper, we will first outline the basic formulation of the Firefly Algorithm (FA) and then discuss the implementation as well as analysis in detail.

Now we can idealize some of the flashing characteristics of fireflies so as to develop firefly-inspired algorithms. For simplicity in describing our Firefly Algorithm (FA), we now use the following three idealized rules:

1) All fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex;

2) Attractiveness is proportional to their brightness, thus for any two flashing fireflies, the less bright one will move towards the brighter one. The attractiveness is proportional to the brightness and they both decrease as their distance increases. If there is no brighter one than a particular firefly, it will move randomly.

3) The brightness of a firefly is affected or determined by the landscape of the objective function. For a maximization problem, the brightness can simply be proportional to the value of the objective function. For a maximization problem, the brightness can simply be proportional to the value of the objective function. Other forms of brightness can be defined in a similar way to the fitness function in genetic algorithms or the bacterial foraging algorithm.

In the firefly algorithm, there are two important issues: the variation of light intensity and formulation of the attractiveness. For simplicity, we can always assume that the attractiveness of a firefly is determined by its brightness which in turn is associated with the encoded objective function.

In the simplest case for maximum optimization problems, the brightness **I** of a firefly at a particular location x can be chosen as $I(x) \propto f(x)$. However, the attractiveness **β** is relative, it should be seen in the eyes of the beholder or judged by the other fireflies. Thus, it will vary with the distance r_{ij} between firefly i and firefly j. In addition, light intensity decreases with the distance from its source, and light is also absorbed in the media, so we should allow the attractiveness to vary with the degree of absorption. In the simplest form, the light intensity $I(r)$ varies according to the inverse square law $I(r) = \frac{I_s}{r^2}$ where I_s is the intensity at the source. For a given medium with a fixed light absorption coefficient γ, the light intensity I vary with the distance r. That is

$$I = I_0 e^{-\gamma r^2} \tag{3.1}$$

where I_0 is the original light intensity.

As a firefly's attractiveness is proportional to the light intensity seen by adjacent fireflies, we can now define the attractiveness β of a firefly by

$$\beta = \beta_0 e^{-\gamma r^2} \tag{3.2}$$

where β_0 is the attractiveness at r = 0.

3.3 Firefly Algorithm Specifications

As the literature of firefly algorithms is rapidly expanding, a natural question is 'why FA is so efficient?'. There are many reasons for its success. By analyzing the main characteristics of the standard/classical FA, we can highlight the following three points:

• FA can automatically subdivide its population into subgroups, due to the fact that local attraction is stronger than long-distance attraction. As a result, FA can deal with highly nonlinear, multi-modal optimization problems naturally and efficiently.

• FA does not use historical individual best, and there is no explicit global best either. This avoids any potential drawbacks of premature convergence as those in PSO. In addition, FA does not use velocities, and there is no problem as that associated with velocity in PSO.

• FA has an ability to control its modality and adapt to problem landscape by controlling its scaling parameter such as γ. In fact, FA is a generalization of SA, PSO, and DE.

It is also notable that the firefly algorithm solves different problems by converting the multi-objective problem to a single-objective problem by a linear combination of different objectives as a weighted sum, while the particle swarm optimization introduces a price penalty factor *h* for the same purpose. Moreover, by using a population of solutions (fireflies) in its search, multiple optimal solutions can be found more quickly, even in one run, in contrast to particle swarm optimization algorithm where each agent (particle) corresponds to one single solution of the problem. Finally, it is important to point out that the firefly algorithm converges in an acceptable time.

In general, the analysis of the experimental results, explained in later sections, has demonstrated that the firefly algorithm performs better than other methods used for the same problem, or at least it obtains good quality optimal solutions in significantly low computing times. It is characterized by a stable and fast convergence compared to other conventional methods and good computation efficiency, as it has been demonstrated by its application. This much-improved speed of computation allows for additional searches and improvements that could be made in order to increase the confidence and efficiency of the generated solutions.

In addition, the standard firefly algorithm can be considered as a generalization to particle swarm optimization (PSO), differential evolution (DE), and simulated annealing (SA).

- From Eq. (3.3), we can see that when β_0 is zero, the updating formula becomes essentially a version of parallel simulated annealing, and the annealing schedule is controlled by α.

- On the other hand, if we set $\gamma = 0$ in Eq. (3.3) and set $\beta_0 = 1$, FA becomes a simplified version of differential evolution without mutation, and the crossover rate is controlled by β_0.

- Furthermore, if we set $\gamma = 0$ and replace P_j with the current global best solution g^*, then Eq. (3.3) becomes a variant of PSO, or accelerated particle swarm optimization, to be more specific.

Therefore, the standard firefly algorithm includes DE, PSO, and SA as its special cases. As a result, FA can have all the advantages of these three algorithms. Consequently, it is no surprise that FA can perform very efficiently.

Chapter 4
Neural Networks

4.1 Introduction

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

Neural networks (or connectionist systems) are a computational model used in computer science and other research disciplines, which is based on a large collection of simple neural units (artificial neurons), loosely analogous to the observed behavior of a biological brain's axons to solve problems in the same way that the human brain would.

A neural network is typically defined by three types of parameters:

- The interconnection pattern between the different layers of neurons
- The weights of the interconnections, which are updated in the learning process.
- The activation function that converts a neuron's weighted input to its output activation.

An interesting fact of these systems is that they are unpredictable in their success with self-learning. After training, some become great problem solvers and others don't perform as well. In order to train them, several thousand cycles of interaction typically occur. Like other machine learning methods, systems that learn from data, neural networks have been

used to solve a wide variety of tasks, like computer vision and speech recognition, that are hard to solve using ordinary rule-based programming.

Backpropagation is a common method of training artificial neural networks and used in conjunction with an optimization method such as gradient descent. The algorithm repeats a two-phase cycle, propagation and weight update. When an input vector is presented to the network, it is propagated forward through the network, layer by layer, until it reaches the output layer. The output of the network is then compared to the desired output, using a cost function, and an error value is calculated for each of the neurons in the output layer. The error values are then propagated backward, starting from the output, until each neuron has an associated error value which roughly represents its contribution to the original output.

Backpropagation uses these error values to calculate the gradient of the cost function with respect to the weights in the network. In the second phase, this gradient is fed to the optimization method, which in turn uses it to update the weights, in an attempt to minimize the cost function.

The importance of this process is that, as the network is trained, the neurons in the intermediate layers organize themselves in such a way that the different neurons learn to recognize different characteristics of the total input space. After training, when an arbitrary input pattern is present which contains noise or is incomplete, neurons in the hidden layer of the network will respond with an active output if the new input contains a pattern that resembles a feature that the individual neurons have learned to recognize during their training.

4.1.1 Historical background

Neural network simulations appear to be a recent development. However, this field was established before the advent of computers and has survived at least one major setback and several eras.

Many important advances have been boosted by the use of inexpensive computer emulations. Following an initial period of enthusiasm, the field survived a period of

frustration and disrepute. During this period when funding and professional support was minimal, important advances were made by relatively few researchers. These pioneers were able to develop convincing technology which surpassed the limitations identified by Minsky and Papert. Minsky and Papert published a book (in 1969) in which they summed up a general feeling of frustration (against neural networks) among researchers, and was thus accepted by most without further analysis. Currently, the neural network field enjoys a resurgence of interest and a corresponding increase in funding.

The first artificial neuron was produced in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pits. But the technology available at that time did not allow them to do too much.

4.1.2 Why use neural networks?

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest and answer "what if" questions. Other advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.

2. Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.

3. Real-Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.

4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

4.1.3 Neural networks versus conventional computers

Neural networks take a different approach to problem-solving than that of conventional computers. Conventional computers use 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 cannot solve the problem. That restricts the problem-solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don't exactly know how to do.

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

On the other hand, conventional computers use a cognitive approach to problem-solving; the way the problem is to solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high-level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach to arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

Much is still unknown about how the brain trains itself to process information, so theories abound. In the human brain, a typical neuron collects signals from others through a host of fine structures called dendrites. The neuron sends out spikes of electrical activity through

a long, thin stand known as an axon, which splits into thousands of branches. At the end of each branch, a structure called a synapse converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another change.

4.1.4 Feed-forward networks

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straightforward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

4.1.5 Feedback networks

Feedback networks can have signals traveling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations.

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "input" units is connected to a layer of "hidden" units, which is connected to a layer of "output" units.

- The activity of the input units represents the raw information that is fed into the network.
- The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.

- The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

We also distinguish single-layer and multi-layer architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organizations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

The memorization of patterns and the subsequent response of the network can be categorized into two general paradigms:

Associative mapping in which the network learns to produce a particular pattern on the set of input units whenever another particular pattern is applied to the set of input units. The associative mapping can generally be broken down into two mechanisms:

- *Auto-association*: an input pattern is associated with itself and the states of input and output units coincide. This is used to provide pattern completion, i.e. to produce a pattern whenever a portion of it or a distorted pattern is presented. In the second case, the network actually stores pairs of patterns building an association between two sets of patterns.
- *hetero-association*: is related to two recall mechanisms:

    - *nearest-neighbor* recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented, and

    - *Interpolative* recall, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented. Yet another paradigm, which is a variant associative mapping is

classification, ie when there is a fixed set of categories into which the input patterns are to be classified.

Regularity detection in which units learn to respond to particular properties of the input patterns. Whereas in associative mapping the network stores the relationships among patterns, in regularity detection the response of each unit has a particular 'meaning'. This type of learning mechanism is essential for feature discovery and knowledge representation.

Every neural network possesses knowledge which is contained in the values of the connections weights. Modifying the knowledge stored in the network as a function of experience implies a learning rule for changing the values of the weights.

Information is stored in the weight matrix W of a neural network. Learning is the determination of the weights. Following the way learning is performed, we can distinguish two major categories of neural networks:

- ***Fixed networks*** in which the weights cannot be changed, ie dW/dt=0. In such networks, the weights are fixed a priori according to the problem to solve.

- ***Adaptive networks*** which are able to change their weights, ie dW/dt not= 0.

All learning methods used for adaptive neural networks can be classified into two major categories:

- **Supervised learning** which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process, global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning, and stochastic learning.

  An important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error. One

well-known method, which is common to many learning paradigms is the least mean square (LMS) convergence.

- **Unsupervised learning** uses no external teacher and is based on only local information. It is also referred to as self-organization, in the sense that it self-organizes 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 a separate phase, during which the network is trained, and a subsequent operation phase. We say that a neural network learns off-line if the learning phase and the operation phase are distinct. A neural network learns on-line if it learns and operates at the same time. Usually, supervised learning is performed off-line, whereas unsupervised learning is performed on-line.

4.2 Neural Networks in Practice

Given this description of neural networks and how they work, what real-world applications are they suited for? Neural networks have broad applicability to real-world business problems. In fact, they have already been successfully applied in many industries.

Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including:

- sales forecasting
- industrial process control
- customer research
- data validation
- risk management

But to give you some more specific examples; ANN is also used in the following specific paradigms: recognition of speakers in communications; diagnosis of hepatitis; recovery of telecommunications from faulty software; interpretation of multi-meaning words; undersea

mine detection; texture analysis; three-dimensional object recognition; hand-written word recognition; and facial recognition.

The computing world has a lot to gain from neural networks. Their ability to learn by example makes them very flexible and powerful. Furthermore, there is no need to devise an algorithm in order to perform a specific task; i.e. there is no need to understand the internal mechanisms of that task. They are also very well suited for real-time systems because of their fast response and computational times which are due to their parallel architecture.

Neural networks also contribute to other areas of research such as neurology and psychology. They are regularly used to model parts of living organisms and to investigate the internal mechanisms of the brain.

Perhaps the most exciting aspect of neural networks is the possibility that someday 'conscious' networks might be produced. There is a number of scientists arguing that consciousness is a 'mechanical' property and that 'conscious' neural networks are a realistic possibility. Even though neural networks have a huge potential we will only get the best of them when they are integrated with computing, AI, fuzzy logic and related subjects.

4.3 XOR Neural Network Problem

One of the common backpropagation problems that can be solved using neural networks is the Exclusive-OR problem 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 4.1. 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.

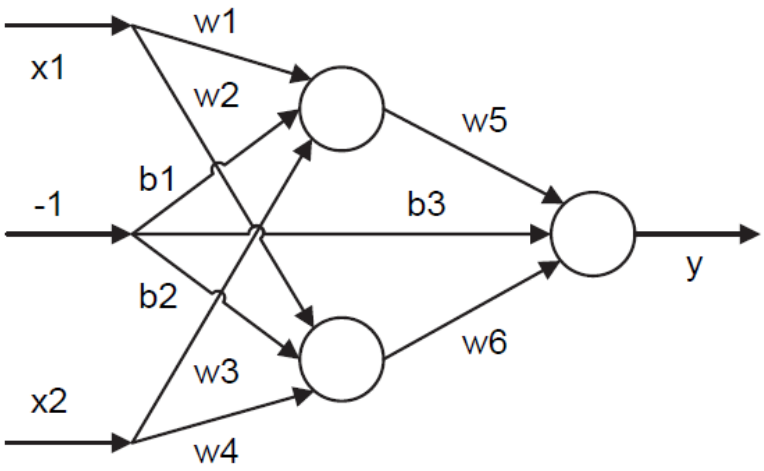


Figure 4.1 Neural network design for two input XOR problem. (Kwan [73], 2016, pg 18)

Results of the coefficient vector for the parameters of two-layer neural network is given in Table 4.1 and 4.2.

TABLE 4.1 2-input one output Neural network design parameters

Parameters	**Obtained values**
W1	6.804825634099212
W2	6.652658888362717
W3	-6.574292782777339
W4	-6.874863726533375
W5	-20
W6	20
B1	-3.345630235052796
B2	3.382454430968793
B3	-9.994892142234166

TABLE 4.2 2-input one output Neural network design

Iterations	**Time Elapsed (Sec)**	**Best Cost Value**
1000	10.536342	9.761402898298260e-18

To verify the results obtained produce an 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 results are shown in Figures 4.2 and 4.3.

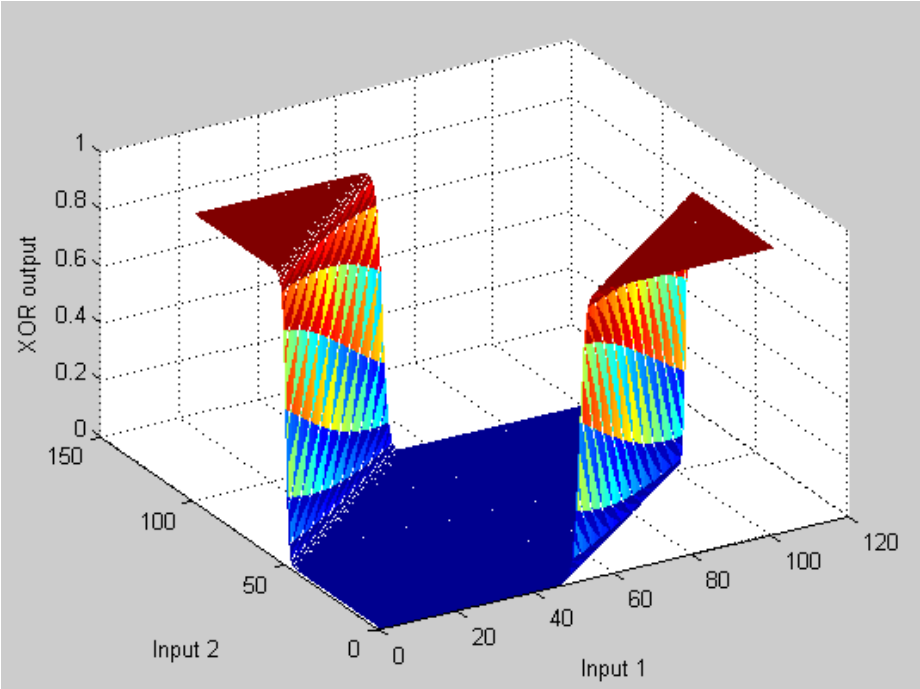


Figure 4.2 XOR output with 2 input grid

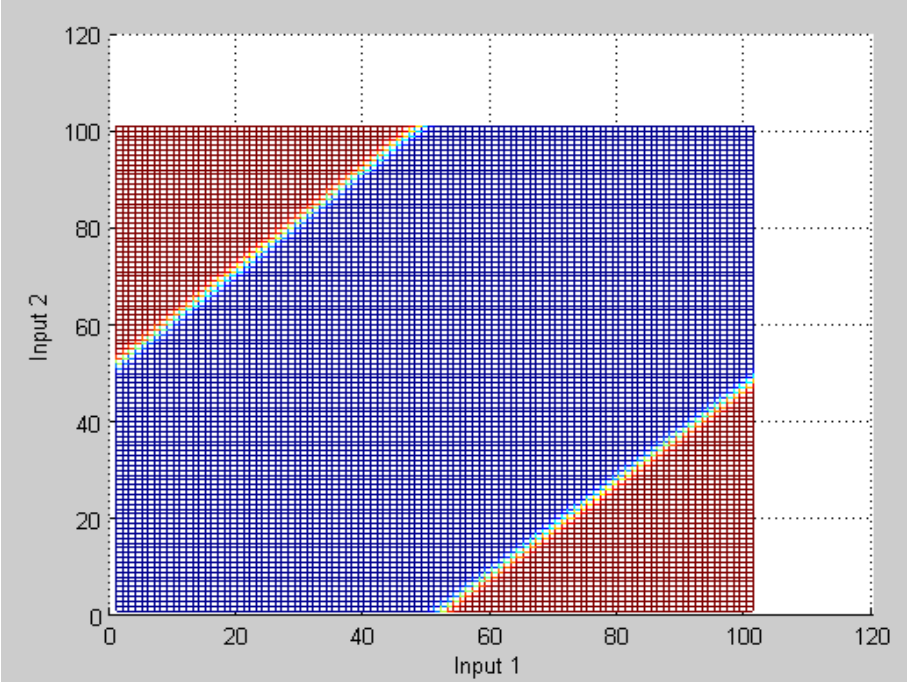


Figure 4.3 2-dimensional view of Figure 4.2

4.4 Advanced feedforward Neural Network Problem

This is a simple design for an advanced feedforward neural network problem using the simplified sigmoid function as defined in H. K. Kwan and C. Z. Tang's paper [8] The idea is to simplify a more complex neural network problem with a large number of inputs and weighing function values in order to produce predefined output results. In this particular problem, 100 bits are taken as input which may be -1 or +1. There are 10 hidden neurons and 4 output neurons. 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:

$$y_{jk}^{[h]} = \mathrm{F}\left(\sum_{i=1}^{N_{h-1}} w_{ij}^{[h]}\, y_{ik}^{[h-1]} + b_j^{[h]}\right) \tag{4.1}$$

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

In eqn. 4.1, $y_{ik}^{[h-1]}$ is the output of a neuron i at the layer $h-1$; $w_{ij}^{[h]}$ is the weight between a neuron i at the layer $h-1$ and a neuron j at layer h; $b_j^{[h]}$ is the bias of a neuron j at the layer h; N_h is the number of neurons at layer h.

The input pattern as defined in [8] for the neural network problem have ten virtual digits each of which is defined by a 10 by 10 grid. So, each number contains 100 input values to determine the output of one digit. The desired training pattern is shown in Figure 4.4.

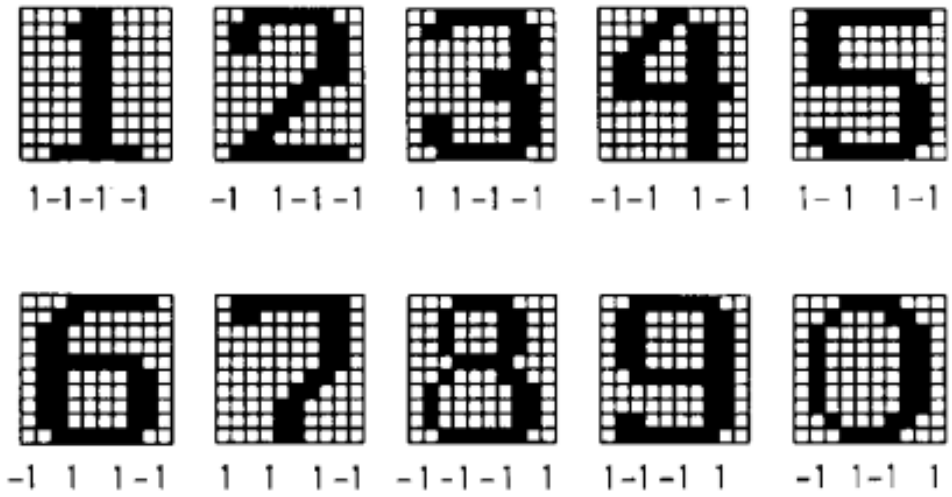


Figure 4.4 Training pattern pairs (Kwan[8],1992, pg 2)

The output response defined in Figure 4.4 is given by the four output neurons. Each digit has hundred input values which make 100 w^1 values to be optimized. 10 digits mean that there are 1000 w^1 values needing to be optimized. The ten neuron hidden neural network layer links to the 4 output neuron with 40 w^2 values to be optimized. That makes it 1040 $w_{ij}^{[h]}$ values to be optimized. Also each neuron has a bias value $b_j^{[h]}$ which helps in shifting the neuron output to the desired sigmoid value. There are total 14 bias values corresponding to each of the 14 neurons. Therefore, a total of 1054 parameters need to be optimized in order to train the neural network system. We have used FA to optimize all the parameters such that when a distorted input is received, the neural network should be able to identify and produce the desired results. The desired and obtained output of the neural network after training the parameters with FA can be seen in Figure 4.5.

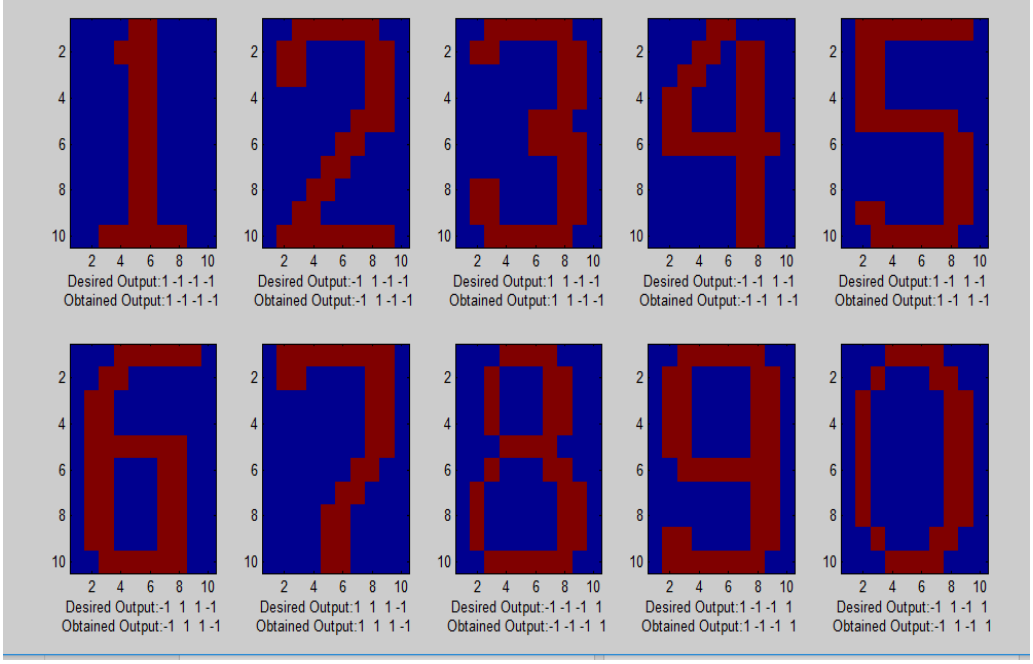


Figure 4.5 Ten digit problem output pattern.

The design specifications and results are shown in Table 4.3

TABLE 4.3 Results table with no input noise.

DIGITS	RESULT	4 – OUTPUT NEURONS				TOTAL ERROR	TIME ELAPSED (Sec)
One	Desired	1	-1	-1	-1	1.3948e-07	28.137744
	Obtained	1	-1	-1	-1		
Two	Desired	-1	1	-1	-1		
	Obtained	-1	1	-1	-1		
Three	Desired	1	1	-1	-1		
	Obtained	1	1	-1	-1		
Four	Desired	-1	-1	1	-1		
	Obtained	-1	-1	1	-1		
Five	Desired	1	-1	1	-1		
	Obtained	1	-1	1	-1		
Six	Desired	-1	1	1	-1		
	Obtained	-1	1	1	-1		
Seven	Desired	1	1	1	-1		
	Obtained	1	1	1	-1		
Eight	Desired	-1	-1	-1	1		
	Obtained	-1	-1	-1	1		
Nine	Desired	1	-1	-1	1		
	Obtained	1	-1	-1	1		
Zero	Desired	-1	1	-1	1		
	Obtained	-1	1	-1	1		

It can be seen from Figure 4.5 that the neural network returns exactly the same desired response and the response time of the training of 1054 parameters using FA is very quick considering the number of parameters needing to be optimized. The same problem is repeated with 5%, 10% and 20% of input noise in order to check if the same response can be obtained with the already trained parameters. The results with error values are shown in Table 4.4 – 4.6 respectively.

TABLE 4.4 Results table with 5% input noise.

DIGITS	RESULT	4 – OUTPUT NEURONS				TOTAL ERROR	TIME ELAPSED (Sec)
One	Desired	1	-1	-1	-1		
	Obtained	1	-1	-0.6	-1		
Two	Desired	-1	1	-1	-1		
	Obtained	-0.2	1	1	-1		
Three	Desired	1	1	-1	-1		
	Obtained	1	1	-1	-1		
Four	Desired	-1	-1	1	-1		
	Obtained	-1	-1	1	-1		
Five	Desired	1	-1	1	-1		
	Obtained	1	-1	1	-1		
Six	Desired	-1	1	1	-1		
	Obtained	-1	1	1	-1		
Seven	Desired	1	1	1	-1		
	Obtained	1	1	1	-1		
Eight	Desired	-1	-1	-1	1	0.081378	27.2567
	Obtained	-1	-1	-1	1		
Nine	Desired	1	-1	-1	1		
	Obtained	1	-1	-1	1		
Zero	Desired	-1	1	-1	1		
	Obtained	-1	1	-1	1		

TABLE 4.5 Results table with 10% input noise.

DIGITS	RESULT	4 – OUTPUT NEURONS				TOTAL ERROR	TIME ELAPSED (Sec)
One	Desired	1	-1	-1	-1		
	Obtained	1	-1	-1	-1		
Two	Desired	-1	1	-1	-1		
	Obtained	-1	1	1	-1		
Three	Desired	1	1	-1	-1		
	Obtained	1	1	1	-1		
Four	Desired	-1	-1	1	-1	0.15603	29.4525
	Obtained	-1	-1	1	-1		

Five	Desired	1	-1	1	-1		
	Obtained	1	-1	0.5	-1		
Six	Desired	-1	1	1	-1		
	Obtained	-1	1	0.65	-1		
Seven	Desired	1	1	1	-1		
	Obtained	1	1	1	-1		
Eight	Desired	-1	-1	-1	1		
	Obtained	-1	-1	-1	1		
Nine	Desired	1	-1	-1	1		
	Obtained	1	-1	-1	1		
Zero	Desired	-1	1	-1	1		
	Obtained	-1	1	-1	1		

TABLE 4.6 Results table with 20% input noise.

DIGITS	**RESULT**	**4 – OUTPUT NEURONS**				**TOTAL ERROR**	**TIME ELAPSED (Sec)**
One	Desired	1	-1	-1	-1	0.26998	27.3941
	Obtained	1	-1	1	-1		
Two	Desired	-1	1	-1	-1		
	Obtained	-1	1	-1	-1		
Three	Desired	1	1	-1	-1		
	Obtained	1	1	-0.3	-1		
Four	Desired	-1	-1	1	-1		
	Obtained	-1	-0.9	1	-1		
Five	Desired	1	-1	1	-1		
	Obtained	1	-1	1	-1		
Six	Desired	-1	1	1	-1		
	Obtained	-1	1	1	-1		
Seven	Desired	1	1	1	-1		
	Obtained	1	1	1	-1		
Eight	Desired	-1	-1	-1	1		
	Obtained	-1	1	-1	1		
Nine	Desired	1	-1	-1	1		
	Obtained	1	-1	1	1		
Zero	Desired	-1	1	-1	1		
	Obtained	-1	1	1	-1		

It can be seen from the results tables that the error values increase with the increasing noise in the input pattern. But once the parameters are trained it can point to the output response really fast with the minimum error possible which makes neural network designs very effective where fast processing is needed. The training is also very fast with FA and gives accurate results which make FA a very effective algorithm to work with neural network designs.

Chapter 5
Results

For simulation of filter designing using Firefly algorithm, FIR type 1 filter was chosen as it is suitable for implementing all 4 basic types (Lowpass, Highpass, Bandpass, Bandstop) of filters. 2 different filter orders (24 & 48 orders) were chosen to show the complexity handling of the algorithm and the filters were designed and compared with the results of DE and PM. The simulation of the aforementioned designed and their analysis is given in this chapter.

This chapter gives a complete picture of how the algorithm performs with different designs and specifications and how much time does it take to produce the results. It is divided into several sections. The LP-FIR designs for 24 order are given in 5.1. All the designs are then compared to DE results provided earlier in chapter 2 in section 5.2. The results are then compared with state-of-the-art designs of PM algorithm in section 5.3. Similarly, the 48^{th} order LP-FIR designs are provided in section 5.4. The 48^{th} order filter designs of FA are then compared with DE and PM in section 5.5 and 5.6 respectively. The General FIR design results are presented in section 5.7.

5.1 FIR 24 order Filter Results Obtained Using FA

For Type 1 LP-FIR filter of order 24, Filter designs using FA are given below:

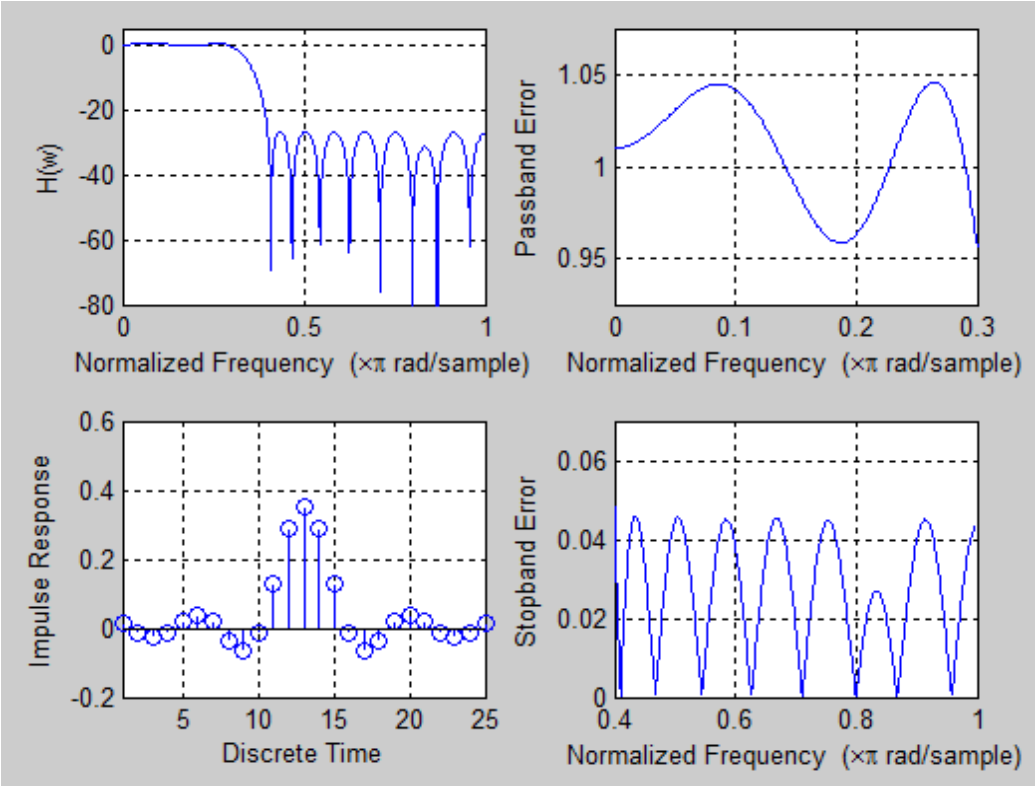


Figure 5.1 24 order Lowpass digital FIR filter using FA

TABLE 5.1 Coefficients of 24th-order type1 Lowpass LP-FIR filter by FA

h(n)	Coefficients	h(n)	Coefficients
h(1) = h(25)	-0.015522008933653	h(8) = h(18)	0.041237502939032
h(2) = h(24)	0.016486080444651	h(9) = h(17)	0.072079031708125
h(3) = h(23)	0.026248621788878	h(10) = h(16)	0.017485734410618
h(4) = h(22)	0.011646728361479	h(11) = h(15)	-0.122801217633660
h(5) = h(21)	-0.018764773645612	h(12) = h(14)	-0.281650153767307
h(6) = h(20)	-0.037309926526026	h(13) =	-0.345426609420225
h(7) = h(19)	-0.015878216920711		

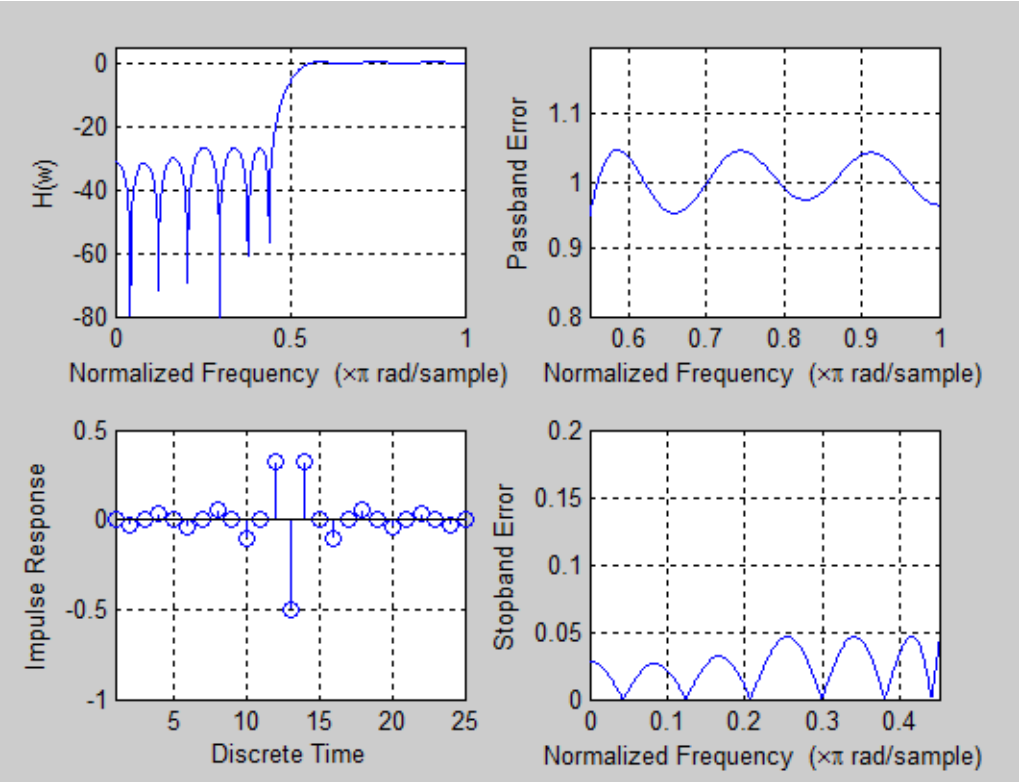


Figure 5.2 24 order Highpass digital FIR filter using FA

TABLE 5.2 Coefficients of 24th-order type1 Highpass LP-FIR filter by FA

h(n)	Coefficients	h(n)	Coefficients
h(1) = h(25)	0.000260047828761	h(8) = h(18)	-0.057430962743755
h(2) = h(24)	0.030529809103280	h(9) = h(17)	-0.002419357765453
h(3) = h(23)	-0.000429243385549	h(10) = h(16)	0.101126670449230
h(4) = h(22)	-0.027462082232009	h(11) = h(15)	-0.000640410712477
h(5) = h(21)	-0.002089836305828	h(12) = h(14)	-0.319934972869165
h(6) = h(20)	0.037157890650607	h(13) =	0.500420471134895
h(7) = h(19)	-0.003287213256156		

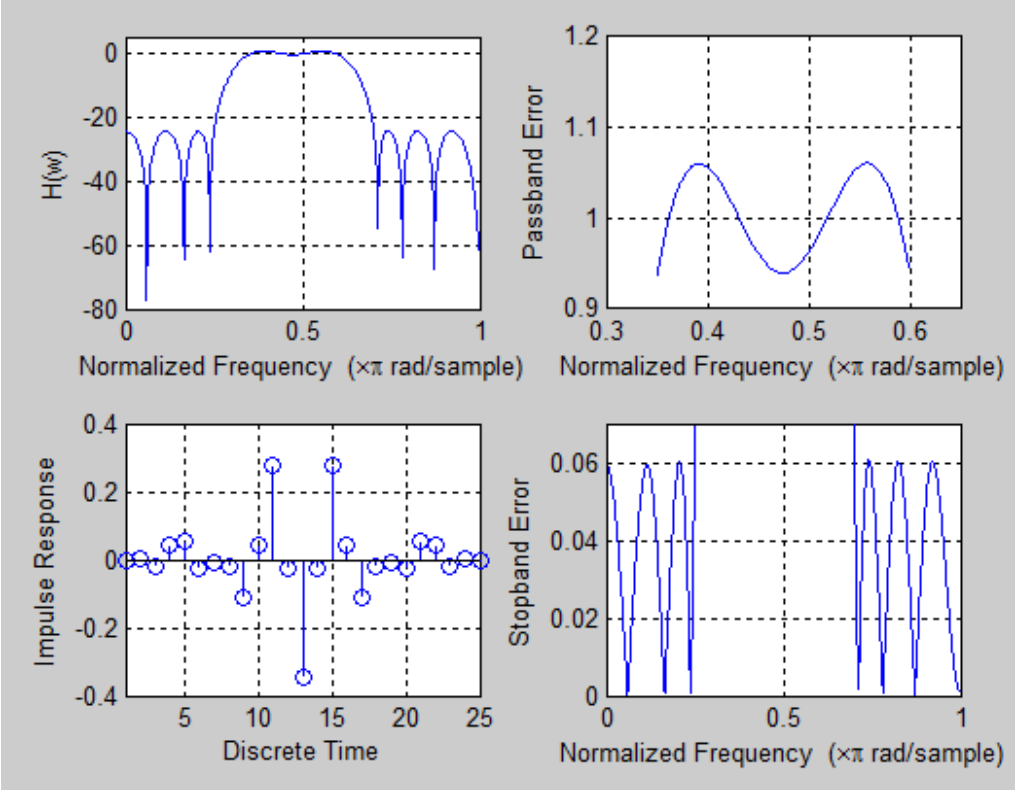


Figure 5.3 24 order Bandpass digital FIR filter using FA

TABLE 5.3 Coefficients of 24th-order type1 Bandpass LP-FIR filter by FA

h(n)	Coefficients	h(n)	Coefficients
h(1) = h(25)	0.006539698675755	h(8) = h(18)	0.022251374087707
h(2) = h(24)	-0.000883807845255	h(9) = h(17)	0.119648078224730
h(3) = h(23)	0.025203886110461	h(10) = h(16)	-0.040381104590304
h(4) = h(22)	-0.039184475398611	h(11) = h(15)	-0.270395031717470
h(5) = h(21)	-0.050330891221388	h(12) = h(14)	0.032305553853011
h(6) = h(20)	0.028866262629929	h(13) =	0.355094279682638
h(7) = h(19)	0.016178791697218		

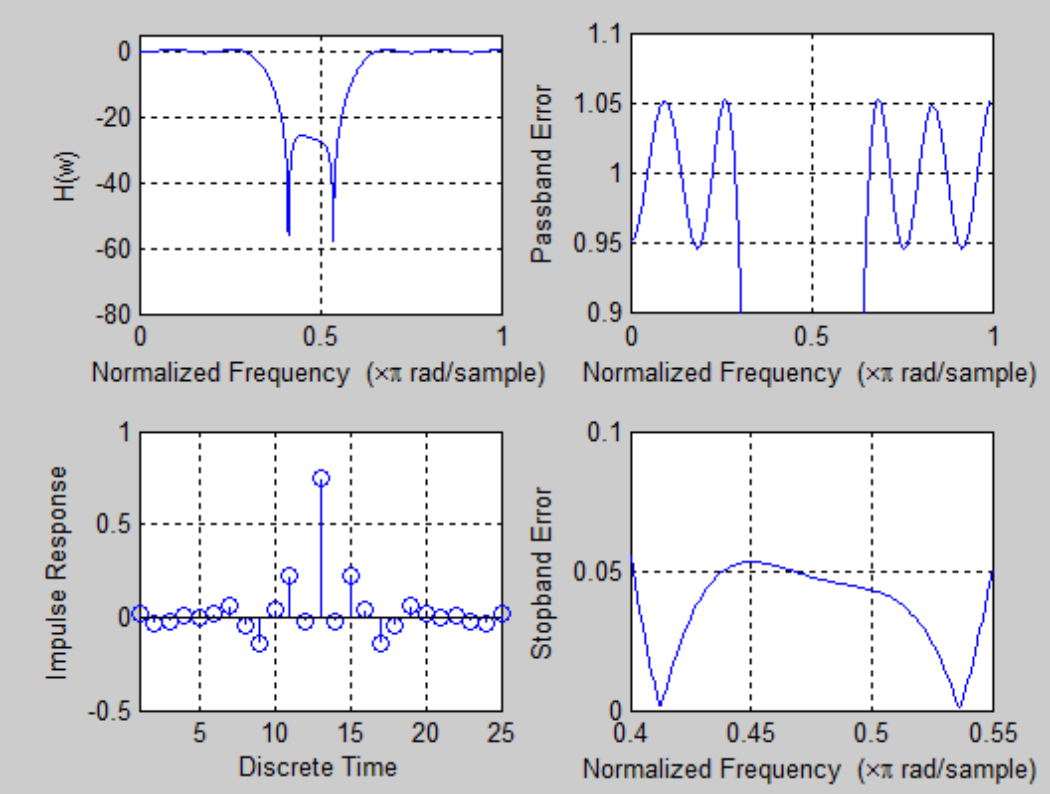


Figure 5.4 24 order Bandstop digital FIR filter using FA

TABLE 5.4 Coefficients of 24th-order type1 Bandstop LP-FIR filter by FA

h(n)	Coefficients	h(n)	Coefficients
h(1) = h(25)	-0.017929329230994	h(8) = h(18)	0.039262844253155
h(2) = h(24)	0.030979466221750	h(9) = h(17)	0.144555430625010
h(3) = h(23)	0.021042245597871	h(10) = h(16)	-0.044486440743630
h(4) = h(22)	-0.010160321400097	h(11) = h(15)	-0.223730643349791
h(5) = h(21)	-0.000173841974654	h(12) = h(14)	0.019371489733680
h(6) = h(20)	-0.018582630214610	h(13) =	-0.743730083562852
h(7) = h(19)	-0.062225263304964		

5.2 Comparison of DE and FA results (Order 24)

For 24 order filter designs, Digital filters are designed using FA as seen in 5.1. Also for comparing the results, same filters were also designed using Differential Evolution algorithm in order to judge the difference in performances of the two algorithms. The designs of filters using DE can be seen in 2.3 from Fig. 2.1-2.4. Visual comparison of the two algorithms is given in figure 5.5 - 5.8.

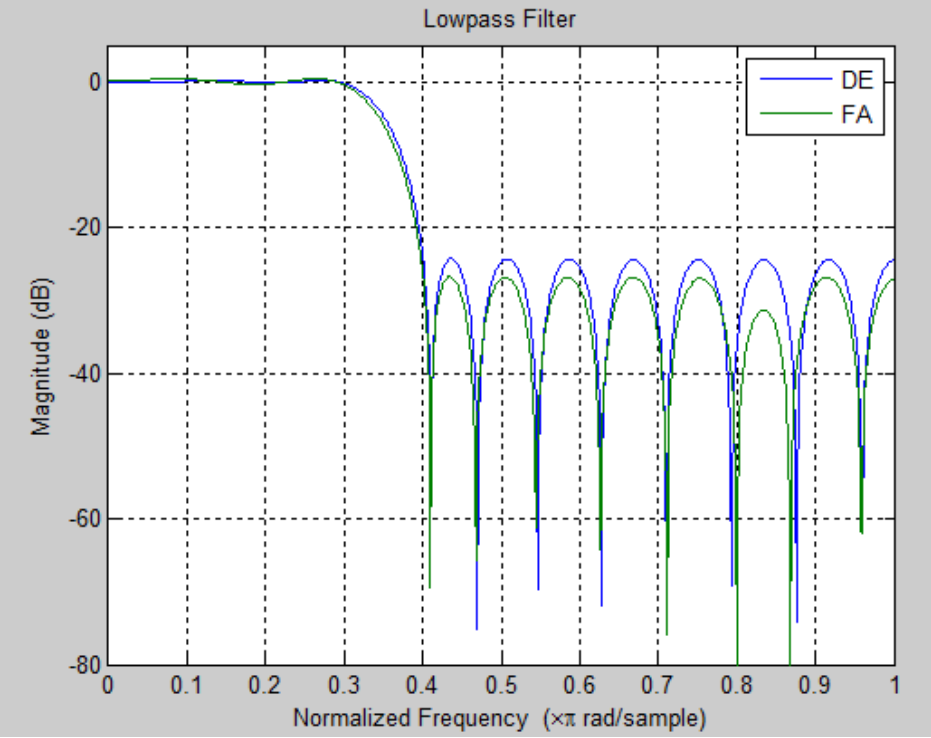


Figure 5.5 Lowpass FIR filter comparing FA and DE

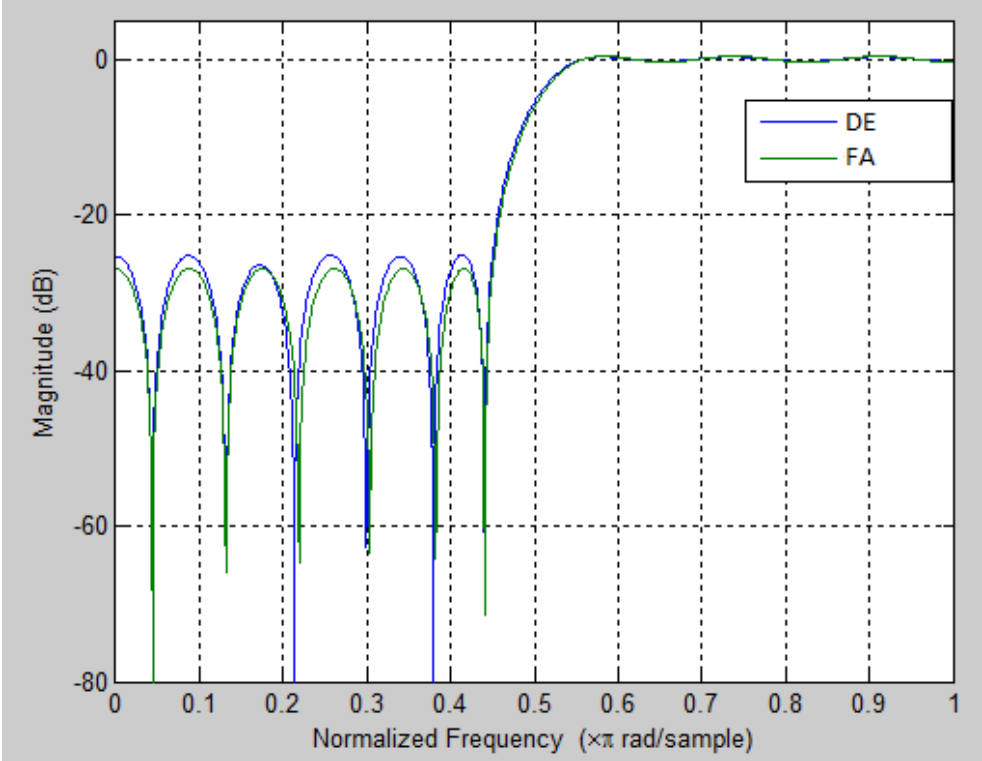


Figure 5.6 Highpass FIR filter comparing FA and DE

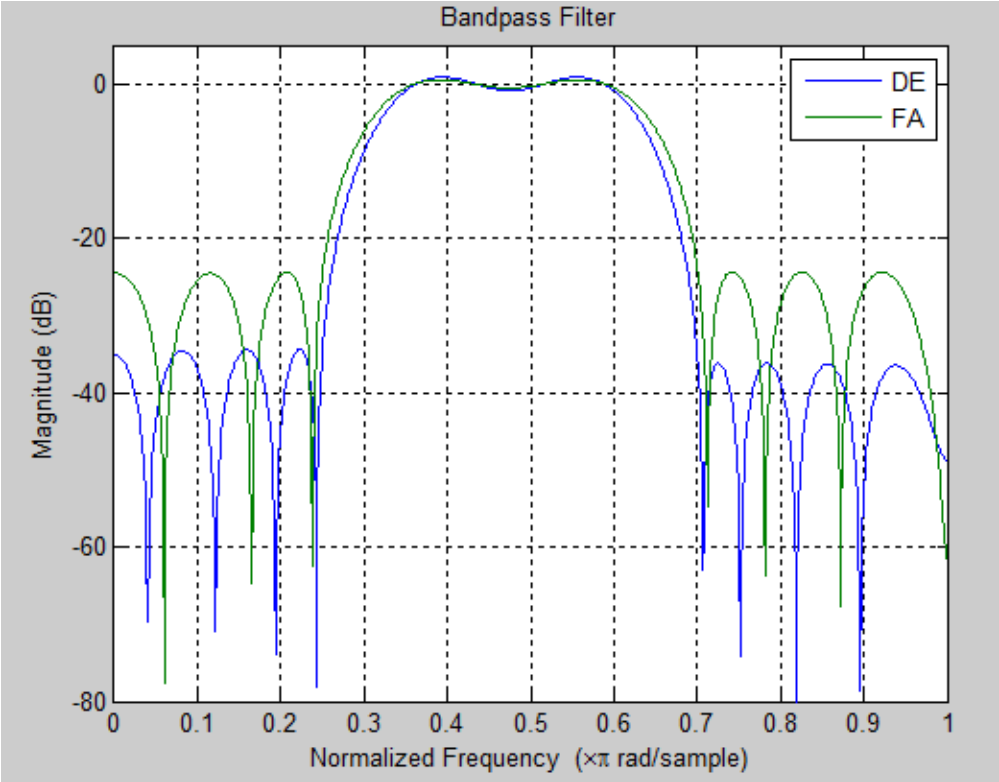


Figure 5.7 Bandpass FIR filter comparing FA and DE

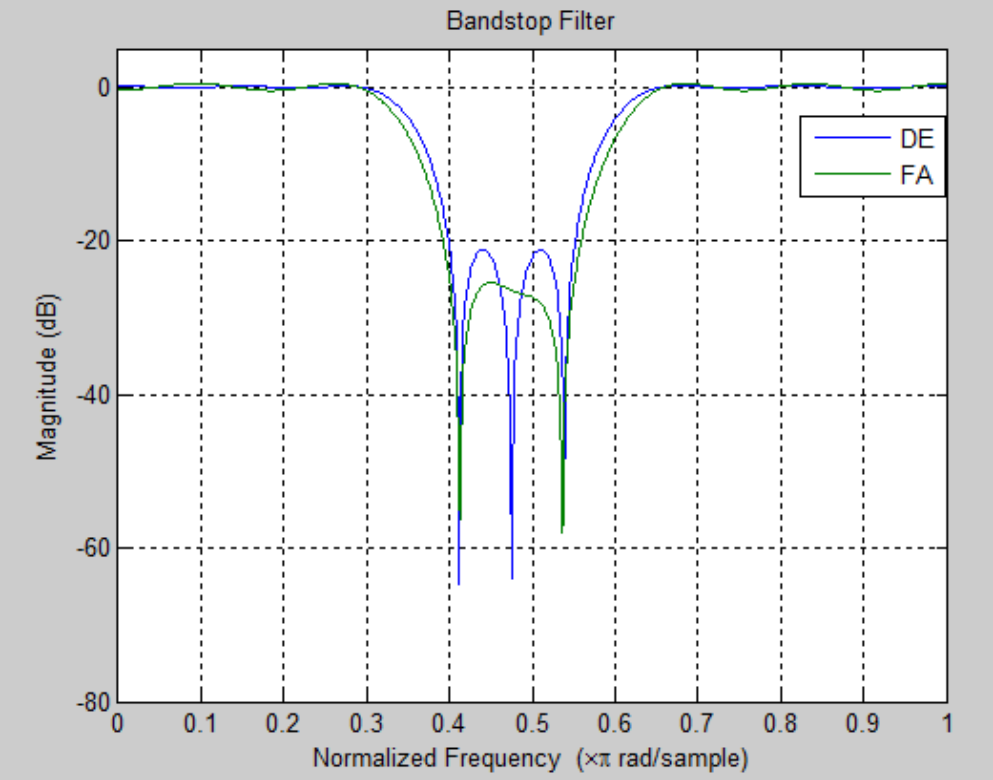


Figure 5.8 Bandstop FIR filter comparing FA and DE

TABLE 5.5 24 order FIR type 1 filter design results comparison (DE: Differential Evolution; FA: Firefly Algorithm)

Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU (Sec)**
Lowpass	FA	0.048261126877512	0.045898650619239	-	61.06
	DE	0.057892831435627	0.024816996495112	-	5905
Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU (Sec)**
Highpass	FA	0.046743317585723	0.050000576593838	-	51.66
	DE	0.052408544937316	0.038951269285244	-	7106
Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU (Sec)**
Bandpass	FA	0.058934863943933	0.062612456660805	0.06168617624984	83.96
	DE	0.018159709479349	0.103707424193301	0.01815970947934	11117
Filter	**Alg**	**Peak(Passband)**	**Peak(Stopband)**	**Peak(Passband 2)**	**CPU (Sec)**
Bandstop	FA	0.051684607201277	0.055622602909391	0.05769081459274	56.19
	DE	0.014716165596515	0.090720149556089	0.01471616559651	7447

5.3 FA results compared with FIRPM (24 order)

For 24 order filter designs, Digital filters designs using FA were also compared with the state-of-the-art designs of Parks McClellan algorithm. It can be seen from the figure 5.9-5.12 that the results of FA and PM are almost identical.

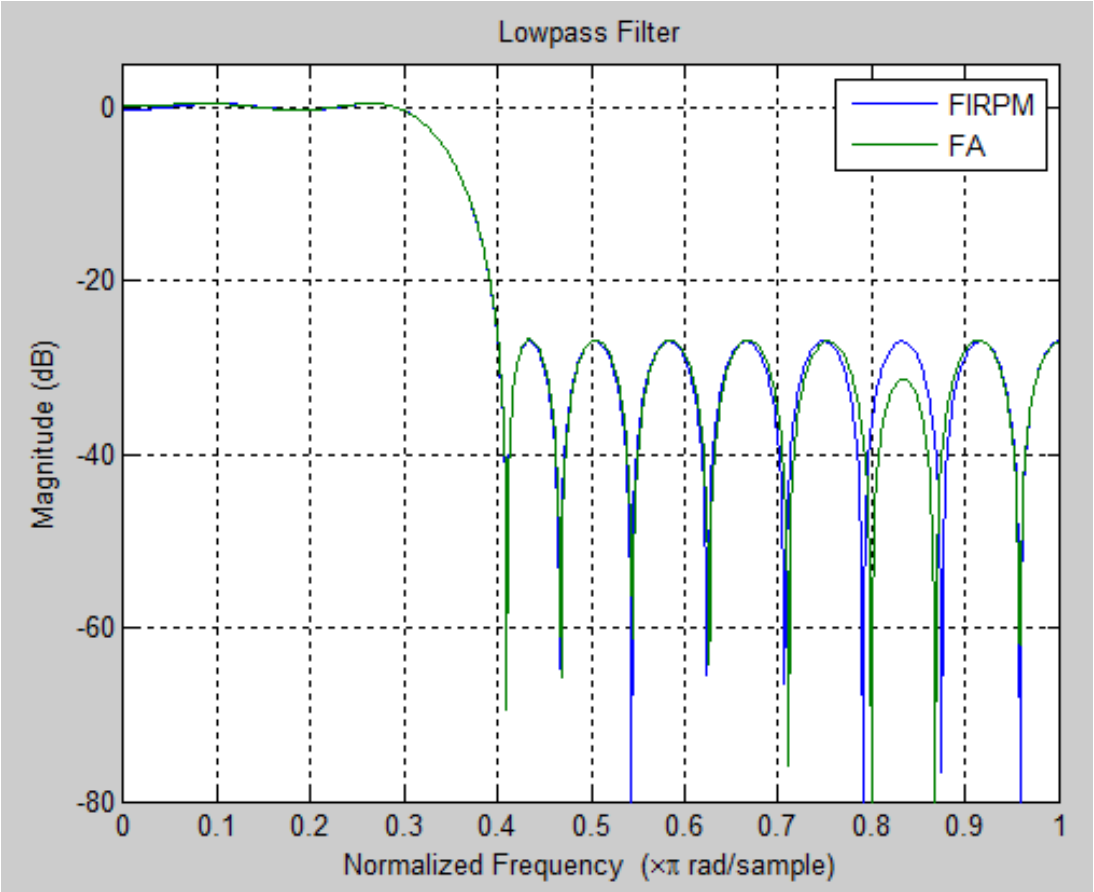


Figure 5.9 Lowpass FIR filter comparing FA and FIRPM

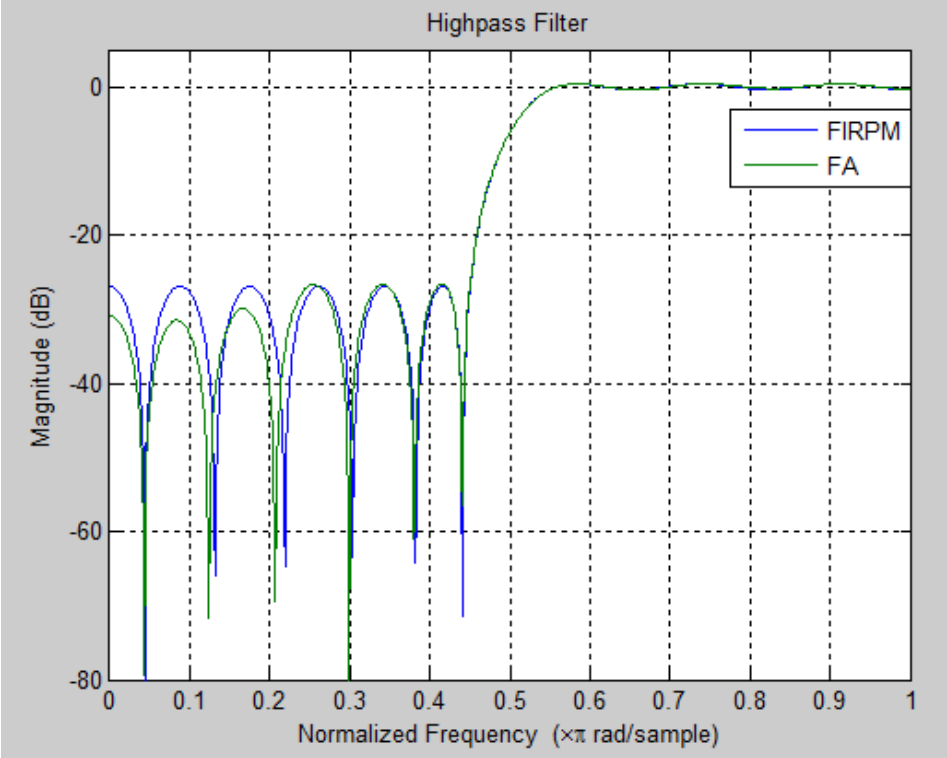


Figure 5.10 Highpass FIR filter comparing FA and FIRPM

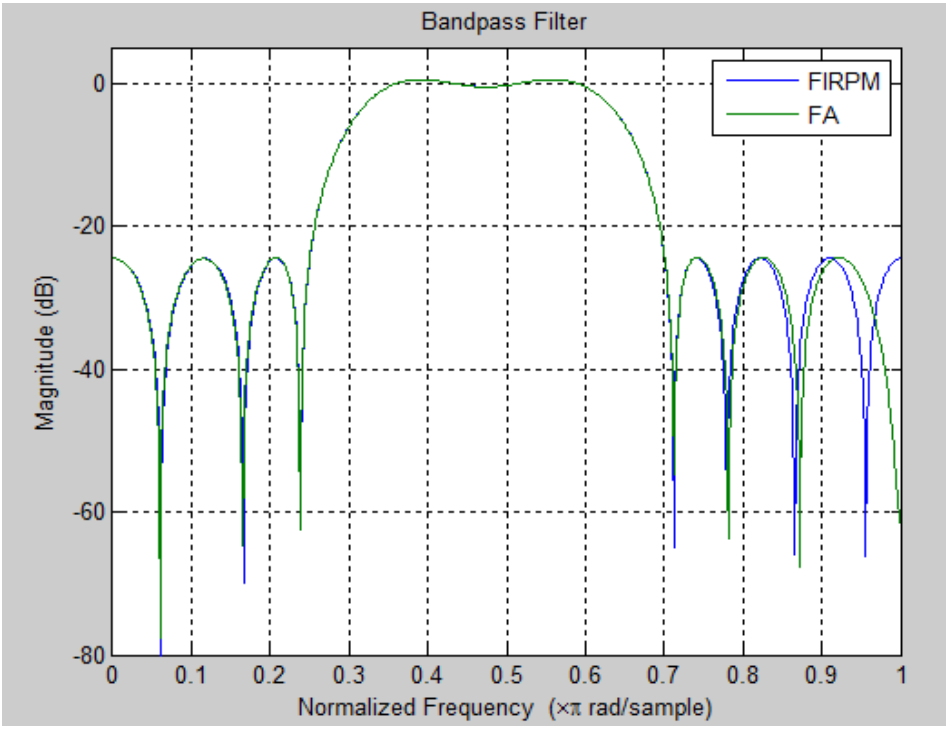


Figure 5.11 Bandpass FIR filter comparing FA and FIRPM

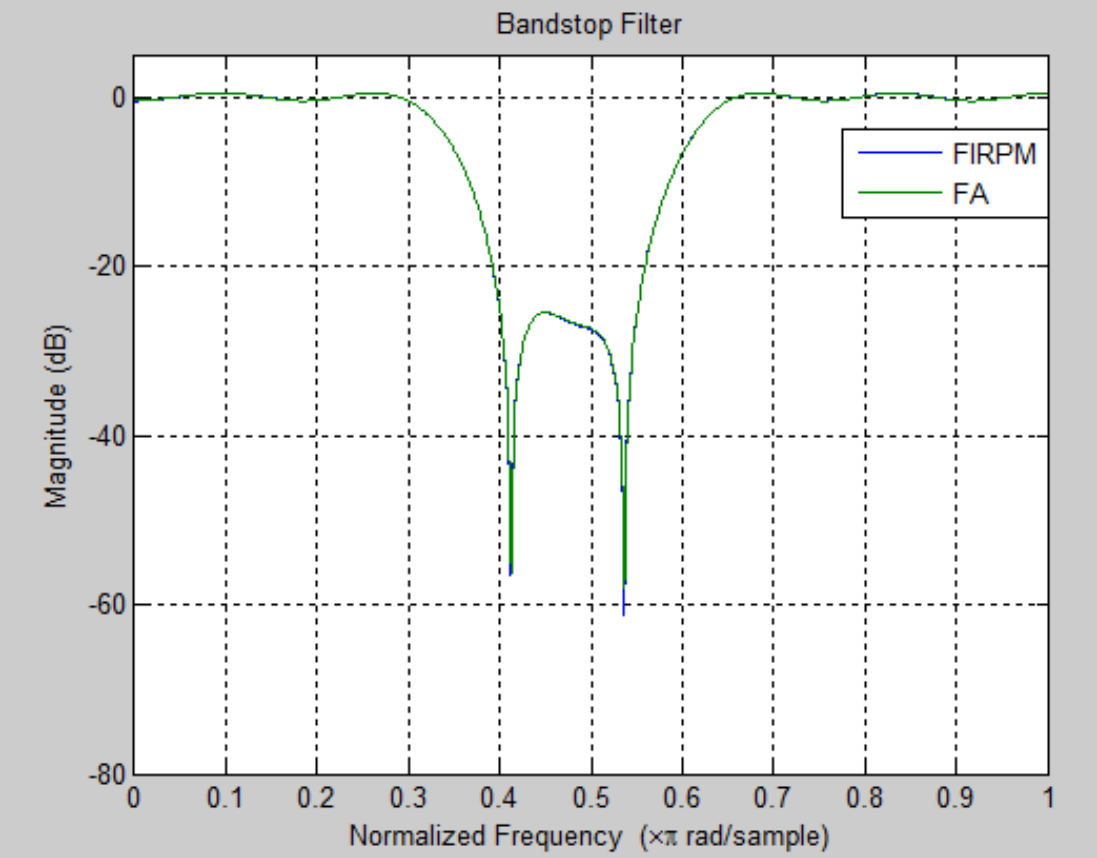


Figure 5.12 Bandstop FIR filter comparing FA and FIRPM

An overall result of all four LP-FIR digital filters for order 24 obtained by using Firefly algorithm compared to both Differential Evolution algorithm and Parks–McClellan algorithm can be shown in Table 5.6.

TABLE 5.6 24 order FIR type 1 filter design results comparison (FA: Firefly Algorithm, FIRPM: Parks–McClellan algorithm; DE: Differential Evolution Algorithm)

Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU (Sec)**
Lowpass	PM	0.04680441780656	0.044663349407852	-	0.049
	DE	0.05789283143562	0.024816996495112	-	5905
	FA	0.04826112687751	0.045898650619239	-	61.06
Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU (Sec)**
Highpass	PM	0.04527382939466	0.048082904678732	-	0.1836
	DE	0.05240854493731	0.038951269285244	-	7106
	FA	0.04674331758572	0.050000576593838	-	51.66
Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU (Sec)**
Bandpass	PM	0.05657992127561	0.061630490482057	0.063817460882562	0.1822
	DE	0.01815970947934	0.103707424193301	0.018159709479349	11117
	FA	0.05893486394393	0.062612456660805	0.061686176249849	83.96
Filter	**Alg**	**Peak(Passband)**	**Peak(Stopband)**	**Peak(Passband 2)**	**CPU (Sec)**
Bandstop	PM	0.05313202024494	0.055140706326915	0.057141822013960	0.1904
	DE	0.01471616559651	0.090720149556089	0.014716165596515	7447
	FA	0.05168460720127	0.055622602909391	0.05769081459274	56.19

5.4 FIR 48 order Filter Results Obtained Using FA

For Type 1 LP-FIR filter of order 48, Filter designs using FA are given below:

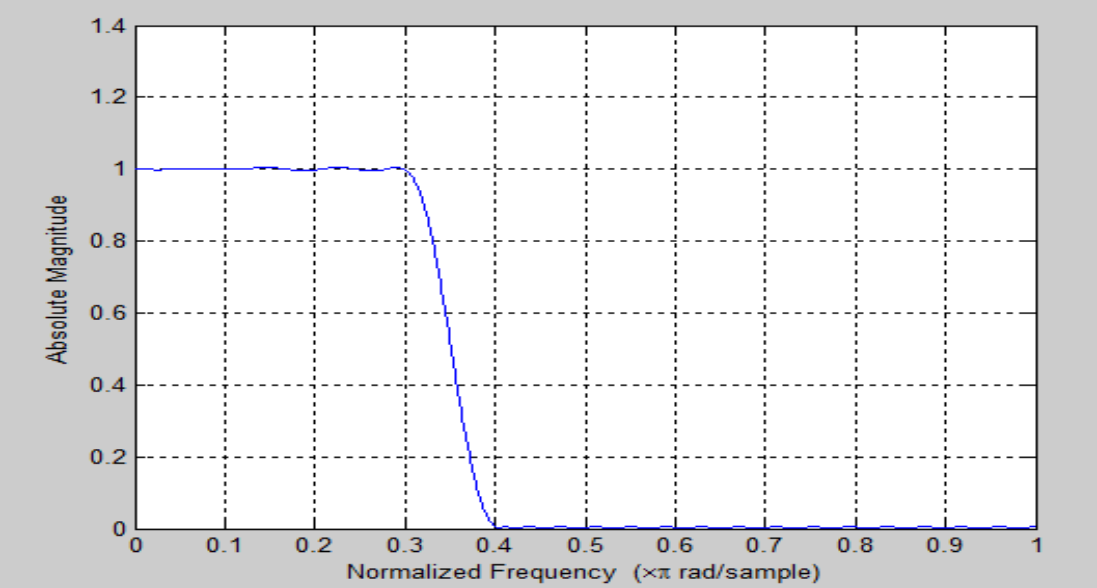


Figure 5.13 Magnitude response of 48 order Lowpass digital FIR filter using FA

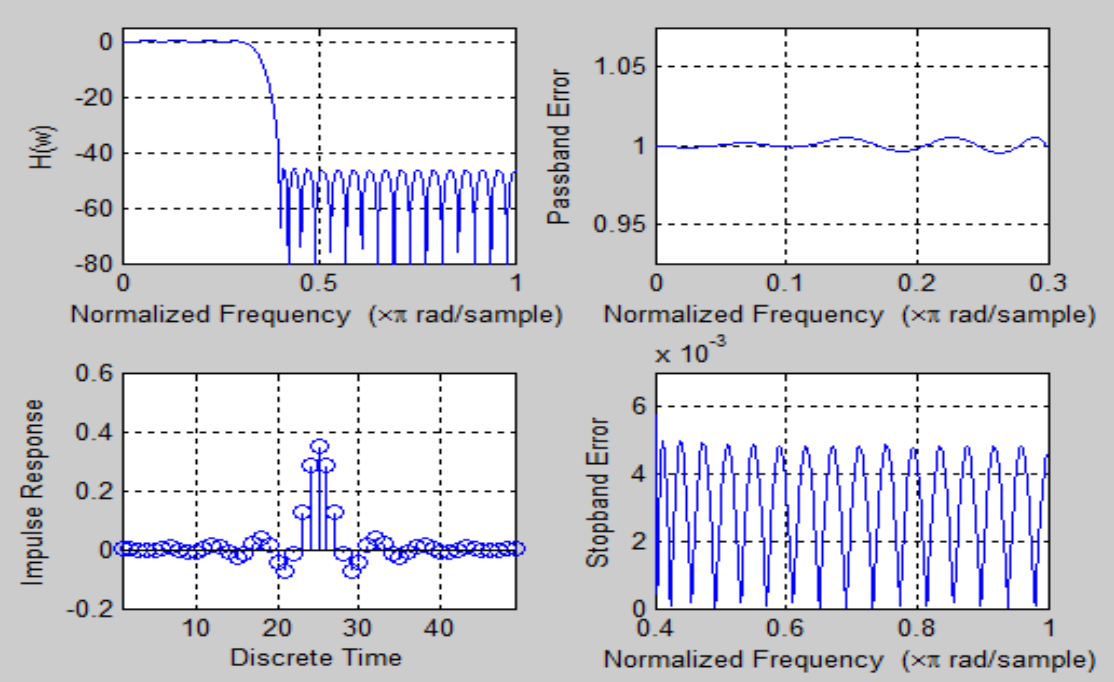


Figure 5.14 48 order Lowpass digital FIR filter using FA

TABLE 5.7 Coefficients of 48th-order type1 Lowpass LP-FIR filter by FA

h(n)	Coefficients	h(n)	Coefficients
h(1) = h(49)	0.003156991481139	h(14) = h(36)	-0.009002611563698
h(2) = h(48)	0.000068595895858	h(15) = h(35)	-0.024724309036358
h(3) = h(47)	-0.003020249400455	h(16) = h(34)	-0.013882774516647
h(4) = h(46)	-0.004101377999152	h(17) = h(33)	0.019097832049613
h(5) = h(45)	-0.000432163819737	h(18) = h(32)	0.039719977020496
h(6) = h(44)	0.005252127754140	h(19) = h(31)	0.015708843665374
h(7) = h(43)	0.006133847108435	h(20) = h(30)	-0.041730093575953
h(8) = h(42)	-0.000817695074935	h(21) = h(29)	-0.073077087480593
h(9) = h(41)	-0.009297338042015	h(22) = h(28)	-0.017168425910954
h(10) = h(40)	-0.008597220424668	h(23) = h(27)	0.126968291502862
h(11) = h(39)	0.003638781917081	h(24) = h(26)	0.283447617450385
h(12) = h(38)	0.015476767707981	h(25) =	0.351071736112283
h(13) = h(37)	0.011282505001528		

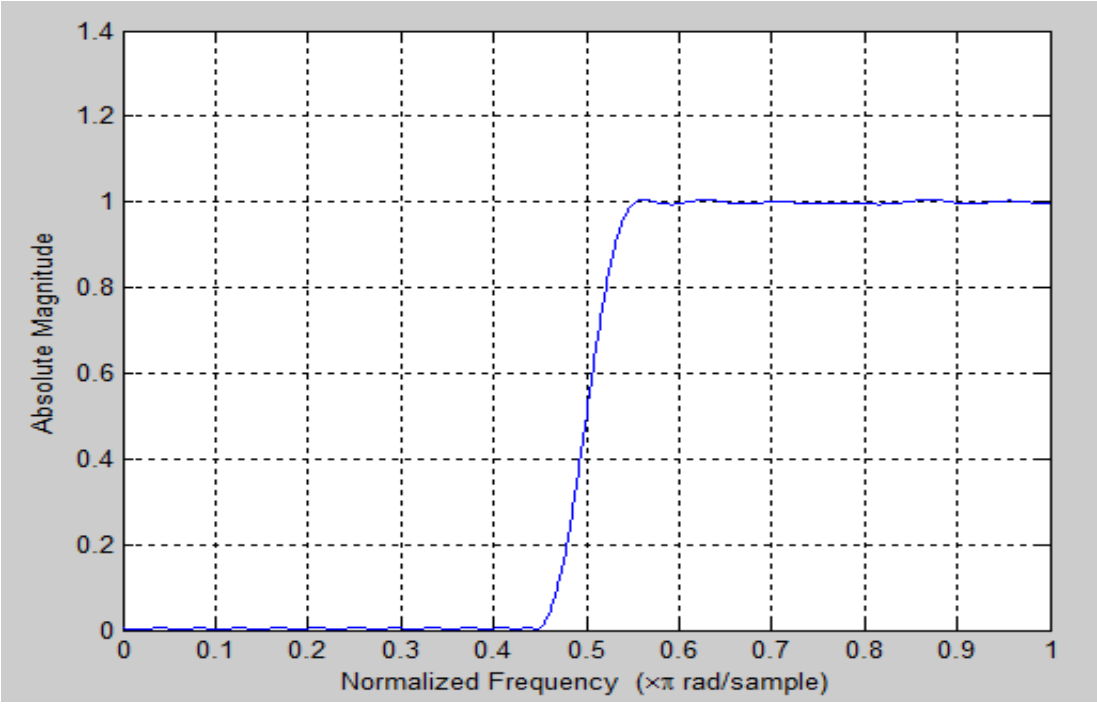


Figure 5.15 Magnitude response of 48 order Lowpass digital FIR filter using FA

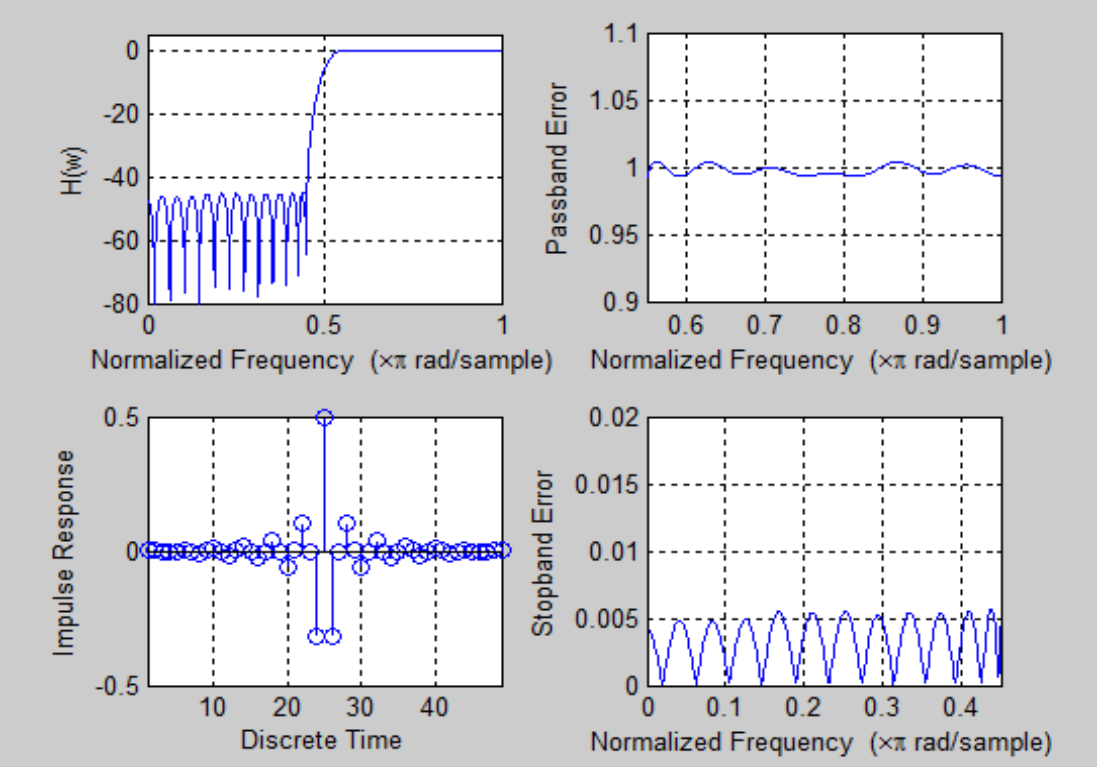


Figure 5.16 48 order Highpass digital FIR filter using FA

TABLE 5.8 Coefficients of 48th-order type1 Highpass LP-FIR filter by FA

h(n)	Coefficients	h(n)	Coefficients
h(1) = h(49)	0.000517758207302	h(14) = h(36)	0.020920507913662
h(2) = h(48)	0.003539481313068	h(15) = h(35)	-0.000523206771986
h(3) = h(47)	-0.000051994567141	h(16) = h(34)	-0.028173261913607
h(4) = h(46)	-0.003803752599296	h(17) = h(33)	-0.000075755845606
h(5) = h(45)	-0.000296720795616	h(18) = h(32)	0.040186029220737
h(6) = h(44)	0.005998974325234	h(19) = h(31)	-0.000338910621310
h(7) = h(43)	-0.000312713136642	h(20) = h(30)	-0.059916443717716
h(8) = h(42)	-0.008273365392330	h(21) = h(29)	0.000661620629984
h(9) = h(41)	0.000381884059713	h(22) = h(28)	0.103404844582466
h(10) = h(40)	0.011108103085080	h(23) = h(27)	-0.000428354594165
h(11) = h(39)	-0.000053897962200	h(24) = h(26)	-0.317116051339188
h(12) = h(38)	-0.015541505004602	h(25) =	0.499866153122865
h(13) = h(37)	0.000343275007539		

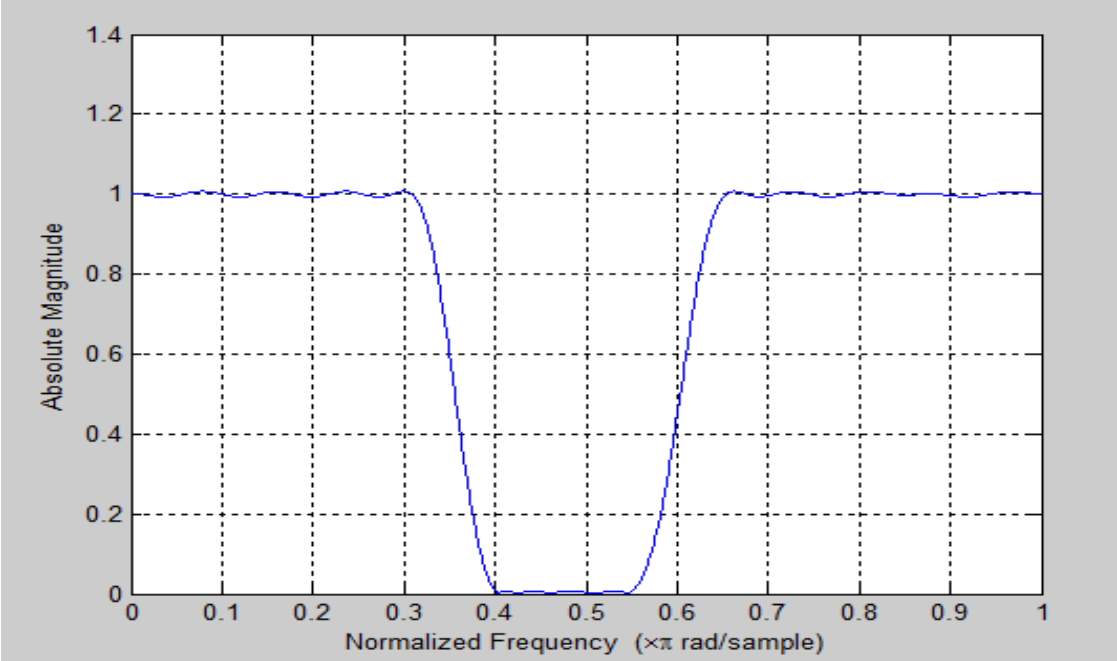


Figure 5.17 Magnitude response of 48 order Bandstop digital FIR filter using FA

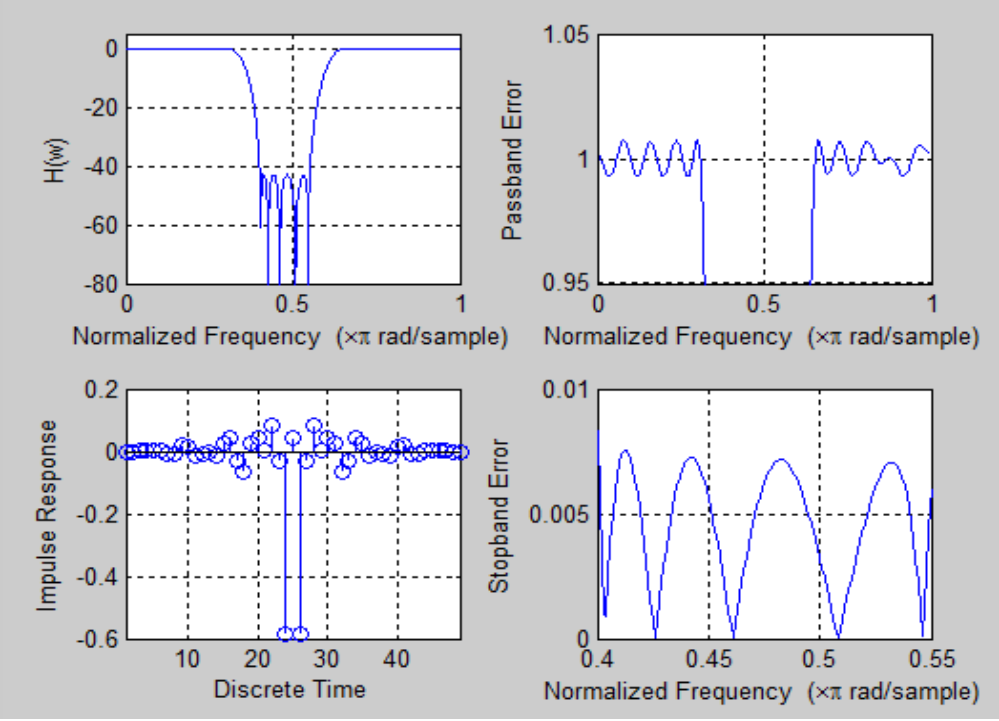


Figure 5.18 48 order Bandstop digital FIR filter using FA

TABLE 5.9 Coefficients of 48th-order type1 Bandstop LP-FIR filter by FA

h(n)	Coefficients	h(n)	Coefficients
h(1) = h(49)	-0.005378250063156	h(14) = h(36)	-0.013456511708727
h(2) = h(48)	-0.001921766427042	h(15) = h(35)	0.023661396150739
h(3) = h(47)	0.004717293141307	h(16) = h(34)	0.044828577667078
h(4) = h(46)	0.002029385360111	h(17) = h(33)	-0.033921696344892
h(5) = h(45)	0.001336172352747	h(18) = h(32)	-0.066087823911229
h(6) = h(44)	0.000803948846018	h(19) = h(31)	0.025405138893122
h(7) = h(43)	-0.011023484273117	h(20) = h(30)	0.041044183255582
h(8) = h(42)	-0.006600621836804	h(21) = h(29)	0.001488011978820
h(9) = h(41)	0.017842886607902	h(22) = h(28)	0.080755418812471
h(10) = h(40)	0.012335177537555	h(23) = h(27)	-0.029827622307303
h(11) = h(39)	-0.013662772245499	h(24) = h(26)	-0.586625924604684
h(12) = h(38)	-0.008025998718848	h(25) =	0.042338175594954
h(13) = h(37)	-0.001751130090847		

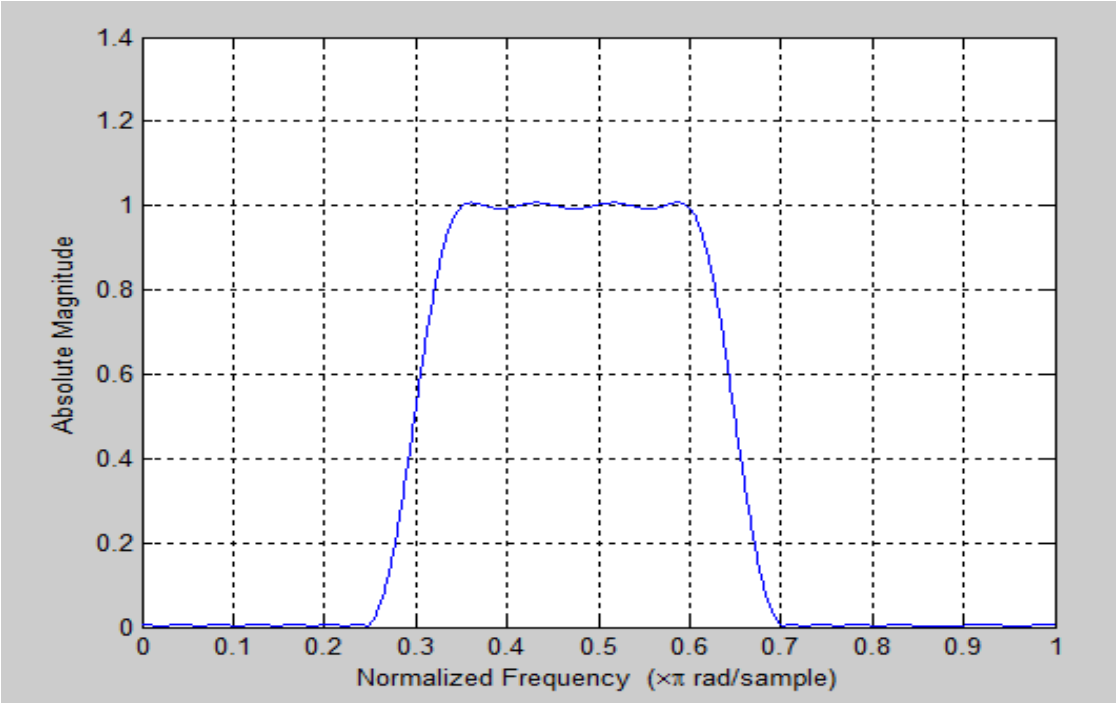


Figure 5.19 Magnitude response of 48 order Bandpass digital FIR filter using FA

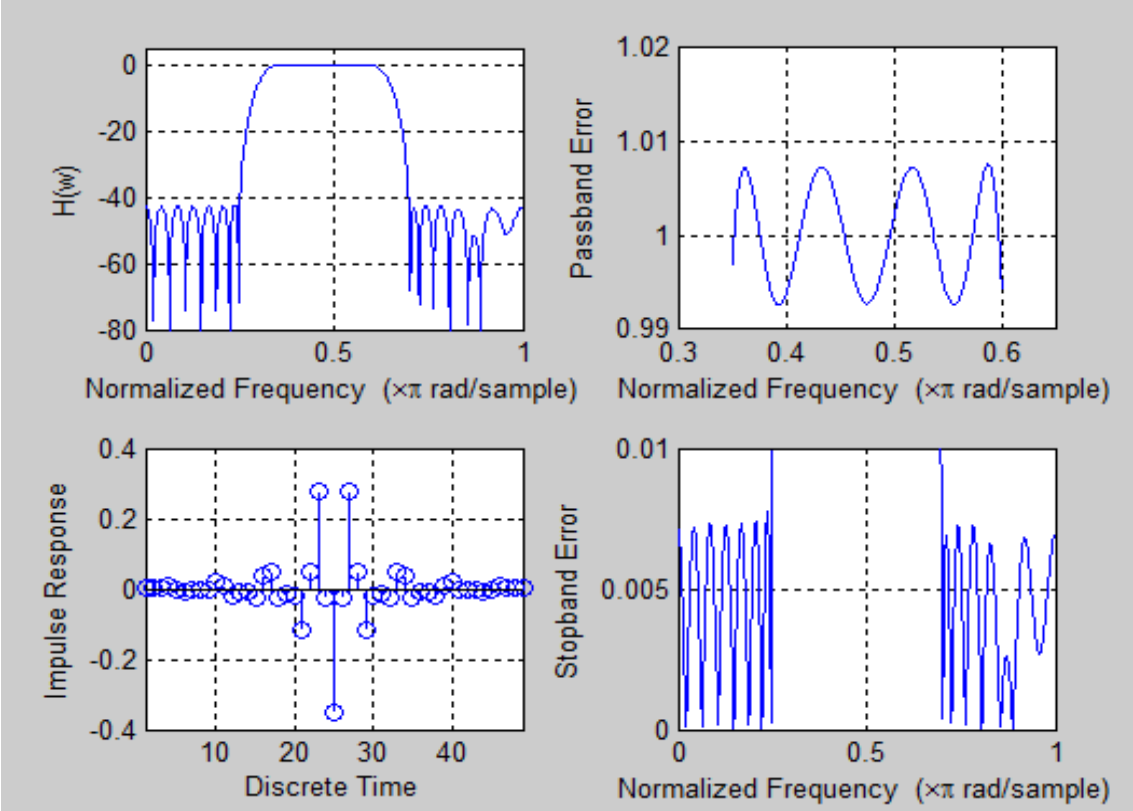


Figure 5.20 8 order Bandpass digital FIR filter using FA

TABLE 5.10 Coefficients of 48th-order type1 Bandpass LP-FIR filter by FA

h(n)	Coefficients	h(n)	Coefficients
h(1) = h(49)	0.000786646366987	h(14) = h(36)	-0.007530040362122
h(2) = h(48)	0.001093911308246	h(15) = h(35)	-0.023034701024358
h(3) = h(47)	0.000869133481166	h(16) = h(34)	0.037419157533026
h(4) = h(46)	0.007005821482122	h(17) = h(33)	0.050245240248314
h(5) = h(45)	-0.000588993480590	h(18) = h(32)	-0.028383086351011
h(6) = h(44)	-0.010127401441240	h(19) = h(31)	-0.013627050573068
h(7) = h(43)	-0.000220571990600	h(20) = h(30)	-0.017994123819306
h(8) = h(42)	-0.001222649435483	h(21) = h(29)	-0.116928651804409
h(9) = h(41)	-0.002568195063428	h(22) = h(28)	0.051200919993218
h(10) = h(40)	0.019460623234224	h(23) = h(27)	0.276415577757441
h(11) = h(39)	0.010201399517217	h(24) = h(26)	-0.026888590456298
h(12) = h(38)	-0.020518581869875	h(25) =	-0.350820669057934
h(13) = h(37)	-0.006094767653371		

5.5 Comparison of DE and FA results (Order 48)

For 48 order filter designs, Digital filters designs using FA were also compared with the Differential Evolution filter designs. The comparison be seen from the figure 5.21-5.25 and from Table 5.11

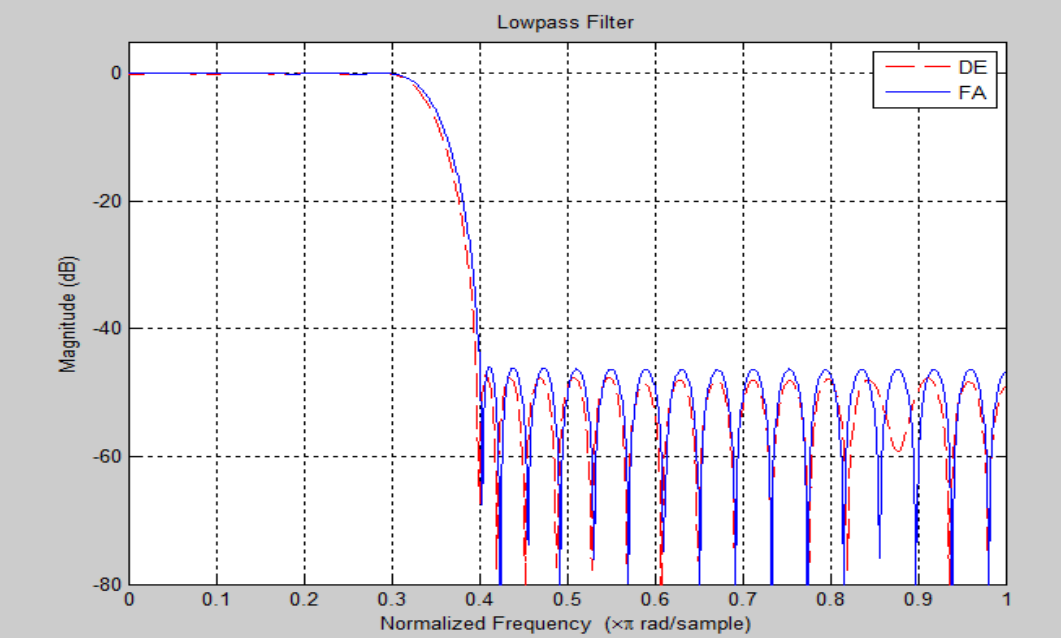


Figure 5.21 Lowpass 48 order FIR filter comparing FA and DE

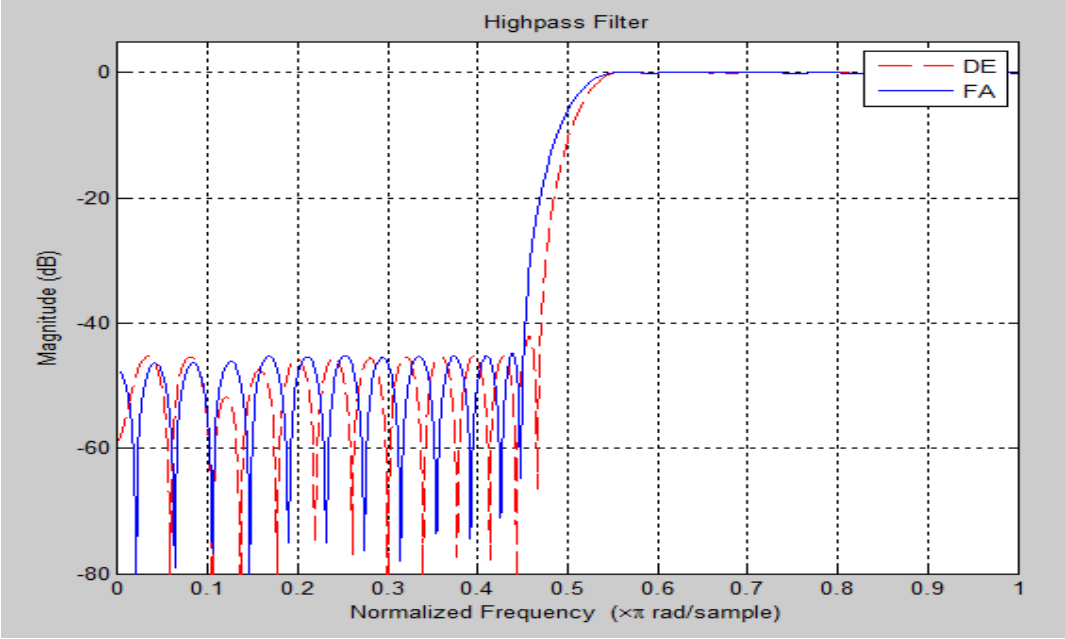


Figure 5.22 Highpass 48 order FIR filter comparing FA and DE

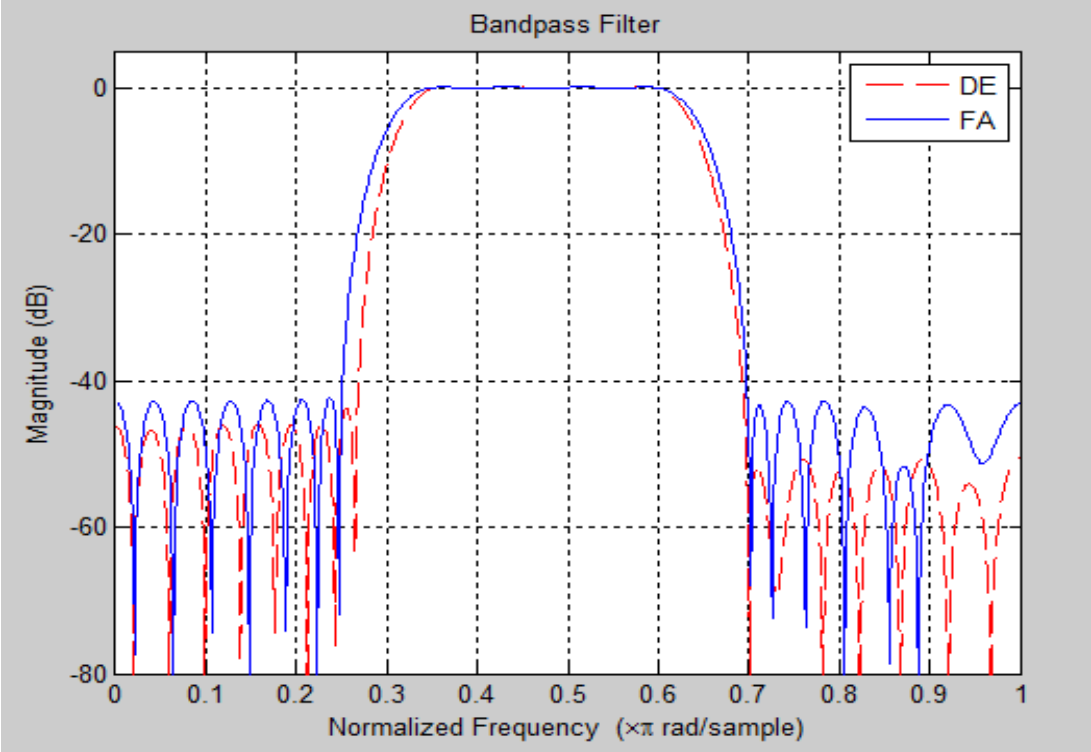


Figure 5.23 Bandpass 48 order FIR filter comparing FA and DE

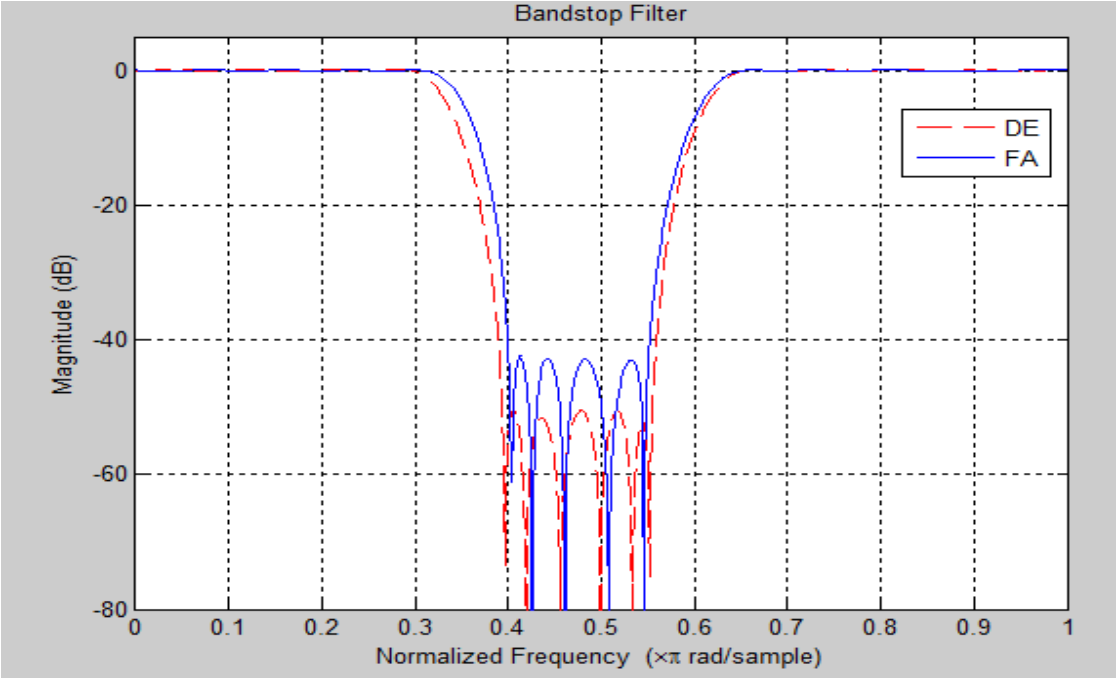


Figure 5.24 Bandstop 48 order FIR filter comparing FA and DE

TABLE 5.11 48th order FIR type 1 filter design results comparison (DE: Differential Evolution; FA: Firefly Algorithm)

Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU (Sec)**
Lowpass	FA	0.004918920471757	0.005732852275220	-	7458
	DE	0.003493591533472	0.008891830501439	-	22398
Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU (Sec)**
Highpass	FA	0.005698136342156	0.006061880324284	-	760.5
	DE	0.005148454290066	0.016946875805872	-	26037
Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU (Sec)**
Bandpass	FA	0.006804012796994	0.007574117936477	0.0069365942779	3633
	DE	0.004846592046081	0.01688614516569	0.0029581502868	56842
Filter	**Alg**	**Peak(Passband)**	**Peak(Stopband)**	**Peak(Passband 2)**	**CPU (Sec)**
Bandstop	FA	0.007154438758696	0.008353973095570	0.0019619551991	3605
	DE	0.018585510244063	0.002991846446768	0.0145169627263	38289

5.6 FA results compared with FIRPM (48 order)

For 48 order filter designs, Digital filters designs using FA were also compared with the state-of-the-art designs of Parks McClellan algorithm. It can be seen from the figure 5.25-5.28 that the results of FA and PM are almost identical.

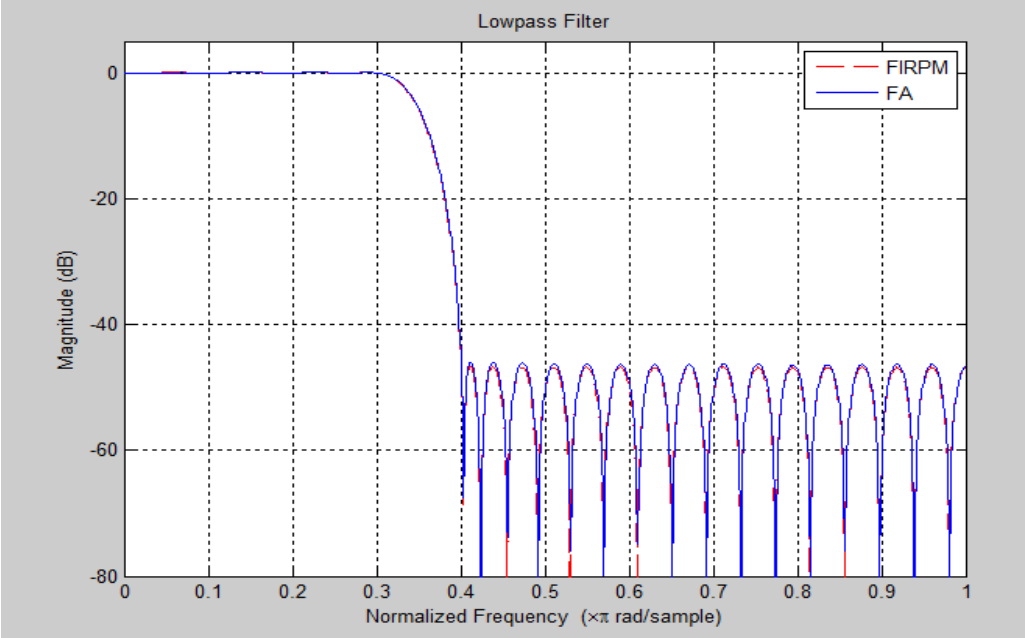


Figure 5.25 Lowpass 48 order FIR filter comparing FA and FIRPM

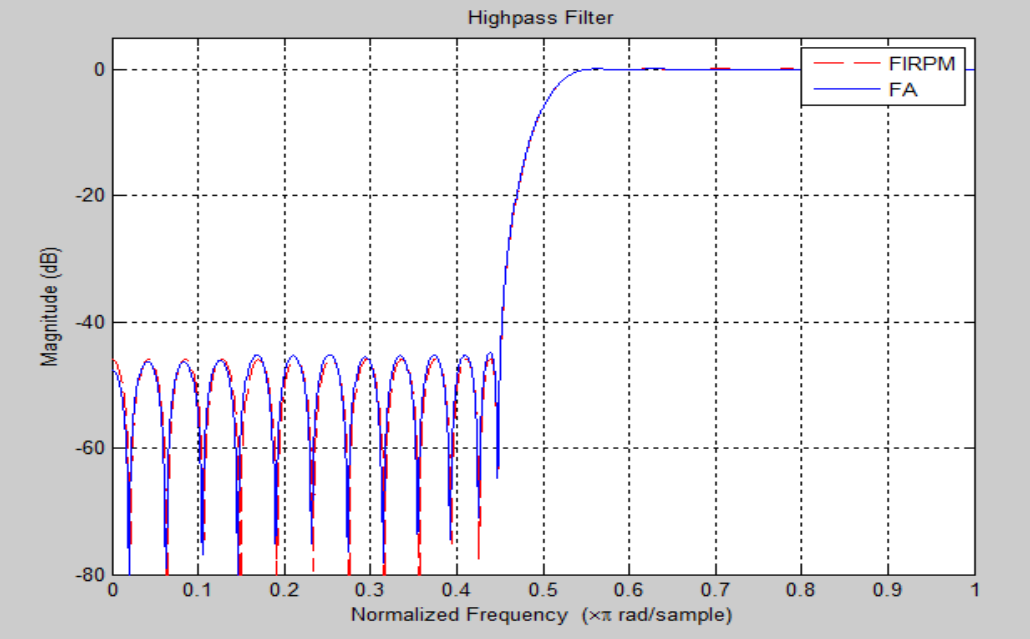


Figure 5.26 Highpass 48 order FIR filter comparing FA and FIRPM

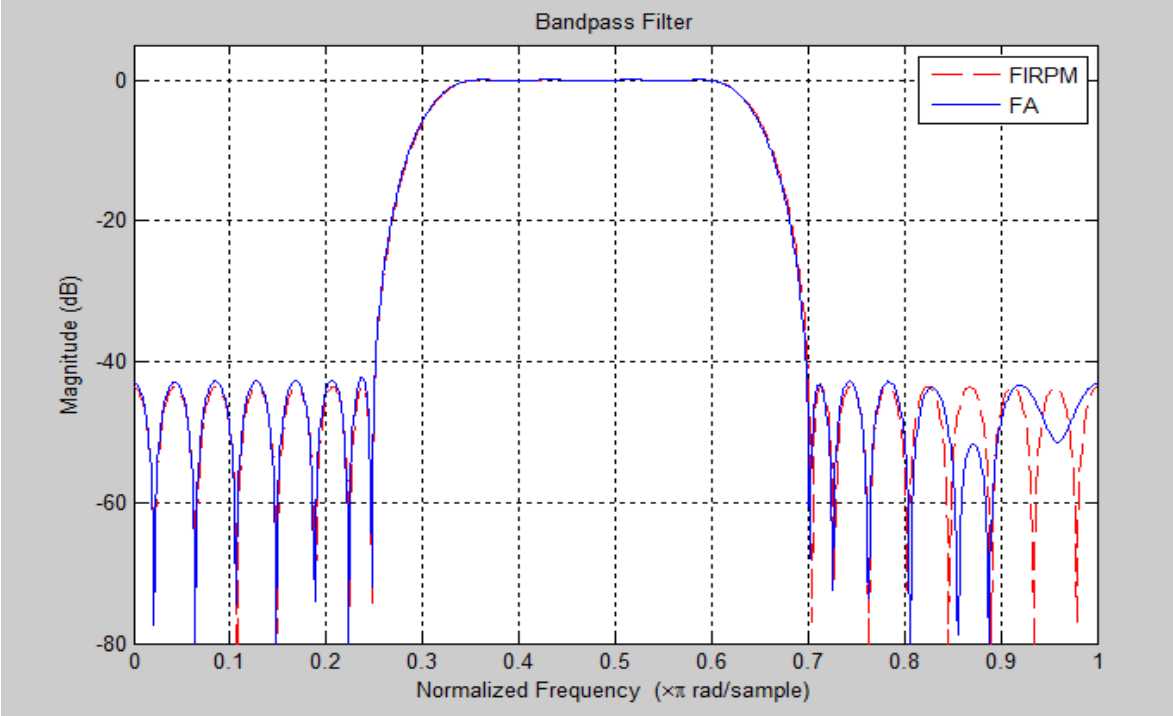


Figure 5.27 Bandpass 48 order FIR filter comparing FA and FIRPM

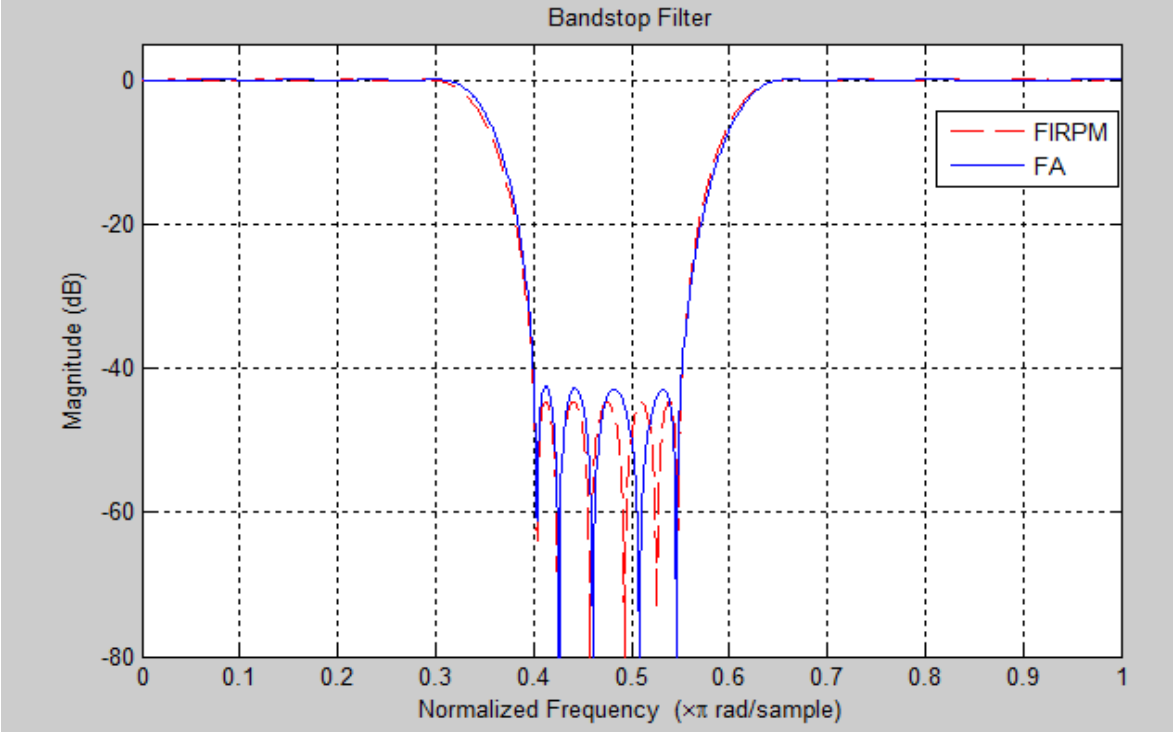


Figure 5.28 Bandstop 48 order FIR filter comparing FA and FIRPM

An overall result of all four LP-FIR digital filters for order 48 obtained by using Firefly algorithm compared to both Differential Evolution algorithm and Parks–McClellan algorithm can be shown in Table 5.12.

TABLE 5.12 48 order FIR type 1 filter design results comparison (FA: Firefly Algorithm, FIRPM: Parks–McClellan algorithm; DE: Differential Evolution Algorithm)

Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU (Sec)**
Lowpass	PM	0.00454629718373	0.0046315094120	-	0.1855
	FA	0.00491892047175	0.0057328522752	-	7458.7
	DE	0.00349359153347	0.0088918305014	-	22398
Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU (Sec)**
Highpass	PM	0.00415270222262	0.0061844490845	-	0.1909
	FA	0.00569813634215	0.0060618803242	-	760.5
	DE	0.00514845429006	0.0169468758058	-	26037
Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU (Sec)**
Bandpass	PM	0.00608494846052	0.0083788963211	0.0066587444012	0.1904
	FA	0.00680401279699	0.0075741179364	0.0069365942779	3633.2
	DE	0.00484659204608	0.0168861451656	0.0029581502868	56842
Filter	**Alg**	**Peak(Passband)**	**Peak(Stopband)**	**Peak(Passband 2)**	**CPU (Sec)**
Bandstop	PM	0.00521666237529	0.0067739876725	0.0058431362537	0.2106
	FA	0.00715443875869	0.0083539730955	0.0019619551991	3605.1
	DE	0.01858551024406	0.0029918464467	0.0145169627263	38289

5.7 General FIR results using FA:

General FIR (GFIR) filters are a special type of non-symmetrical filters specifically designed to keep the passband group delay of the filter constant. In order to keep the group delay constant, both magnitudes as well as group delays are optimized using the algorithm so the overall results affect the error values of the filters but it gives you a constant group delay which can be helpful in a number of applications. For Type 1 GFIR filter of order 24, Filter designs and passband group delays of filters using FA are given below:

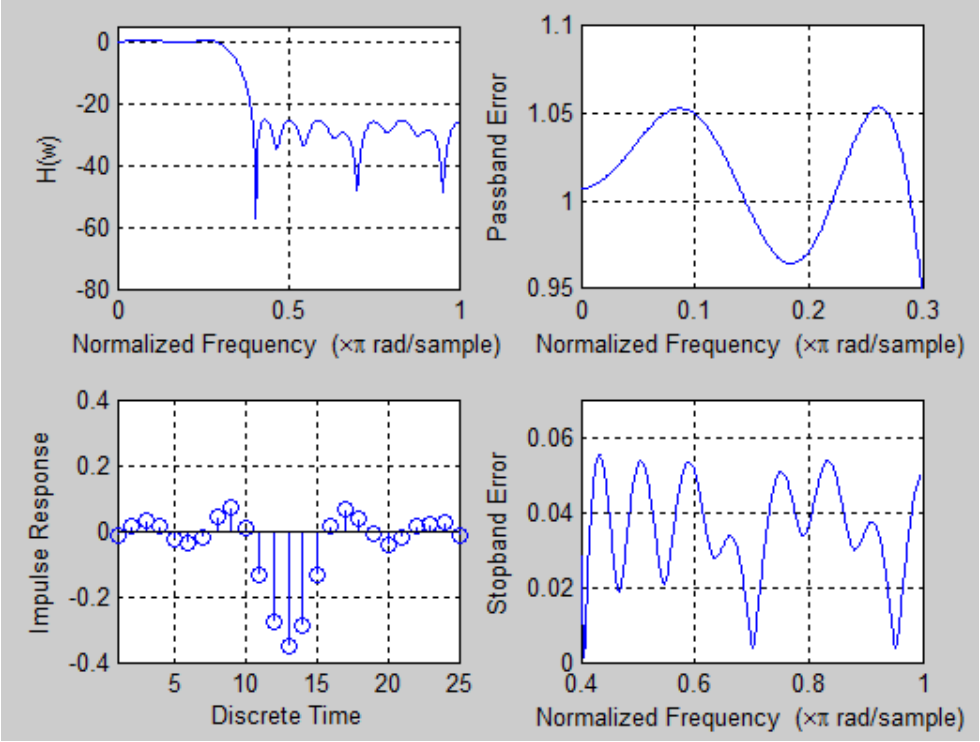


Figure 5.29 Lowpass GFIR filter using FA

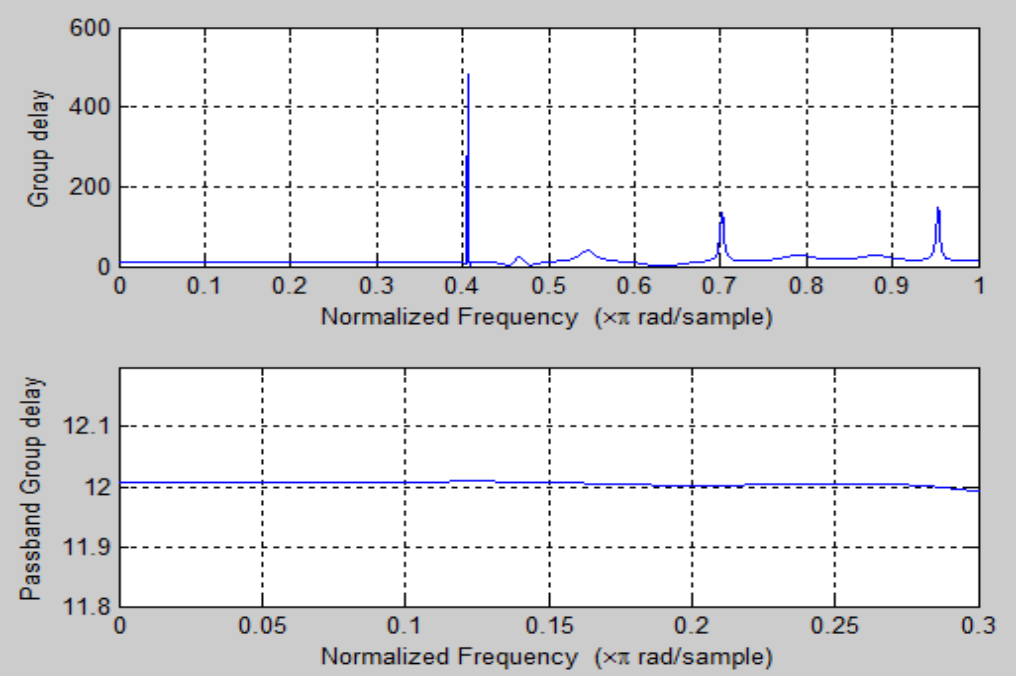


Figure 5.30 Passband Group delay of Lowpass GFIR filter using FA

TABLE 5.13 Coefficients of 24th-order type1 Lowpass LP-GFIR filter by FA

h(n)	Coefficients	h(n)	Coefficients
h(1) =	-0.012647936205156	h(14) =	-0.289090298187066
h(2) =	0.014258103471751	h(15) =	-0.134531923853745
h(3) =	0.029594275230518	h(16) =	0.017058764791044
h(4) =	0.011928151573815	h(17) =	0.067056248712457
h(5) =	-0.023300315621584	h(18) =	0.035132578530568
h(6) =	-0.036765348355070	h(19) =	-0.005803138746866
h(7) =	-0.017419207409947	h(20) =	-0.042526848525596
h(8) =	0.043592872642109	h(21) =	-0.021178295943202
h(9) =	0.070420412454785	h(22) =	0.013879926589004
h(10) =	0.010762317812604	h(23) =	0.021108654778234
h(11) =	-0.134152685617516	h(24) =	0.022999641854860
h(12) =	-0.279773650746019	h(25) =	-0.015682603266573
h(13) =	-0.352274608106788		

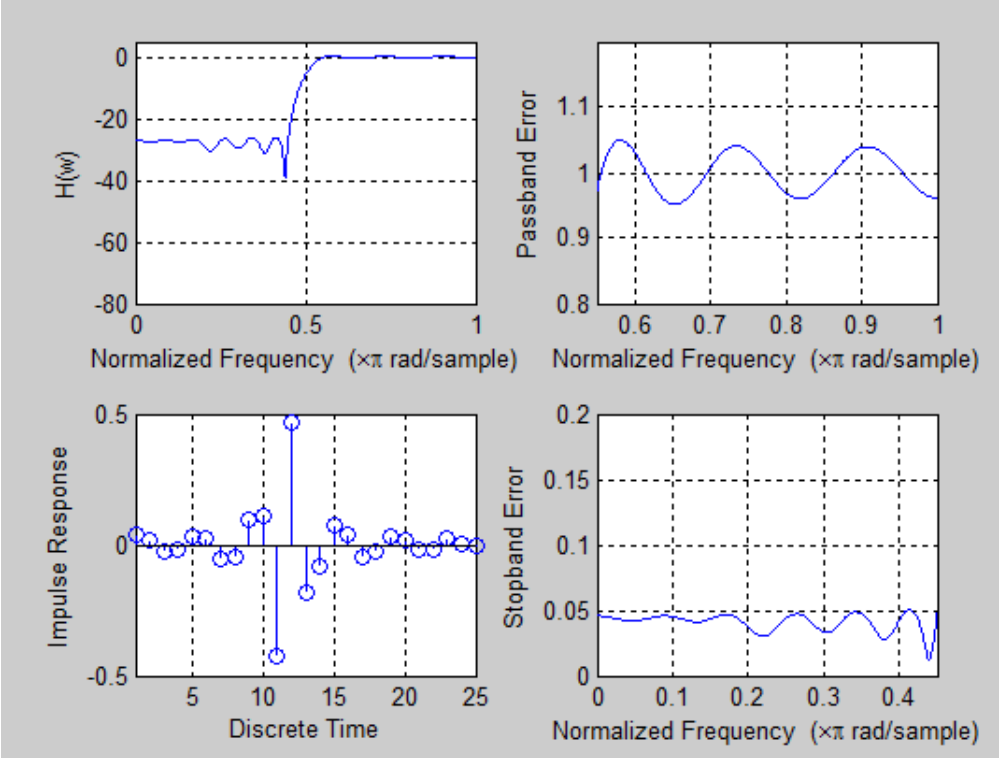


Figure 5.31 Highpass GFIR filter using FA

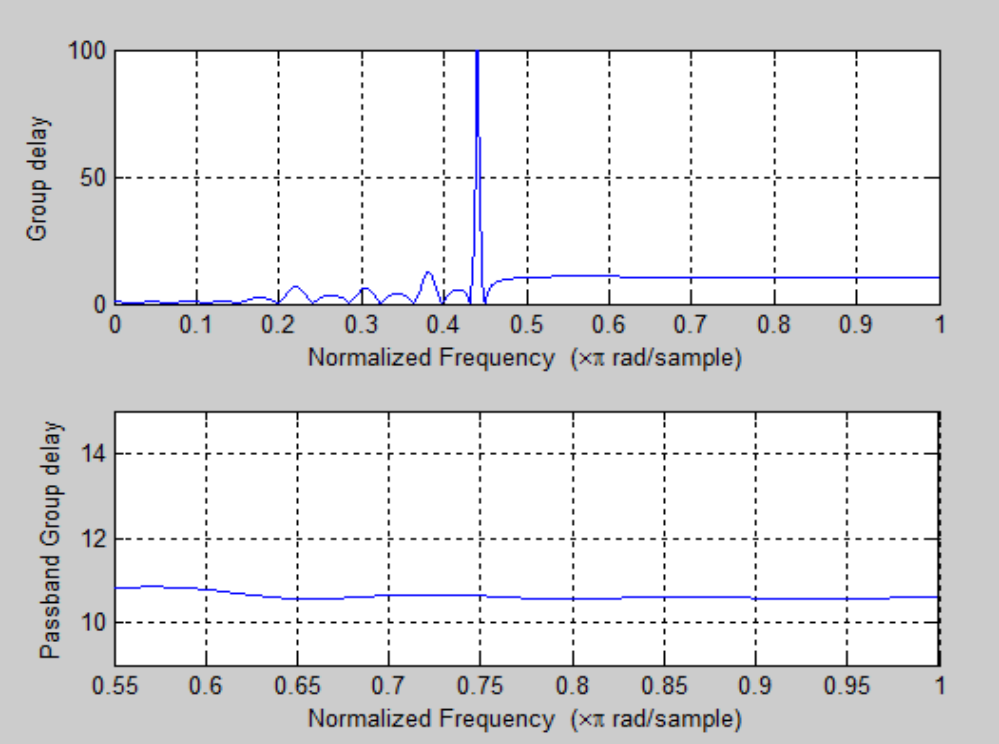


Figure 5.32 Passband Group delay of Highpass GFIR filter using FA

TABLE 5.14 Coefficients of 24th-order type1 Highpass LP-GFIR filter by FA

h(n)	Coefficients	h(n)	Coefficients
/h(1) =	0.039849797603173	h(14) =	-0.080555514819023
h(2) =	0.018695840043839	h(15) =	0.076379847444691
h(3) =	-0.021916261014334	h(16) =	0.041968726816746
h(4) =	-0.016956412697173	h(17) =	-0.044549079346631
h(5) =	0.034767744893026	h(18) =	-0.027203438122901
h(6) =	0.025802648830348	h(19) =	0.029606783071643
h(7) =	-0.051446915332656	h(20) =	0.017414782201597
h(8) =	-0.043861037434722	h(21) =	-0.019948113484051
h(9) =	0.092881624954950	h(22) =	-0.017359701722941
h(10) =	0.112686160775050	h(23) =	0.023694691381695
h(11) =	-0.428196623798469	h(24) =	0.002329498906252
h(12) =	0.470506281241059	h(25) =	-0.006033871107464
h(13) =	-0.182699661191621		

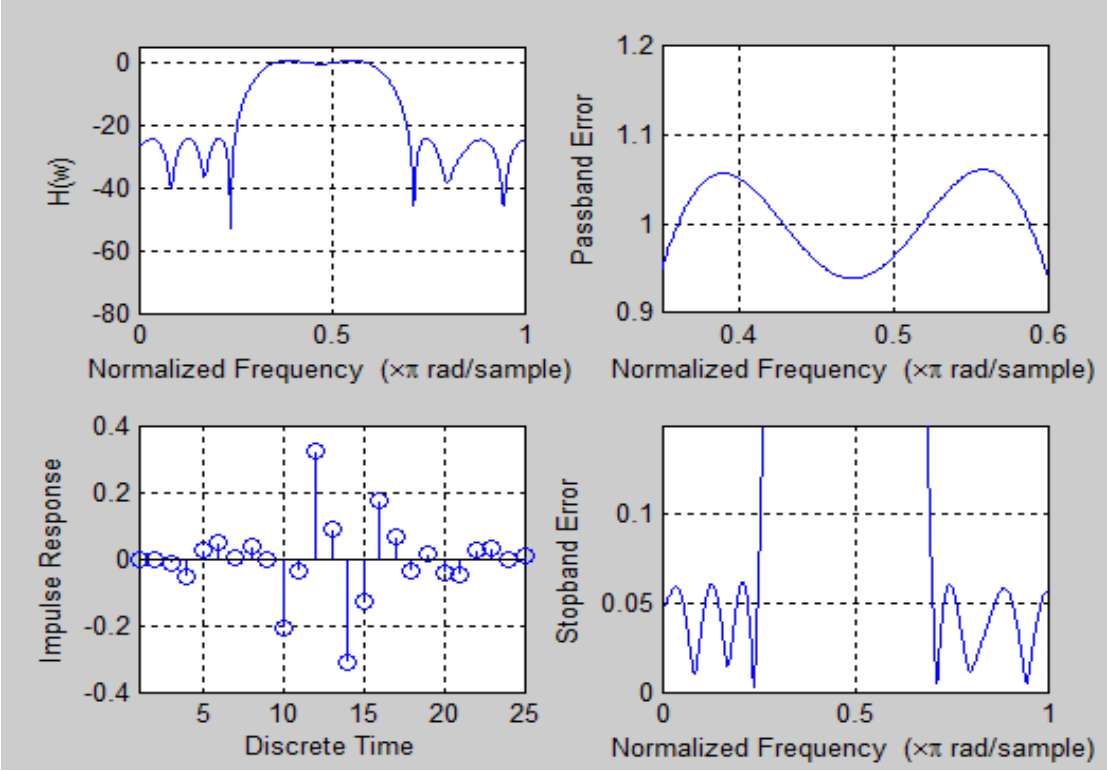


Figure 5.33 Bandpass GFIR filter using FA

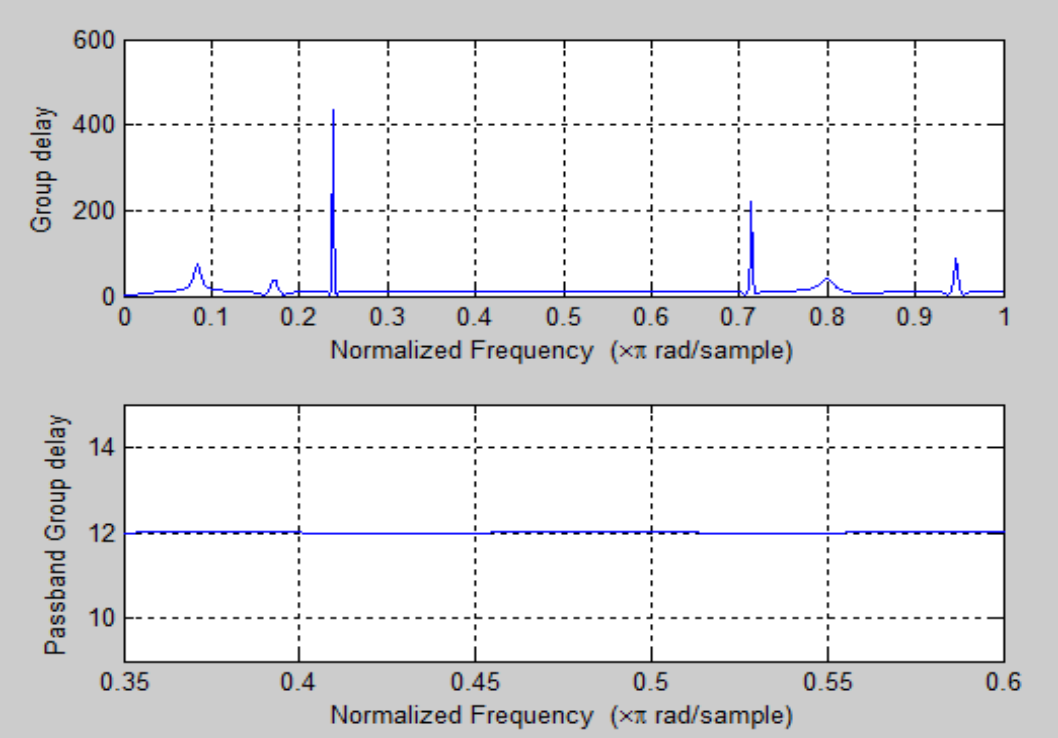
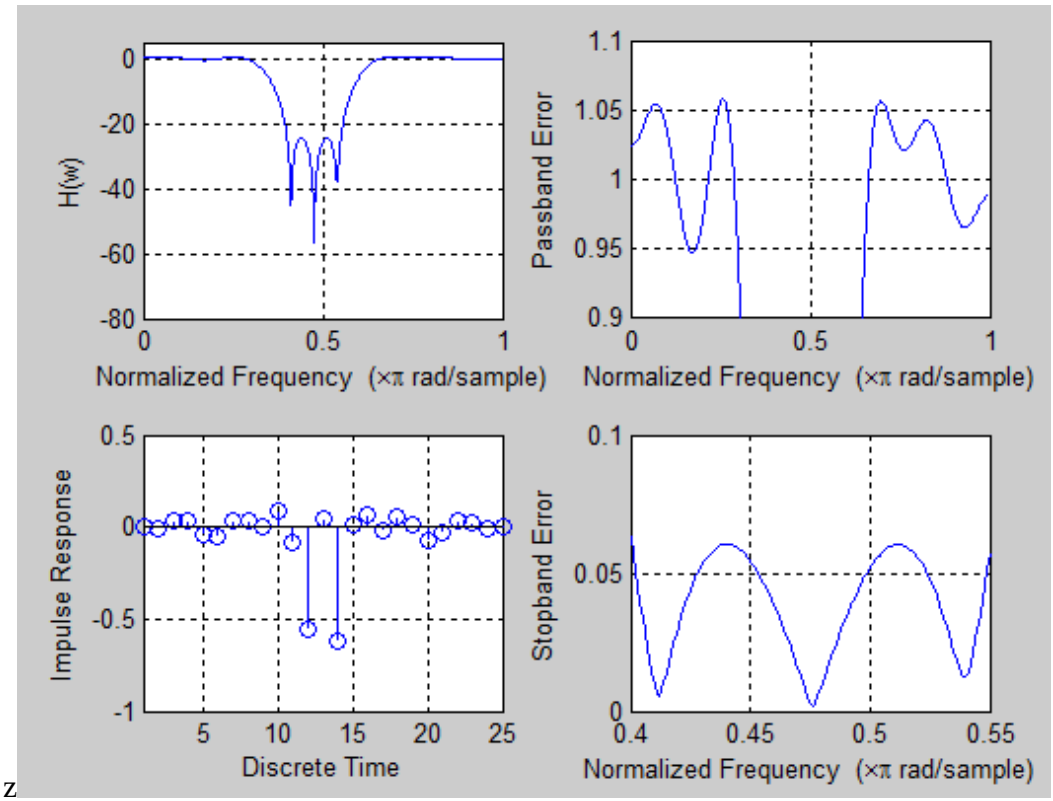


Figure 5.34 Passband Group delay of Bandpass GFIR filter using FA

TABLE 5.15 Coefficients of 24th-order type1 Bandpass LP-GFIR filter by FA

h(n)	Coefficients	h(n)	Coefficients
h(1) =	-0.005264442142425	h(14) =	-0.309752115931914
h(2) =	-0.003251180048295	h(15) =	-0.127019843912715
h(3) =	-0.017057964010024	h(16) =	0.174016579318099
h(4) =	-0.056600976847803	h(17) =	0.067385224044092
h(5) =	0.025332557516588	h(18) =	-0.039710193331936
h(6) =	0.046003050491257	h(19) =	0.016265014619499
h(7) =	0.000000289935042	h(20) =	-0.041765937423746
h(8) =	0.037708658061923	h(21) =	-0.046313697870373
h(9) =	-0.002763051929089	h(22) =	0.025364363108612
h(10) =	-0.208018565775791	h(23) =	0.032311020087952
h(11) =	-0.035528386024917	h(24) =	-0.001113477097570
h(12) =	0.324828562050157	h(25) =	0.008354772068004
h(13) =	0.088949149048855		



z

Figure 5.35 Bandstop GFIR filter using FA

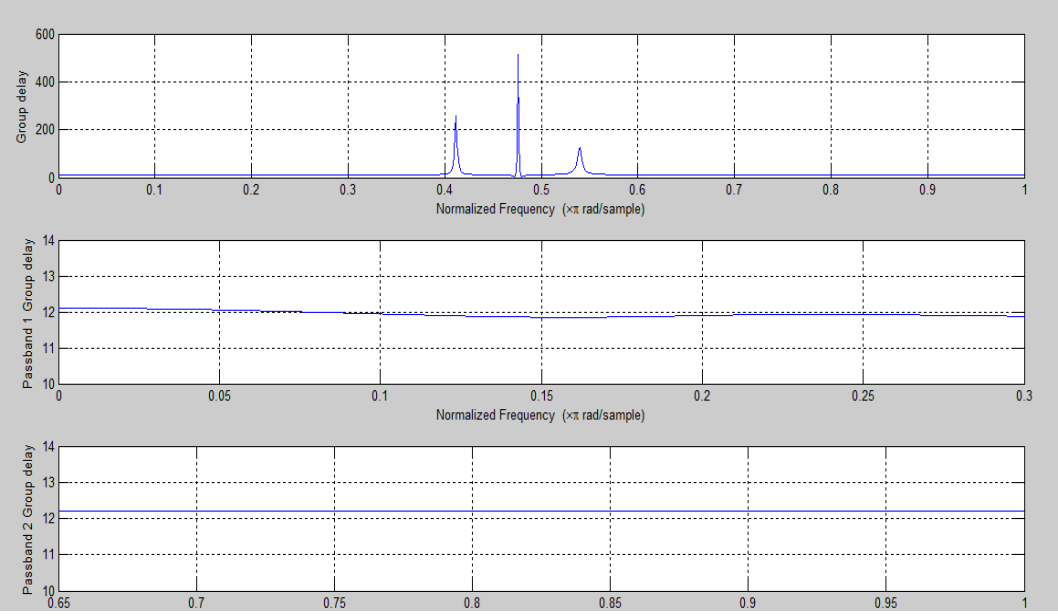


Figure 5.36 Passband Group delays of bandstop GFIR filter using FA

TABLE 5.16 Coefficients of 24th-order type1 Bandstop LP-GFIR filter by FA

h(n)	Coefficients	h(n)	Coefficients
h(1) =	0.000410104279232	h(14) =	0.573220421065861
h(2) =	-0.021979005473702	h(15) =	-0.340155687341897
h(3) =	-0.044774668187071	h(16) =	0.066711038957858
h(4) =	-0.006436500068437	h(17) =	-0.059778275610259
h(5) =	0.055831832982663	h(18) =	-0.092363321274940
h(6) =	0.004001510031709	h(19) =	0.055822010089099
h(7) =	-0.008563906049588	h(20) =	0.079036725701632
h(8) =	0.012171689799200	h(21) =	-0.018976317251613
h(9) =	-0.042274620543853	h(22) =	-0.029994890714089
h(10) =	0.027296746105513	h(23) =	0.009284805984681
h(11) =	0.320470024533179	h(24) =	-0.014668308284967
h(12) =	0.376280373760145	h(25) =	-0.005870395557371
h(13) =	0.064553050371079		

Table 5.17 shows the error values of the designed GFIR filters. All four filters of order 24 are shown in the table.

TABLE 5.17 24 order GFIR filter design results (FA: Firefly Algorithm)

Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU(Sec)**
Lowpass	FA	0.055061161922455	0.054006708974618	-	295.73
Filter	**Alg**	**Peak(Stopband)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU(Sec)**
Highpass	FA	0.050490285441068	0.050309748431140	-	246.01
Filter	**Alg**	**Peak(Stopband1)**	**Peak(Passband)**	**Peak(Stopband 2)**	**CPU(Sec)**
Bandpass	FA	0.06243906488996	0.061536725079860	0.056918298933613	249.74
Filter	**Alg**	**Peak(Passband)**	**Peak(Stopband)**	**Peak(Passband 2)**	**CPU(Sec)**
Bandstop	FA	0.0.05356740268252	0.057264335609858	0.051718701381042	250.95

Chapter 6
Conclusions

The algorithm presented in this thesis finds the filter coefficients and neural network parameters on par with the state of the art methods. With other evolutionary methods like DE, it is very difficult to achieve global optimum solutions in a reasonable amount of time. The adjustable parameters of Firefly algorithm helps the user to achieve results according to their needs and meets the error constraints accordingly. The design examples shows that results are achieved faster using FA and efficient compared to other methods except for the Parks McClellan method which is globally considered as the best method for linear phase digital filter design but is very complex to execute. FA proves to be a good alternative as it gives results near to PM in almost all the designs. The flexibility of the algorithm with adjustable parameters means that the algorithm can be implemented and adjusted with ease with less expert intervention.

Although the scope of the thesis is to present a study on Firefly algorithm and extract the best possible results for the chosen designs, FA can also be used for different applications and it is being used worldwide as a powerful and time efficient optimization algorithm.

The major limitation, which needs to be further addressed is that, how to deal with complex problems without getting stuck in local minima. Due to the sensitivity of some of the parameters, it is difficult to avoid the local minima problem if the right parameters are not chosen. All in all, it is a good algorithm capable of solving practical problems with ease and reduced time compared to the other popular algorithms.

REFERENCES

[1] X. S. Yang, “Firefly algorithm, Levy flights and global optimization,” in Research and Development in Intelligent Systems XXVI, pp. 209–218, Springer, London, UK, 2010.

[2] S. K. Saha, R. Kar, D. Mandal, and Sakti Prasad Ghoshal, B.K. Panigrahi Optimal Stable IIR Low Pass Filter Design Using Modified Firefly Algorithm, Suman K. Saha, Rajib Kar, Durbadal Mandal, and Sakti Prasad Ghoshal, B.K. Panigrahi et al. (Eds.): SEMCCO 2013, Part I, LNCS 8297, pp. 98–109, 2013. Springer International Publishing Switzerland 2013

[3] H. K. Kwan, Optimization Methods for Digital Filter Design, Edition 1.1, dfisp.org, ISBN: 9780993670794, 19 February 2016.

[4] H. K. Kwan, Digital filters and system, e-book Edition 1.1, dfisp.org, date?

[5] Sabbir U. Ahmad and Andreas Antoniou, "Design of Digital filters using Genetic Algorithms," IEEE Transaction on Signal Processing, Vol. 1, No. 1, pp. 1-9, 2006.

[6] N. Karaboga, and B. Cetinkaya (2006). Design of digital FIR filters using differential evolution algorithm. Circuits, Systems and Signal Processing, 25(5), 649-660

[7] N. Karaboga, (2009). A new design method based on artificial bee colony algorithm for digital IIR filters. Journal of the Franklin Institute, 346(4), 328-348.

[8] H.K. Kwan and C. Z.Tang, Designing multilayer feedforward neural networks using simplified sigmoid activation functions, ELECTRONICS LETTERS

[9] X.-S. Yang, “Firefly algorithms for multimodal optimization”, in Stochastic Algorithms: Foundations and Applications, SAGA 2009, Lecture Notes in Computer Sciences, Vol. 5792, pp. 169-178 (2009).

[10] I. Fister, I. Fister Jr., X.-S. Yang, and J. Brest, A comprehensive review of firefly algorithms, Swarm and Evolutionary Computation, vol. 13, pp. 34-46, 2013.

[11] X. S. Yang, Nature-Inspired Meta-Heuristic Algorithms, Luniver Press, Beckington, UK, 2008.

[12] S. Lukasik and S. Zak, “Firefly algorithm for continuous constrained optimization tasks,” in Proceedings of the International Conference on Computer and Computational Intelligence (ICCCI '09), N. T. Nguyen, R. Kowalczyk, and S.-M. Chen, Eds., vol. 5796 of LNAI, pp. 97–106, Springer, Wroclaw, Poland, October 2009.

[13] X. S. Yang, “Firefly algorithms for multimodal optimization,” in Proceedings of the Stochastic solving the Economic Emissions Load Dispatch Problem,” International Journal of Combinatorics, vol. 2011, Article ID 523806, 23 pages, 2011. doi:10.1155/2011/523806

[14] X .S. Yang, “Firefly algorithm, stochastic test functions and design optimization,” International Journal of Bio-Inspired Computation, vol. 2, no. 2, pp. 78–84, 2010.

[15] L. Cagnina, C. Esquivel, and S. C., Coello, C. A. (2008) ‘Solving engineering optimization problems with the simply constrained particle swarm optimizer’, Informatica, 32, 319-326 (2008).

[16] X. S. Yang, “Firefly algorithms for multimodal optimization,” in Proceedings of the Stochastic Algorithms: Foundations and Applications (SAGA '09), vol. 5792 of Lecture Notes in Computing Sciences, pp. 178–178, Springer, Sapporo, Japan, October 2009.

[17] T. Apostolopoulos and A. Vlachos, “Application of the Firefly Algorithm for Sgshi He, (2013). ‘Firefly Algorithm: Recent Advances and Applications’, Int. J. Swarm Intelligence, Vol. 1, No. 1, pp. 36–50. DOI: 10.1504/IJSI.2013.05580

[18] A. A. Abshouri, M. R. Meybodi, and A. Bakhtiari, New firefly algorithm based on multiswarm and learning automata in dynamic environments, Third Int. Conference on 11 Signal Processing Systems (ICSPS2011), Aug 27-28, Yantai, China, pp. 73-77 (2011).

[19] S. K. Azad, and S. K. Azad, Optimum Design of Structures Using an Improved Firefly Algorithm, International Journal of Optimization in Civil Engineering, 1(2), 327-340(2011).

[20] T. Apostolopoulos and A. Vlachos, Application of the Firefly Algorithm for Solving the Economic Emissions Load Dispatch Problem, International Journal of Combinatorics, Volume 2011, Article ID 523806. (2011).

[21] H. Banati and M. Bajaj, Firefly based feature selection approach, Int. J. Computer Science Issues, 8(2), 473-480 (2011).

[22] B. Basu and G. K. Mahanti, Firefly and artificial bees colony algorithm for the synthesis of the scanned and broadside linear array antenna, Progress in Electromagnetic Research B., 32, 169-190 (2011).

[23] A. Chatterjee, G. K. Mahanti, and A. Chatterjee, Design of a fully digital controlled reconfigurable switched beam nonconcentric ring array antenna using firefly and particle swarm optimization algorithm, Progress in Electromagnetic Research B., 36, 113-131 (2012).

[24] S. M. Farahani, A. A. Abshouri, B. Nasiri, and M. R. Meybodi, A Gaussian firefly algorithm, Int. J. Machine Learning and Computing, 1(5), 448-453 (2011).

[25] S. M. Farahani, B. Nasiri and M. R. Meybodi, A multiswarm based firefly algorithm in dynamic environments, in Third Int. Conference on Signal Processing Systems (ICSPS2011), Aug 27-28, Yantai, China, pp. 68-72 (2011) 12

[26] A. H. Gandomi, X. S. Yang, and A. H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solving structural optimization problems, Engineering with Computers, 27, article DOI 10.1007/s00366-011-0241-y, (2011).

[27] T. Hassanzadeh, H. Vojodi, and A. M. E. Moghadam, An image segmentation approach based on maximum variance intra-cluster method and firefly algorithm, in Proc. of 7th Int. Conf. on Natural Computation (ICNC2011), pp. 1817-1821 (2011).

[28] M.-H. Horng, Y.-X. Lee, M.-C. Lee and R.-J. Liou, Firefly metaheuristic algorithm for training the radial basis function network for data classification and disease diagnosis, in Theory and New Applications of

Swarm Intelligence (Edited by R. Parpinelli and H. S. Lopes), pp. 115-132 (2012).

[29] M.-H. Horng, Vector quantization using the firefly algorithm for image compression, Expert Systems with Applications, 39, pp. 1078-1091 (2012

[30] G. K. Jati and S. Suyanto, Evolutionary discrete firefly algorithm for traveling salesman problem, ICAIS2011, Lecture Notes in Artificial Intelligence (LNAI 6943), pp.393- 403 (2011).

[31] S. Nandy, P. P. Sarkar, and A. Das, Analysis of nature-inspired firefly algorithm based back-propagation neural network training, Int. J. Computer Applications, 43(22), 8- 16 (2012).

[32] S. Palit, S. Sinha, M. Molla, A. Khanra, and M. Kule, A cryptanalytic attack on the knapsack cryptosystem using binary Firefly algorithm, in 2nd Int. Conference on Computer and Communication Technology (ICCCT), 15-17 Sept 2011, India, pp. 428- 432 (2011).

[33] A. Rajini, and V. K. David, A hybrid metaheuristic algorithm for classification using microarray data, Int. J. Scientific & Engineering Research, 3(2), 1-9 (2012). 13

[34] M. Sayadi, R. Ramezanian, and N. Ghaffari-Nasab, (2010). 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.

[35] J. Senthilnath, S. N. Omkar, and V. Mani, Clustering using firefly algorithm: performance study, Swarm and Evolutionary Computation, 1(3), 164-171 (2011).

[36] X. S. Yang, Firefly algorithms for multimodal optimization, Proc. 5th Symposium on Stochastic Algorithms, Foundations and Applications, (Eds. O. Watanabe and T. Zeugmann), Lecture Notes in Computer Science, 5792: 169-178 (2009).

[37] X. S. Yang, Firefly algorithm, stochastic test functions and design optimization, Int. J. Bio-Inspired Computation, 2(2), 78-84 (2010).

[38] X. S. Yang and S. Deb, Cuckoo search via L´evy flights, Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009, India), IEEE Publications, USA, pp. 210-214 (2009)

[39] X. S. Yang, Swarm-based metaheuristic algorithms and no-free-lunch theorems, in Theory and New Applications of Swarm Intelligence (Eds. R. Parpinelli and H. S. Lopes), Intech Open Science, pp. 1-16 (2012)

[40] A. Yousif, A. H. Abdullah, S. M. Nor, A. A. Abdelaziz, Scheduling jobs on grid computing using firefly algorithm, J. Theoretical and Applied Information Technology, 33(2), 155-164 (2011).

[41] M. A. Zaman and M. A. Matin, Nonuniformly spaced linear antenna array design using firefly algorithm, Int. J. Microwave Science and Technology, Vol. 2012, Article ID: 256759, (8 pages), 2012. doi:10.1155/2012/256759

[42] T. Samad, 1990, "Back-propagation improvements based on heuristic arguments", Proceedings of International Joint Conference on Neural Networks, Washington, 1, pp. 565-568.

[43] M. T. Hagan and M. B. Menhaj, 1994, "Training feedforward networks with the Marquardt algorithm," IEEE Trans. Neural Netw. , vol. 5, no. 6, pp. 989–993.

[44] C. Charalambous, 1992, "Conjugate gradient algorithm for efficient training of artificial neural networks", IEEE Proceedings, Vol. 139, No. 3, pp. 301-310.

[45] E. Bonabeau, M. Dorigo, and G. Theraulaz: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, (1999)

[46] K. Deb, Optimization of Engineering Design, Prentice-Hall, New Delhi, (1995).

[47] T. Baeck, D. B. Fogel, Z. Michalewicz: Handbook of Evolutionary Computation, Taylor & Francis, (1997).

[48] D. Shilane, J. Martikainen, S. Dudoit, and S. J. Ovaska: A general framework for statistical performance comparison of evolutionary computation algorithms, Information Sciences: an Int. Journal, 178, 2870-2879 (2008).

[49] J. Kennedy, and R.C. Eberhart: Particle swarm optimization. In: Proc. of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942–1948 (1995)

[50] J. Kennedy, R. Eberhart, and Y. Shi: Swarm intelligence. Academic Press, London (2001)

[51] X.S. Yang: Engineering Optimization: An Introduction to Metaheuristic Applications. Wiley, Chichester (2010)

[52] S. Kirkpatrick, C. Gelatt, and M. Vecchi, “Optimization by simulated annealing,” Science, Vol. 220, 671–680, 1983.

[53] M. Panduro, D. H. Covarrubias, and C. Brizuela, “Design of electronically steerable linear arrays with evolutionary algorithms,” Applied Soft Computing, Vol. 8, 46–54, 2008.

[54] A. E. Eiben and J. E. Smith, Introduction to Evolutionary Computing, Springer, 2003.

[55] J. C. Spall, S. D. Hill and D. R. Stark, Theoretical framework for comparing several stochastic optimization algorithms, in: Probabilistic and Randomized Methods for Design Under Uncertainty, Springer, London, pp. 99-117 (2006).

[56] D. Corne and J. Knowles, Some multiobjective optimizers are better than others, Evolutionary Computation, CEC’03, 4, 2506-2512 (2003).

[57] J. He, and X. Yu, Conditions for the convergence of evolutionary algorithms, J. Systems Architecture, 47, 601-612 (2001).

[58] D. Karaboga, (2005). An idea based on honey bee swarm for numerical optimization, Technical Report TR06, Erciyes University, Turkey

[59] D.T. Pham, A. Ghanbarzadeh, S.Rahim, S., and M. Zaidi, (2006). The Bees Algorithm U A Novel Tool for Complex Optimisation Problems, Proceedings of ″ IPROMS 2006 Conference, pp.454-461

[60] J. C. Spall, S. D. Hill, and D. R. Stark, Theoretical framework for comparing several stochastic optimization algorithms, in: Probabilistic and Randomized Methods for Design Under Uncertainty, Springer, London, pp. 99-117 (2006).

[61] X. S. Yang, Engineering Optimization: An Introduction to Metaheuristic Applications, John Wiley, and Sons, USA (2010).

[62] E. Bonabeau, M. Dorigo, G. Theraulaz, (1999). Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press. [5] Blum, C. and Roli, A. (2003)

[63] Blum, Christian, Roli, Andrea, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Computing Surveys (CSUR), v.35 n.3, p.268-308, September 2003 [doi>10.1145/937503.937505]

[64] M. S. Malhi, “Linear-Phase FIR Digital Filter Design with Reduced Hardware Complexity using Extremal Optimization" (2016). Electronic Theses and Dissertations. 5746.

[65] W. Zhong,"Linear Phase FIR Digital Filter Design Using Differential Evolution Algorithms" (2017). Electronic Theses and Dissertations. 5959.

[66] H. K. Kwan and Rija Raju, “FIR filter design using multiobjective artificial bee colony algorithm”, Proceedings of 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, Ontario, Canada, 30 April-3 May 2017, 4 pages.

[67] H. K. Kwan and Jiajun Liang, “FIR filter design using multiobjective cuckoo search algorithm”, Proceedings of 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, Ontario, Canada, 30 April-3 May 2017, 4 pages.

[68] H. K. Kwan and Miao Zhang, “FIR filter design using multiobjective teaching-learning-based optimization”, Proceedings of 2017 IEEE 30th

Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, Ontario, Canada, 30 April-3 May 2017, 4 pages.

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

[70] H. K. Kwan and Jiajun Liang, "Design of linear phase FIR filters using cuckoo search algorithm", in Proceedings of 8th International Conference on Wireless Communications and Signal Processing (WCSP 2016), Yangzhou, Jiangsu, China, October 13-15, 2016, pages 1-4.

[71] H. K. Kwan and Rija Raju, "Minimax design of linear phase FIR differentiators using artificial bee colony algorithm”, in Proceedings of 8th International Conference on Wireless Communications and Signal Processing (WCSP 2016), Yangzhou, Jiangsu, China, October 13-15, 2016, pages 1-4.

[72] H. K. Kwan and Miao Zhang, "Minimax design of linear phase FIR Hilbert transformer using teaching-learning-based optimization”, in Proceedings of 8th International Conference on Wireless Communications and Signal Processing (WCSP 2016), Yangzhou, Jiangsu, China, October 13-15, 2016, pages 1-4.

VITA AUCTORIS

NAME:	Ghazanfer Ali
PLACE OF BIRTH:	Windsor, ON
YEAR OF BIRTH:	1992
EDUCATION:	Sir Syed University of Engineering and Technology, B.S in Electronics Engineering, Karachi, Pakistan, 2013
	University of Windsor, MAS Electrical, Windsor, ON, 2017