2016

Linear-Phase FIR Digital Filter Design with Reduced Hardware Complexity using Extremal Optimization

Manpreet Singh Malhi
University of Windsor

Linear-Phase FIR Digital Filter Design with Reduced Hardware Complexity using Extremal Optimization

By

Manpreet Singh Malhi

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

2016

Linear-Phase FIR Digital Filter Design with Reduced Hardware Complexity using Extremal Optimization

by

Manpreet Singh Malhi

APPROVED BY:

_____

Dr. Guoqing Zhang
Department of Mechanical, Automotive & Materials Engineering

_____

Dr. Huapeng Wu
Department of Electrical and Computer Engineering

_____

Dr. Hon Keung Kwan, Advisor
Department of Electrical and Computer Engineering

31 March, 2016

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

Extremal Optimization is a recent method for solving hard optimization problems. It has been successfully applied on many optimization problems. Extremal optimization does not share the disadvantage of most of the other evolutionary algorithms, which is the tendency to converge into local minima. Design of finite word length FIR filters using deterministic techniques can guarantee optimality at the expense of exponential increase in computational complexity. Alternatively, Evolutionary Algorithms are capable of converging very fast to a minimum, but have higher chances of failure if the ratio of feasible solutions is very less in the search space. In this thesis, a set of feasible solutions are determined by linear programming. In the second step, Extremal Optimization is used to further refine these results. This strategy helps by reducing the search space for the EO algorithm and is able to find good solutions in much shorter time than the existing methods.

Dedicated to my mother, who supported me both financially and morally and to my wife.

ACKNOWLEDGEMENTS

I would like to thank Prof. H. K. Kwan for introducing me the subject of digital filter design using extremal optimization and his guidance and support throughout the research. I would also like to thank Dr. H. Wu and Dr. G. Zhang for their valuable feedback and suggestions. And also, Kunwar Rehan for the valuable discussions during my research work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS/SYMBOLS

ACO	Ant Colony Optimization
b	Passband gain
CSA	Carry Save Adders
CSD	Canonical Signed Digit
DFT	Discrete Fourier Transform
$D(\omega)$	Desired frequency response
δ	Passband ripple
δ_p	Specified passband ripple
δ_s	Specified stopband ripple
EO	Extremal Optimization
EWL	Effective Word-length
$E(\omega)$	Weighted error function
FIR	Finite Impulse Response
g	Passband gain
GA	Genetic Algorithm
h_k	k^{th} filter coefficient
$H(\omega)$	Frequency response of the filter
IIR	Infinite Impulse Response
LP	Linear Program
MAG	Minimized Adder Graph
MCM	Multiple Constant Multiplication
MILP	Mixed Integer Linear Programming
MSB	Most Significant Bit
N	Filter length
NP	Non-deterministic Polynomial Time
NPRM	Normalized Peak Ripple Magnitude
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
RAG-n	n-Dimensional Reduced Adder Graph
RCA	Ripple Carry Adders
SA	Simulated Annealing
S_{max}	Maximum absolute value of frequency response in stopband
SPOT (SPT)	Sum of Power of Two
$W(\omega)$	Weighting function
ω	Normalized frequency
ω_p	Passband cut-off frequency
ω_s	Stopband cut-off frequency

Chapter 1

Introduction

An analog filter in electronics is used to remove or attenuate certain frequencies from the input signal and allow others to pass. Digital filters have similar purpose, but they work on digital signals. A digital filter works by doing mathematical operations on the input signal and its delayed versions.

The use of digital filters is widespread nowadays due to number of advantages over analog filters, some of which are shown below:

1. Digital filters can be used to achieve frequency responses which are impossible or difficult to achieve using analog filters. For example, it is extremely difficult to construct linear phase analog filters.
2. 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 response of similar analog filters are never exactly same. Also, the values of capacitors, resistors, etc. used in the analog filters may change with ageing, resulting in the change in filter characteristics.
3. Analog filters usually take lots of space for their construction compared to digital filters.
4. The characteristics of the digital filters can be easily changed by reprogramming, while an analog filter might need a complete redesign of the circuit.

1.1 Types of filters

1.1.1 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.

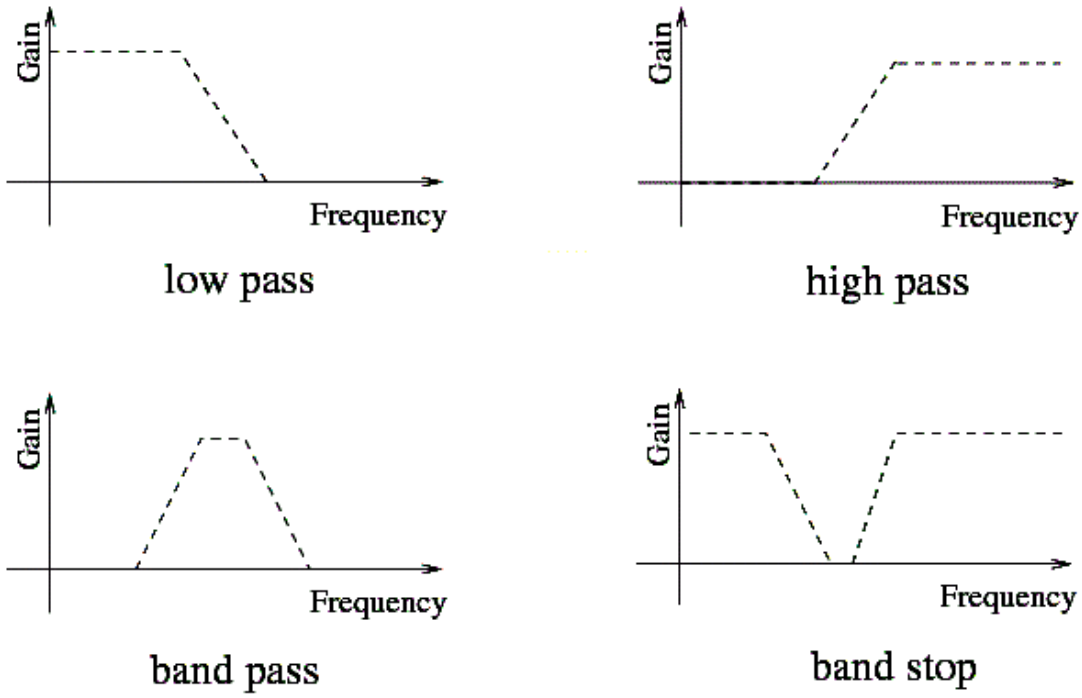


Fig. 1.1. Types of filters based on the frequency response

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. **Band pass filter:** The band pass filter allows only a certain band of frequencies to pass.
4. **Band stop filter:** The band stop 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 slant 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.

1.1.2 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 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.

1.2 FIR filters

Fig. 1.2 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.

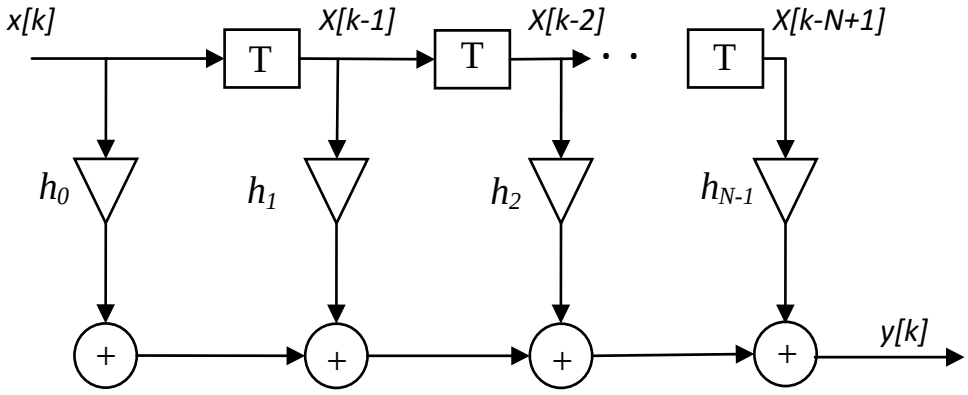


Fig. 1.2. FIR filter in direct form

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

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

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

$$H(z) = \sum_{n=0}^{N-1} h_n z^{-n} \qquad 1.2$$

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.

$$H\left(e^{j\omega T}\right) = \sum_{n=0}^{N-1} h_n e^{-j\omega nT} = \sum_{n=0}^{N-1} h_n \cos(\omega nT) - j \sum_{n=0}^{N-1} h_n \sin(\omega nT) \qquad 1.3$$

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.

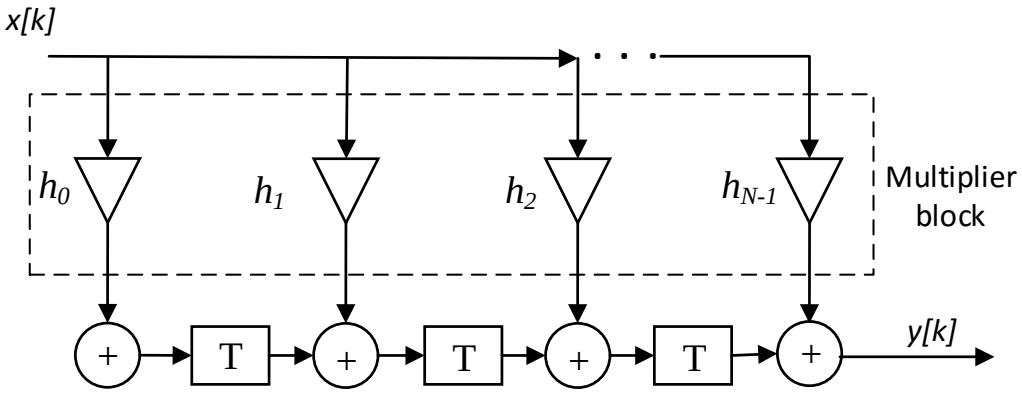


Fig. 1.3. FIR filter in transposed direct form

1.2.1 Linear Phase Filters

The equations 1.1-1.3 represent general FIR filters with 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 the following table.

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

Table 1.1. Symmetric filters

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.

$$\begin{aligned} &\text{Type I: } h_k = h_{N-k+1}, \ N \text{ is odd} \\ &\text{Type II: } h_k = h_{N-k+1}, \ N \text{ is even} \\ &\text{Type III: } h_k = -h_{N-k+1}, N \text{ is even} \\ &\text{Type IV: } h_k = -h_{N-k+1}, \ N \text{ is odd} \end{aligned} \quad 1.4$$

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

Type I:

$$H(\omega) = h(M) + 2\sum_{n=0}^{M-1} h_n \cos((M-n)\omega) \quad 1.5$$

Type II:

$$H(\omega) = 2\sum_{n=0}^{N/2-1} h_n \cos((M-n)\omega) \quad 1.6$$

Type III:

$$H(\omega) = 2 \sum_{n=0}^{M-1} h_n \sin((M-n)\omega) \quad 1.7$$

Type IV:

$$H(\omega) = 2 \sum_{n=0}^{N/2-1} h_n \cos((M-n)\omega) \quad 1.8$$

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 band pass 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:

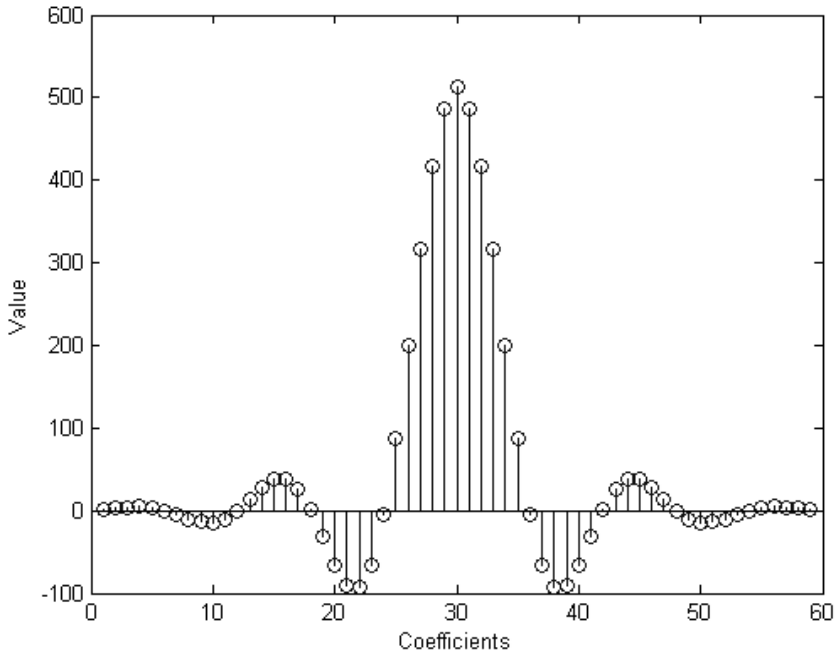


Fig. 1.4. Type I FIR filter coefficients

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:

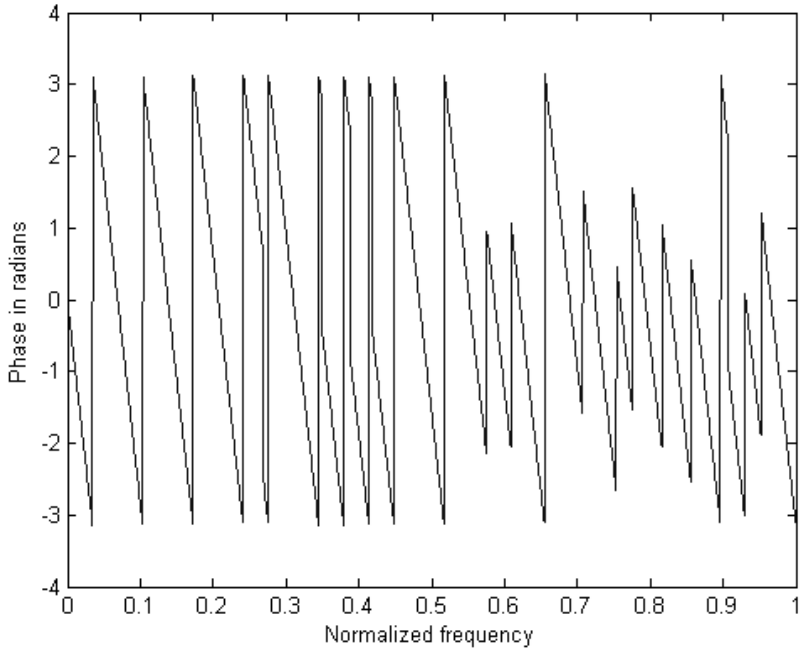


Fig. 1.5. Phase response of a linear phase filter

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.

1.3 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.

1.3.1 Equi-ripple design of linear phase FIR filters

The specifications of the filter are given, before it is designed. Fig. 1.6 shows the specifications of a low-pass filter, which is to be designed based on equi-ripple method.

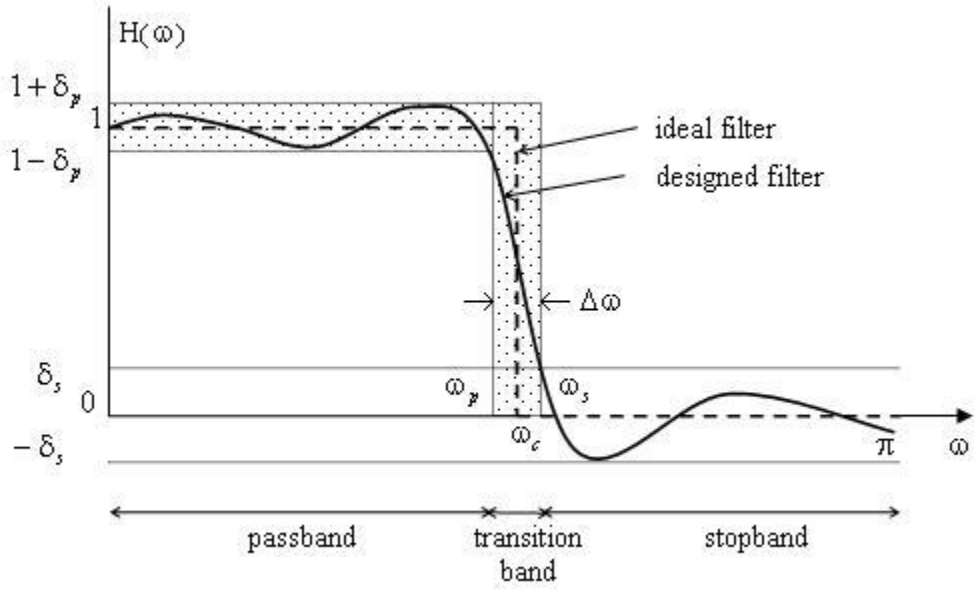


Fig. 1.6. Filter specifications

The broken plot 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. 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:

$$\begin{aligned} &\text{Minimize } \delta \\ &\text{Such that: } 1 - \delta \le H(\omega) \le 1 + \delta, \text{for } \omega \in [0,\ \omega_p] \\ &-(\delta_s\delta)/\delta_p \le H(\omega) \le (\delta_s\delta)/\delta_p, \text{for } \omega \in [\ \omega_s, 1] \end{aligned} \qquad 1.9$$

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

$$H(\omega) = \sum_{n=0}^{\left\lfloor\frac{N-1}{2}\right\rfloor} h(n)\text{Trig}(\omega, n) \qquad 1.10$$

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 1.5-1.8).

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 pass band 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.

1.3.2 Effect of finite precision

In practice due to the finite length of registers in processors, a filter with continuous filter coefficients is not possible to implement. The coefficients must be represented by numbers of finite word-length. This affects the frequency response of the filter negatively.

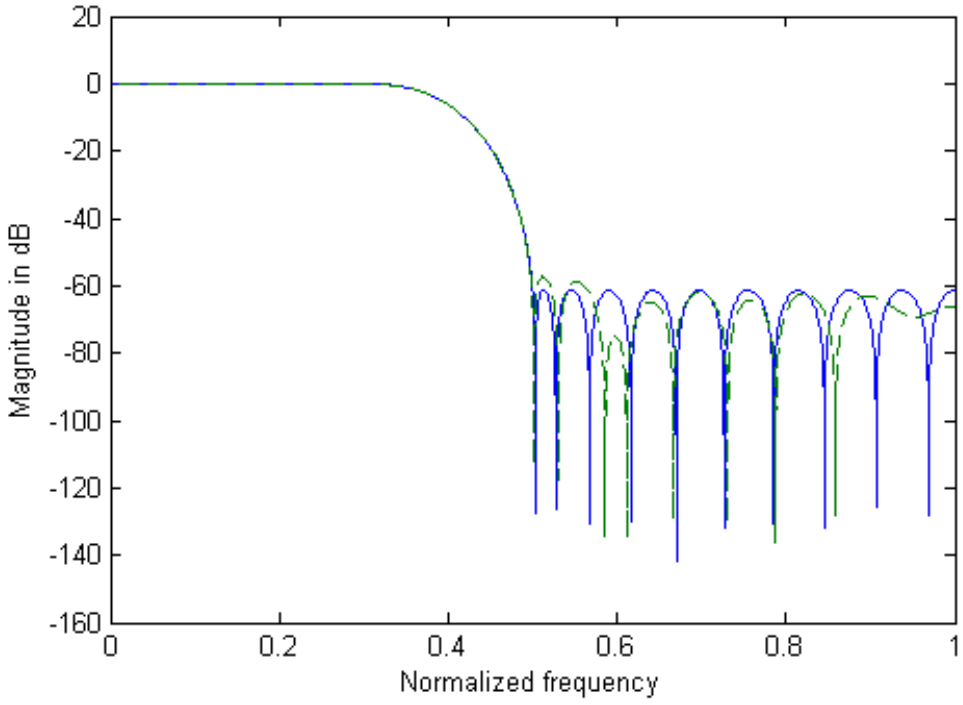


Fig. 1.7. Comparison between finite and infinite precision FIR filter responses

In Fig. 1.7 the blue solid plot represents the response of an optimal un-quantized filter designed by **firpm** function of MATLAB. We can notice that the height of ripples is

constant. After the filter coefficients are quantized to an effective word-length (EWL) of 10 bits excluding the sign bit, the frequency response is again plotted and is shown in green, dashed graph. It is seen that even with 10 bits EWL, there is a significant distortion in the frequency response of the filter, which is clearly noticeable in the stop band of the response. In addition, the equi-ripple character of the response is lost.

1.4 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:

1.4.1 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 $(1000)_2$.

1.4.2 One's compliment 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 compliment 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

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

1.4.3 Two's compliment representation

To avoid the task of adding the carry bit after the addition of two one's compliment numbers, in the two's compliment 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 compliment numbers is straightforward and can be done using normal addition.

1.4.4 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.

1.4.5 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.

1.4.6 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

$$h_n = c_{n1}2^{-b_{n1}} + c_{n2}2^{-b_{n2}} \qquad 1.11$$

Where,

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

A plot of the values which h_n can take is shown in the Fig. 1.8.

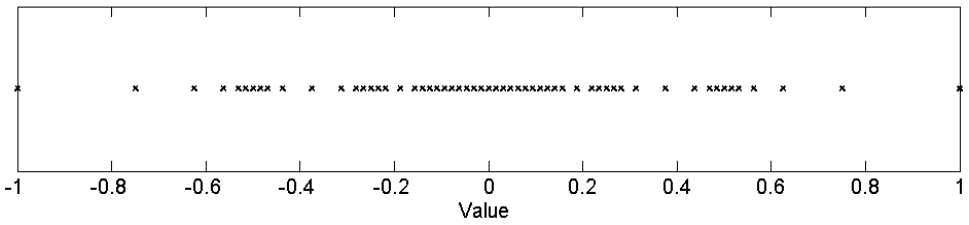


Fig. 1.8. Possible values in a non-uniform quantization scheme

As can be seen, the values are closely spaced near 0 and the gaps between the values generally increases near -1 and 1. Therefore, quantizing larger coefficients results in large quantization errors.

1.4.7 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 with 2^9, we get

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

1.5 Normalized Peak Ripple Magnitude (NPRM)

The equations 1.5-1.8 give the frequency response of the filters. It can be noted that if all the coefficients of the filter are multiplied by a constant K, then the resulting frequency response is identical to the original response except that it is scaled by a factor of K. This is not important in most of the applications, as the filtered signal can be passed through an amplifier with gain 1/K, to cancel the scaling factor. Therefore, in the binary representation of filter coefficients, if all the coefficients are shifted left or right by same number, i.e. they are multiplied by 2, 4, 8, , etc., the resulting filter remains equivalent to the original filter.

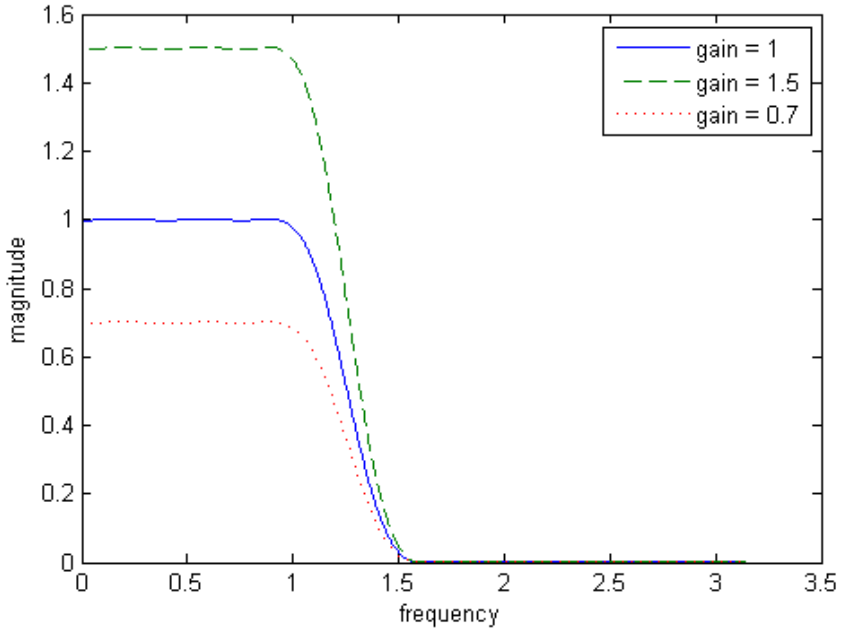


Fig. 1.9. Frequency response with scaling of the coefficients

In Fig. 1.9 blue solid plot represents frequency response of a filter with unity passband gain. If the coefficients are multiplied by 1.5, the frequency response of the resulting filter is identical but scaled by a factor of 1.5, as shown in green dashed line. And similar is the case with scaling with 0.7.

Consider the example given in section 1.4.7, where the coefficients are scaled to integers. The largest coefficient is 234. But it can be easily seen that if the coefficients were scaled such that the largest coefficient lies between 128 and 255, even then the EWL of the filter would remain the same. In other words, if h_{max} is the largest coefficient in terms of magnitude and EWL is the effective word-length, then the gain of the scaled filter can be varied from $\frac{2^{EWL-1}}{h_{max}}$ to $\frac{2^{EWL}}{h_{max}}$.

Therefore, it is possible that the optimum filter is realized while searching in the neighborhood of filters quantized using gain other than 1. Therefore while optimizing the filter coefficients the gain can be allowed to change. In order to do this, the optimization problem is defined as below:

$$\text{minimize: } {}^{\delta}/_{g} = \frac{1}{g}\max_{\omega \in F}|E(\omega)| \qquad 1.12$$

Where, F is a set of frequency points excluding the transition bands.

$$\text{And, } \quad E(\omega) = W(\omega)\big(gD(\omega) - H(\omega)\big) \qquad 1.13$$

Where, $D(\omega)$ is the desired frequency response, which is 1 in passbands and 0 in stopbands, $H(\omega)$ is the frequency response of the designed filter and $W(\omega)$ is a weighting function. The weighting function is used to give different weights to different frequencies. The quantity ${}^{\delta}/_{g}$ is called the Normalized Peak Ripple Magnitude (NPRM) of the filter.

The passband gain g of the designed filter can be calculated using the following equations.

$$g = \frac{P_{max} + P_{min}}{2} \quad \text{if } \frac{P_{max} - P_{min}}{2} > S_{max}$$

$$g = S_{max} + P_{min} \quad \text{if } \frac{P_{max} - P_{min}}{2} < S_{max} \qquad 1.14$$

Where, P_{max} is the maxima of the frequency response in the pass-band, P_{min} is the minima of frequency response in the pass-band and S_{max} is the maxima of frequency in the stop-band.

1.6 Reduced hardware complexity designs

Fig. 1.2 and 1.3 show two forms of hardware implementations of FIR filters. For linear-phase filters the complexity can be further reduced by utilizing the fact that half of the coefficients have same magnitude as the other half. The following figure shows type I filter in transposed direct form, where only the distinct coefficients are used as multipliers,

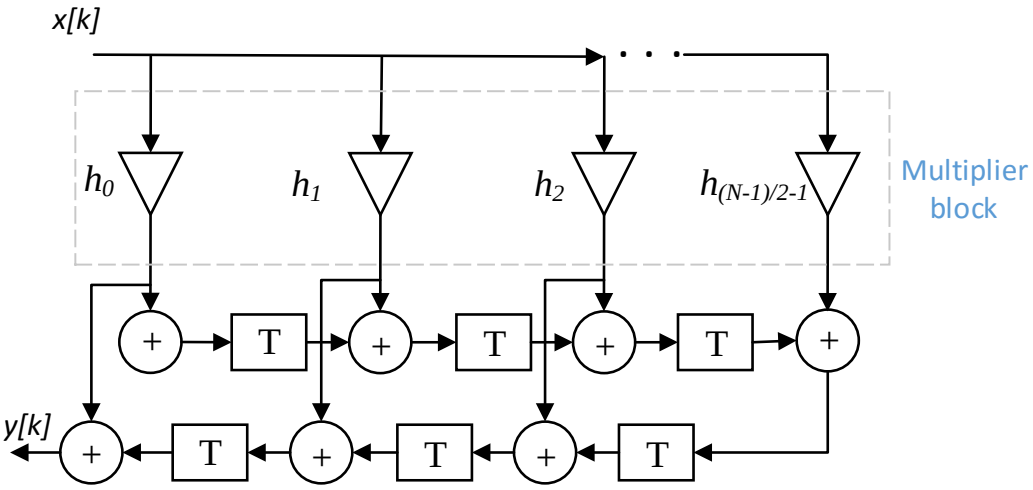


Fig. 1.10. Linear phase type I FIR filter with reduced number of multipliers

1.6.1 Using CSD coding of the coefficients

The coefficients can be converted into CSD format. As CSD representation has minimum number of non-zero binary digits, therefore the complexity of the multiplier block is reduced significantly.

1.6.2 Reduced adder graph designs

Although using CSD instead of simple multipliers can reduce the adder complexity in the multiplier block, yet it is possible to further reduce the number of adders by utilizing the common additive factors among the coefficients. For example the coefficients 5, 9 and 23 can each be realized using 1, 1 and 2 adders (4+1, 8+1 and 32-8-1) respectively, requiring total of 4 adders. But if 23 is realized by utilizing the previously realized coefficients (5 and 9) as $2\times9+5$, then only 1 adder is required for synthesizing 23. (Note that multiplying by 2 requires only shifts and has negligible hardware requirement). Various algorithms [2]-[5] has been developed to find the optimal adder graph, some of which are explained in chapter 2.

1.7 Motivation

The focus of the thesis is the design of linear phase FIR filter which satisfies the given design specifications and also tries to achieve minimum number of hardware adders required for the multiplierless design. In the literature, there are algorithms which can design optimum filters having minimum number of adders, but the run-time of these algorithms usually increases exponentially as the length of the filter to be designed is increased. On the other hand there are some algorithms based on evolutionary methods which can design the filters in relatively less time but the number of adders of the designed filter is usually far from optimum.

In this thesis, the recently proposed Extremal Optimization (EO) is used to design such filters trying to achieve near optimum results and on the other hand avoiding the exponential increase in run-time as the filter length increases.

1.8 Thesis organization

The rest of the thesis report is organized as follows:

In the second chapter, the literature is reviewed focusing on some of the state of the art methods for designing linear phase FIR filters. Some of the important deterministic and heuristic methods are briefly explained. In addition, a couple of algorithms for designing

minimum adder multiplier graphs, such as [2] and RAG-n [4] are explained. In addition, some techniques used to reduce the search space and making the linear program based calculation of the filter coefficients faster are reviewed

In the third chapter, the theoretical background of the extremal optimization (EO) method is given. Also, some major improvements such as τ-EO and population based EO are discussed.

In the fourth chapter, the proposed method is explained in detail. Various steps used in the algorithm like reducing the search space, partitioning the gain and the methods used to speed-up the calculations are discussed.

In the fifth chapter, the analysis of the convergence characteristics of the algorithm is done. Some benchmark filters are designed using the algorithm and their multiplier blocks are synthesized and shown. The run-time of the algorithm is also calculated and discussed and a comparison is made with the state of the art methods.

In the sixth chapter, the conclusion is given.

1.9 Main contributions

The main contribution of the work done here is the implementation of EO algorithm and adapting it to make it suitable for designing FIR filters. The algorithm has not been previously used for the design of digital 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. EO on the other hand has just one adjustable parameter, which is relatively easy to tune. Also, it is easy to tune EO such that it doesn't get trapped in local minima, although at the cost of increasing the runtime.

A specific technique is also developed to make the algorithm fast. The frequency response is not calculated entirely for each objective function evaluation. Instead, only the differential contributions are calculated.

Chapter 2

Review of Literature

Multiplierless digital filters can be realized without the use of multipliers by a shift and add network and thus are efficient in terms of hardware complexity and power consumption. The design goal is to minimize the number of adders. There are both deterministic as well as heuristic methods available for designing finite word length filters. Deterministic methods based on tree search are able to find the optimum solutions but consume lots of time if the filter length is high.

Initially, the design of finite word length filters was concentrated on reducing the representation of filters coefficients. The canonic signed digits (CSD) and Signed Power of Two (SPOT) representation of numbers was utilized to minimize the bits needed to represent the filter coefficients. The minimal bit representation led to a reduction in the number of adders needed to implement the filter either using ripple carry adders (RCA) or using carry save adders (CSA). The RCA topology required less chip area and consumed less power but at the expense of reduced operating frequency. The CSA topology increased the speed at the expense of additional hardware. However, both the topologies had minimal adder counts under the SPOT design criteria.

With time, research efforts in finite word length filter design drifted towards developing efficient multiple constant multiplication (MCM) algorithms. The FIR filter in its transposed direct form could be modelled as a case of MCM wherein the input is multiplied by all the filter coefficients and then saved in registers and added along the delay line. The MCM algorithms utilized the redundancies in the filter coefficients and modelled the shift and add network as a one-input-*M*-output network (*M* being the number of unique filter coefficients). The MCM algorithms were either graph based [3]-[4], or were pattern based [2].

MCM algorithms produced the minimal adder count for a given set of filters coefficients but given a filter specification, the coefficient set that can meet the requirement is not unique. Thus, adder counts of other coefficients may be less and hence the filter design

problem and the implementation were consolidated. This led to the dynamically expanding subexpression space design methods.

Deterministic methods based on tree search are able to find the optimum solutions at the expense of exponential computational complexity [6]. Thus, design of large order and high word length filters becomes infeasible. The research in deterministic methods is being carried out to reduce the search space by excluding the section of the search space from the algorithm that shows no promise of feasible solutions. However, the exponential computational growth is unavoidable in all tree search methods and a polynomial time algorithm can only be guaranteed if every node produces only one child. In [7], a method is proposed whose runtime is polynomial with the filter length. In this method the passband gain is divided into large number of sections. In each section a solution is found by successively finding the feasible range of a coefficient and fixing it the value near the middle of the range. A feasible solution is chosen which results in minimum number of adders.

In heuristic methods for designing finite word length filters, the most common strategy is to find an initial solution, for example by rounding the continuous solution or by finding an initial good solution with greedy optimization and then search using small number of values around the rounded values. Totally random search with no initial solutions can only yield feasible solutions for filter orders less than 30. The shift and add network can be constructed using MCM algorithms and the objective function is created to direct the search for finding minimal adder coefficients.

2.1 Filter design algorithms

In the following section some important deterministic and heuristic algorithms from literature are reviewed.

2.1.1 Deterministic algorithms

1. In [8], a branch and bound algorithm (called FIRGAM) is proposed for the design of hardware efficient filters. The algorithm designs filters with reduced number of SPT terms. The coefficients are designed by finding the feasible range of each coefficient successively

and fixing the coefficient first to the quantization value nearest to the center of the feasible range. After a coefficient has been fixed the feasible range of the next coefficients is recalculated and the same procedure is repeated until all the coefficients has been successfully quantized.

The number of SPT terms of the solution are calculated and the algorithm then goes through the search space again to find if a solution with lesser number of SPT terms can be found.

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.

2. [6] proposes an algorithm to design optimum filters in terms of the number of adders. The method involves mixed integer linear programming (MILP). The filter coefficients are synthesized based on a dynamically expanding subexpression space.

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

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

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.

This algorithm is guaranteed to return 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.

3. In [7], a polynomial time algorithm is proposed. The method is inspired from FIRGAM [8] algorithm. In contrast to FIRGAM, the proposed method fixes each coefficient to the middle of its range and does not try to expand the search in the neighboring values.

As the runtime of linear program is polynomial in number of filter coefficients and the number of linear programs to be solved also increases linearly with the number of the filter coefficients, therefore the above algorithm is polynomial in runtime.

In this algorithm, first some of the coefficients are fixed to 0 before proceeding with the remaining algorithm. The reason is that, if a coefficient is 0, then there is an immediate reduction in two adders in the delay chain.

After this, the passband gain is divided into number of partitions and the tree search algorithm, as in [8], is used to fix successively the coefficients to the middle of their feasible range. The filter coefficients from each gain are synthesized using an MCM algorithm [2] and the one which yields the result with minimum number of adders is chosen as the final solution.

4. [9] proposes a polynomial time algorithm which optimizes the coefficients in two steps. The first step is similar to the one proposed in [7]. After the initial solutions for each gain are obtained, the ones which are feasible are selected for the second step of the optimization.

In the second step the coefficients are divided into groups. Each group has 20 coefficients (the last group might have less than 20 coefficients). Each group is optimized one by one using a tree search algorithm similar to [6]. Expanding subexpression technique is used to synthesize the coefficients as they are constructed to yield minimum number of adders. Once all the groups have been optimized, the process is repeated again several times starting from the first group. After 2-3 iterations the algorithm converges. The solution at the gain which yields the best result in terms of number of adders is chosen as the final solution.

The above algorithm is capable of finding solutions which, in most of the cases, are optimal or near optimal. Also, the runtime of the algorithm is much less than the optimal algorithms.

2.1.2 Heuristic algorithms

1. In [10], 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 sum of power of two terms (SOPT) is used, the quantization is done such that the quantized value is the nearest value in the domain of SOPT from the continuous value.

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.

2. In [11], a genetic algorithm is proposed for the design of multiplierless filters. In this paper the search space is partitioned into a number of search spaces. This is done by dividing the passband gain into a number of partitions, such that the EWL remains same as specified. The continuous filter is constructed for each of the partition. Then a search space is constructed around each solution and GA is used to find feasible solution in this space having minimum hardware adders. The search space around the nth coefficient is defined in the following way:

$$h_{qm}^{u}(n) = h'_{qm}(n) + 2^{ceil(B_m(n)/3)} - 1$$
$$h_{qm}^{l}(n) = h'_{qm}(n) - 2^{ceil(B_m(n)/3)} + 1 \qquad 2.2$$

Where, h'_{qm} are the quantized coefficients scaled to integers. h_{qm}^{u} is the upper limit of the coefficients and h_{qm}^{l} is the lower limit of the coefficients. $B_m(n)$ is the EWL of the nth coefficient.

3. Another GA based algorithm is proposed in [12] for the design of multiplierless filters. In this method, adaptive crossover and mutation rates are used, which prevents premature

convergence of the algorithm and also avoids the good solutions to be cast away. The gain is divided into a number of partitions as in [11]. In the second part of the paper the algorithm is used to design filters in cascade form.

4. In [13] a two stage algorithm is proposed with sums of SPT terms. In the first stage, a time domain method is used to assign SPT terms to the filter coefficients. In the second stage a Verterbi's algorithm type method is used. The problem is cast as dynamic programming and a trellis search is done to iteratively add SPT terms one by one. The number of adders are further reduced by using a subexpression sharing method to exploit the redundancies among the coefficients.

5. A gradient based method, is used in [1]. In this method, the gradient information is calculated and used to direct the search. The search is directed towards low gradients routes first and when further improvement stops, the search is redirected to the steeper routes. The method is semi-random in nature and has low computational cost.

2.2 MCM algorithms

In the following sub-sections two important MCM algorithms for designing multiplierless realizations of filter coefficients are explained.

2.2.1 MCM algorithm of A. Yurdakul and G. Dündar [2]

There are a number of algorithms in the literature to find the multiplierless realizations of filter coefficients. RAG-n and C1 algorithms are notable for producing realizations with optimal number of adders in most of the cases. But these algorithms suffer from the disadvantage of large memory requirements due to the use of large lookup tables. Also, when the word-length is higher, the lookup tables become prohibitively large. In this case, the algorithms can still construct these coefficients using heuristics, but do not guarantee the optimal number of adders. A fast and efficient algorithm has been introduced in [2]. It has pseudo-polynomial run-time and memory requirements in the worst case scenario.

The algorithm is based on pattern search in the binary or CSD values of the coefficients. The pattern search is done at length of 2 throughout the algorithm. Pattern length is defined as the number of non-zero terms in the coefficient. For example, if a coefficient of effective

word-length 8 is given as 11010110. Then, 1001 and 101 shown in bold as **1**10**1**0110 and 110**1**0**1**10 respectively are patterns of length 2, whereas 1101 and 100101 are patterns of length 3 shown in bold as **11**0**1**0110 and **1**10**1**0**1**10 respectively. The algorithm iteratively combines 2 non-zero terms to generate all the coefficients. The step by step realization of constant multiplication is shown in the following figure:

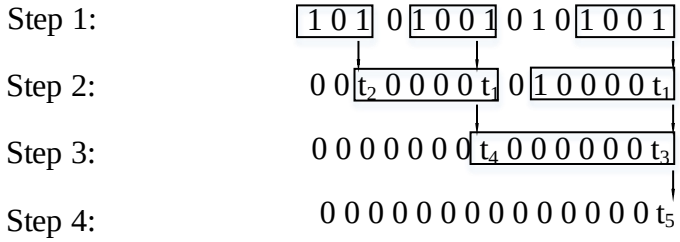


Fig. 2.1. Constant multiplication realization using the algorithm.

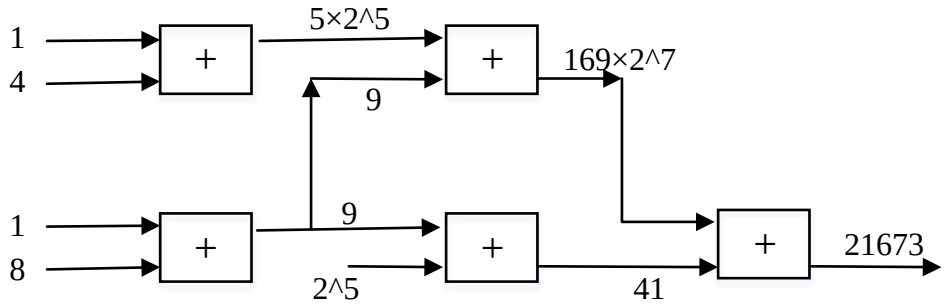


Fig. 2.2. The adder tree for the above example

From the figure, we can see that the number of adders required for the synthesis is 5.

2.2.2 RAG-n algorithm

RAG-n algorithm [4] synthesizes the coefficients in two steps. The first part of the algorithm is optimal and the second part is heuristic. If the algorithm is able to synthesize all the coefficients in the first step, then the synthesized set is guaranteed to be optimal. The algorithm uses a lookup table constructed by MAG algorithm. The lookup table contains the minimum adder graphs of all the integers up to a certain cost, where cost is the number of additions or subtractions required to synthesize that number. Only the odd integers are considered, because the even ones can be constructed from the odd ones by

multiplying with 2, which is cost-free. For example, 2, 4, 8, and so on have cost 0 as these can be constructed without any adders. 3, 5, 7, 9, etc. have cost 1 as they require one adder for their synthesis as shown below:

4 - 1 = 3, 4 + 1 = 5, 8 - 1 = 7, 8 + 1 = 9

The basic steps of the algorithm are as follows:

- First, all the coefficients are reduced to odd-fundamentals. This is done by making any negative coefficients positive. Then making the coefficients odd by successively dividing by 2.
- Any repeated values are removed. The values 0 and 1 are also removed from the set.
- All the odd-fundamentals which have cost 1 are removed from the set. The set of remaining fundamentals is called incomplete set. The number of adders used until now is the number of fundamentals removed. The set of removed fundamentals is called complete set.
- Use the complete set and try to construct fundamentals in the incomplete set by adding two coefficients from the complete set and their multiples with power of 2. The fundamentals which are able to be constructed in this step are added to the complete set and removed from the complete set.
- The step 4 is repeated until there are no more fundamentals in the incomplete set or no more fundamentals in the incomplete set can be constructed.

If by this time the incomplete set is empty then the algorithm stops and the resulting synthesis is optimal, otherwise the remaining fundamentals are constructed using the second part of the algorithm which is heuristic.

As the number of coefficients increases for a given word-length the chances of a set constructed by the algorithm to be optimal increases. For example, when the word-length is 8, the chances of optimal set are already more than 50% with a set size of around 10 and almost 100% with a set size of 20. On the other hand, when the word-length is 12, around 60 coefficients are required to reach 50% chances of optimality.

2.3 Speeding up the linear program

The maximum number of extrema (i.e. maxima and minima) in the frequency response of a filter of length N is equal to

$$N_{ext} = \left\lfloor \frac{N+1}{2} \right\rfloor \tag{2.3}$$

The above number includes the points at frequencies $\omega = 0$ and π, but does not include the frequencies at the edge of the transition bands.

The filter coefficients for a low pass filter with pass and stop band cut-off frequency ω_p and ω_s respectively, can be evaluated by solving the following linear program

$$\begin{aligned} &\text{minimize: } \delta \\ &\text{subject to: } 1-\delta \le H(\omega) \le 1+\delta, \quad \text{for } \omega \in [0, \omega_p] \\ &-\frac{\delta_s}{\delta_p}\delta \le H(\omega) \le \frac{\delta_s}{\delta_p}\delta, \quad \text{for } \omega \in [\omega_s, \pi] \end{aligned} \tag{2.4}$$

Where, $\frac{\delta_s}{\delta_p}$ is the ratio of stop and pass band error tolerances and $H(\omega)$ is the frequency response of the filter.

In order to account for all the frequencies, the number of points on the frequency grid need to be as large as possible. If the number of points is very large, then the number of constraints of the LP is also very large, which can create instability in the solving algorithm and also require large computational time. It can be noticed that the only points where optimization is required are the extremal points on the frequency response. The problem is that the extremal points are not known a priori.

In [14], a method, similar to the one used in Parks-mcCllellan is used to solve the above equations. The algorithm is explained below:

1. N_{ext} number of points are initially chosen in the pass and stop band which are roughly equally spaced.
2. The equations are solved at these points, in addition to pass and stop band cut-off frequencies, to get filter coefficients.
3. Frequency response is evaluated using these filter coefficients at roughly $2N$ equally spaced points on the frequency grid.
4. Newton's extrapolation method is applied to accurately determine the extrema of the frequency response, which are the new N_{ext} number of points.
5. The steps 2-4 are repeated until the change in the extrema points is less than a given tolerance.

2.4 Reducing the search space

If the magnitude of the coefficients are quantized to an EWL of B bits and the filter length is N, then the total number of possible filter realizations are equal to $(2^{B+1})^N = 2^{(B+1)N}$. Even for a moderate word-length of 8 and filter length 31, the number of possibilities is $2^{279} \approx 10^{84}$, which is enormous. It is impossible to check each and every possibility by Brute force to optimize the filter.

A large majority of these possibilities give filters which deviate greatly from the desired frequency response. The optimum filter coefficients lie close to the quantized filter coefficients. Therefore it makes sense to look for the optimum filter coefficients in the vicinity of the quantized filter coefficients. For example in [10], a neighborhood is selected around each coefficient, which is much less than the full range of the given coefficient. This way each filter's neighborhood is encoded with a number of bits which is less than the EWL of the given coefficient. In this way the search space is drastically reduced.

Instead of quantizing the coefficients and then defining an arbitrary search space as explained above, a more accurate method is to find the range of each coefficient value

which satisfy the given filter specifications. This is done by solving the following LP for each of the coefficients. To find the feasible range of i^{th} coefficient, solve

$$\text{minimize: } f_0 = h(i) \text{ and } f_1 = -h(i)$$

$$\text{subject to: } \left(1 - \delta_p\right) \leq H(\omega) \leq \left(1 + \delta_p\right), \text{for } \omega \in \left[0, \omega_p\right] \qquad 2.5$$

$$-\delta_p \leq H(\omega) \leq \delta_p, \text{for } \omega \in [\omega_s, \pi]$$

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

The above equations represent two linear programs, one minimizing $f_0 = h(i)$ and the other minimizing $f_0 = -h(i)$. The results of these give the minimum and maximum value of the i^{th} coefficient. The search space of the filter can now be limited to the feasible range of each coefficient.

Chapter 3

Extremal Optimization

Extremal Optimization is a recent heuristic method inspired by models of co-evolution such as Bak-Sneppen model [19]. It is one of many evolutionary algorithms, such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Bee Colony Optimization, etc. Evolutionary algorithms can successfully optimize a wide variety of optimization problems as they do not make any assumptions about the underlying fitness landscape of the problem. Therefore, these have been widely used in the optimization of engineering problems, which are otherwise very difficult to solve due to the complex nature of the fitness landscape, or due to the huge size of the search space which is intractable with other algorithms.

GA algorithm is inspired from the Darwinian natural selection. It consists of basic steps of biological evolution, which are inheritance, crossover, mutation and selection. [10]-[12] and [15]-[17] use GA based algorithms for design of FIR filters. One of the limitations of GA is the tendency to converge to local optima instead of global optima.

S. J. Gould [18], suggested that evolution of species takes place intermittently, instead of in a gradual way. The evolution of a species affects its neighboring species and instead of evolving towards an equilibrium state, they enter a state in which intermittent bursts of evolution take place separated by quite states. The changes resemble avalanches. The sizes of the avalanches come in all sizes and are scale free. In other words, there are sudden large changes which are rare in frequency like large extinction events and small changes which occur with high frequency. This state of the system is called self-organized critical state.

In genetic algorithms (GA) better solutions are produced from previous solutions by selectively breeding good solutions. In contrast, in extremal optimization the bad components of the solution are successively eliminated in order to achieve a better solution. In order to achieve optimum solutions in GA, the parameters such as crossover and mutation rate should be selected properly. This is often hard to achieve and requires

experienced human intervention. On the other hand the original EO lacks any adjustable parameters and thus can be implemented easily on wide variety of problems.

EO can be compared to Simulated annealing (SA) which also works on a single solution. In SA, initially the magnitude of the changes is high and as the time progresses the magnitude of changes decreases. After a long time the algorithm converges and the algorithm sticks to the minimum which it has found at that place. In contrast, in EO the changes occur forever. The probability of a change of a given magnitude remains the same throughout the algorithm. Also, the probability of the changes is inversely related to the magnitude of the changes. In other words, a change of small magnitude has higher chances of taking place and of larger magnitude has lower chances of taking place. Due to this, EO does not stick to a local minimum forever and always has chances of escaping a minimum of any depth.

3.1 Bak-Sneppen Model

Per Bak and Kim Sneppen [19] proposed a model of evolution in which they say that an entire species has a single fitness level. Multiple species affect the fitness of their neighboring species and therefore in turn affect their evolution.

The species evolve only if, after mutation, the new configuration has higher fitness. There is some low probability of mutations with lower fitness, allowing the evolution to evolve out of the local minima of the fitness landscape.

3.2 Extremal Optimization

Extremal optimization was proposed by Stefan Boettcher and Allon Percus [20], [21] and is based on Bak-Sneppen model of co-evolution. The method successively eliminates bad components of a single solution, instead of breeding better solutions like GA. In the paper, the method has been successfully applied on graph partitioning and travelling salesman problem. The graph partitioning problem is NP hard, in which a graph has to be divided into 2 sub-sets, such that minimum number of edges cut through the two sub-sets. The results of large graphs are compared with those obtained from GA and SA (Simulated

annealing). It is observed that EO obtains optimal results in much shorter time for large graphs. The EO can be explained in the following steps:

1. An individual is generated randomly by randomly initializing its n variables in their respective feasible ranges. Let's call this individual S. Set the optimal solution $S_{best} = S$.

2. For the current individual S:

(a) Evaluate the fitness λ_i for each decision variable, $x_i \in (1, \dots, n)$

(b) Find j satisfying $\lambda_j < \lambda_i$, for all i, i.e., x_j is the worst variable.

(c) Choose $S' \in N(S)$ such that x_j must change its state, where, $N(S)$ is the neighborhood of S.

(d) Accept $S = S'$ unconditionally.

(e) If the current cost function value is less than the minimum cost function value, i.e. $C(S) < C(S_{best})$, then set $S_{best} = S$.

3. Repeat Step 2 as long as desired.

4. Return S_{best} and $C(S_{best})$.

3.3 Generalized EO

In [22], a generalized EO algorithm is presented, which can be applied to broad class of engineering problems. In an optimization problem consisting of N design variables, each one is randomly initialized and represented in fixed number of binary bits. All these binary numbers are joined to form a string.

1. Let's call this individual S. Set the optimal solution $S_{best} = S$. Let C be the fitness of this individual. Set $C_{best} = C$.

2. For the current individual S:

(a) Evaluate the fitness λ_i for each decision variable, $x_i \in (1, \dots, n)$ of S. This is done by flipping each i^{th} bit of S. The resulting fitness of the individual S' with the flipped bit is calculated. Let C' be its fitness. Then the fitness of the bit is $C' - C$.

(b) Rank the fitnesses of all the bits with rank $k = 1$, for the best fitness and $k = n$ for the worst fitness.

(c) Select a random rank k, with probability distribution of k, proportional to $k^{-\tau}$, where τ is a constant.

(c) Choose $S' \in N(S)$ such that x_k must change its state, where, $N(S)$ is the neighborhood of S,

(d) Accept $S = S'$ unconditionally,

(e) If the current cost function value is less than the minimum cost function value, i.e. $C(S) < C(S_{best})$, then set $S_{best} = S$.

3. Repeat Step 2 as long as desired.

4. Return S_{best} and $C(S_{best})$.

3.4 Population based EO

A population based EO algorithm is proposed in [23]. It claims to have higher search efficiency than the traditional EO. In this method instead of having only one individual, we have a number of individuals as in GA. The basic steps of this algorithm are as below.

1. Generate m individuals S, $i \in (1, \dots, m)$. Find the individual with best fitness and set S_{best} equal to this individual. Let C be the fitness of this individual. Set $C_{best} = C$.

2. For each individual S:

(a) Evaluate the fitness λ_i for each decision variable, $x_i \in (1, \dots, n)$ of S. This is done by flipping each i^{th} bit of S. The resulting fitness of the individual S' with the flipped bit is calculated. Let C' be its fitness. The fitness of the bit is $C' - C$.

(b) Rank the fitnesses of all the bits with rank $k = 1$, for the best fitness and $k = n$ for the worst fitness.

(c) Select a random rank k, with probability distribution of k, proportional to $k^{-\tau}$, where τ is a constant.

(c) Choose $S' \in N(S)$ such that x_k must change its state, where, $N(S)$ is the neighborhood of S,

(d) Accept $S = S'$ unconditionally,

(e) If the current cost function value is less than the minimum cost function value, i.e. $C(S) < C(S_{best})$, then set $S_{best} = S$.

3. Repeat Step 2 as long as desired.

4. Return S_{best} and $C(S_{best})$.

Chapter 4

Proposed Method

The original EO has very poor global search properties and gets stuck in local minima very easily. τ-EO is a significant improvement over the original EO and is used in this work. This algorithm has just one adjustable parameter called τ, whose optimum value in wide variety of problems lies in the range from 0.5 to 3 and thus easy to adjust. τ-EO never gets trapped in a local minima and thus given enough time it can potentially find the optimum solution. In actual world problems with limited computation time, good results can be achieved by choosing τ properly.

As, the purpose of the algorithm developed in this thesis is to reduce the number of adders required for the synthesis of the coefficients, therefore the objective function for the EO algorithm is chosen in such a way that it decreases with the number of adders. The RAG-n algorithm used in our method is very time consuming. Therefore, it is not used within the objective function until the algorithm has found the feasible solutions. The objective function is shown below.

$$Obj = \begin{cases} \delta_m, & \delta_m > \delta_p \\ \delta_p * \dfrac{n}{n_{max}}, & \text{otherwise} \end{cases} \qquad 4.1$$

δ_m is the minimax error of the current solution in the current iteration. n is the number of adders found using RAG-n for this solution and n_{max} is a constant which is greater than the minimum number of adders achieved in the first step.

The above objective function has the property that as long as error δ_m is greater than the specified passband error δ_p, the objective function is equal to δ_m. Once the error becomes less than δ_p, the objective function depends only on the number of adders n. Using this type of objective function the algorithm first tries to decrease the passband error. Once the passband error decreases below the specified value the algorithm starts trying to decrease

the number of adders. n_{max} in the objective function is taken as the number of adders achieved in the first step of the algorithm. The idea here is that $n < n_{max}$ as we are trying to improve the first step solution. Due to this the objective function is monotonically decreasing function in δ_m.

The algorithm of this thesis is explained below:

Step 1. Floating point filter coefficients are calculated with some coefficients fixed to 0. This step is explained in detail in section 4.2.

Step 2. As floating the passband gain results in a filter with a scaled version of the frequency response, therefore the feasible gain is partitioned into a number of partitions. If h_{max} is the coefficient with maximum magnitude in the floating point filter coefficients and EWL is the effective word-length of the filter, then the feasible range of the gain is $\left[\frac{2^{EWL-1}}{h_{max}}, \frac{2^{EWL}}{h_{max}}\right]$. The details are given in section 4.3.

Step 3. An initial fixed point solution is found in each of the gains by, **a)** simple quantization, or **b)** quantizing each coefficient one by one and re-optimizing the rest of the coefficients or **c)** finding the feasible range of each coefficient one by one and quantizing the coefficient in the middle of its range and re-optimizing the rest of the coefficients. In the thesis the third approach is used. The details are given in section 4.5.

Step 4. The EO algorithm is used to further improve the above solutions.

The EO algorithm is explained below in detail:

Step 1. For a given passband gain, let $h = \{h_0, h_1, \dots, h_N\}$ be the initial fixed point coefficients. A neighborhood $R = \{r_0, r_1, \dots, r_N\}$ is defined around each coefficient. Therefore, each coefficient can take value in the range $[h_n - r_n, h_n + r_n]$. The search space of EO is limited to this neighborhood. The details of deciding the neighborhood are given in section 4.4.

Step 2. Initialize, the solution $h_{sol} = h$.

Step 3. Calculate the objective function by changing each coefficient in its range and keeping others constant. If there are l number of points obtained, then there are l values of the objective function given by $\{Obj_1, Obj_2, \dots, Obj_l\}$. Sort these values in ascending order.

Step 4. Select an integer p from 1 to l randomly.

Step 5. Select another random number *rand* from 0 to 1.

Step 6. If $rand < p^{-\tau}$, select the point in the search space corresponding to the p^{th} value of the sorted objective functions and replace h_{sol} with this point, else go to step 4 until a replacement is achieved.

Step 7. Repeat steps 3 to 6 until desired number of iterations.

The variation of cost function with the iterations is shown in the Fig. 4.1. It can be noticed that even late during the evolutionary process there are large fluctuations in the cost function. This is an important property which helps EO in exploring wider search space and avoid getting stuck in local minima.

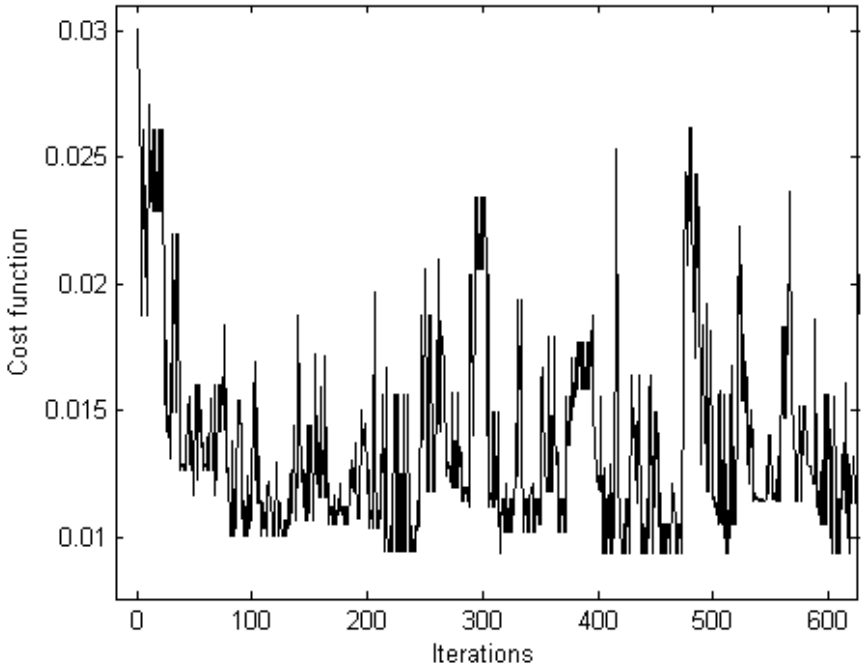


Fig. 4.1. Variation of cost function with iterations

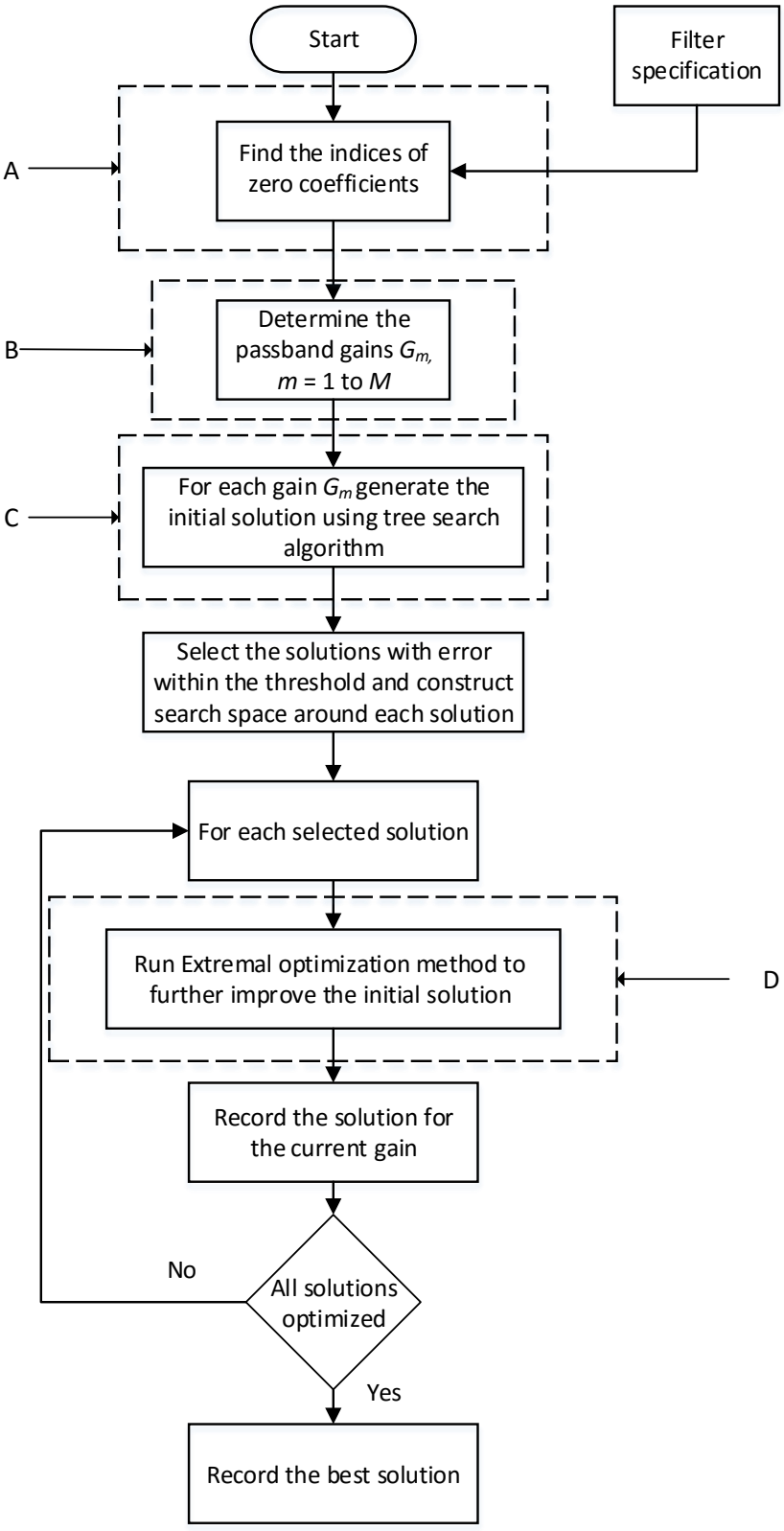
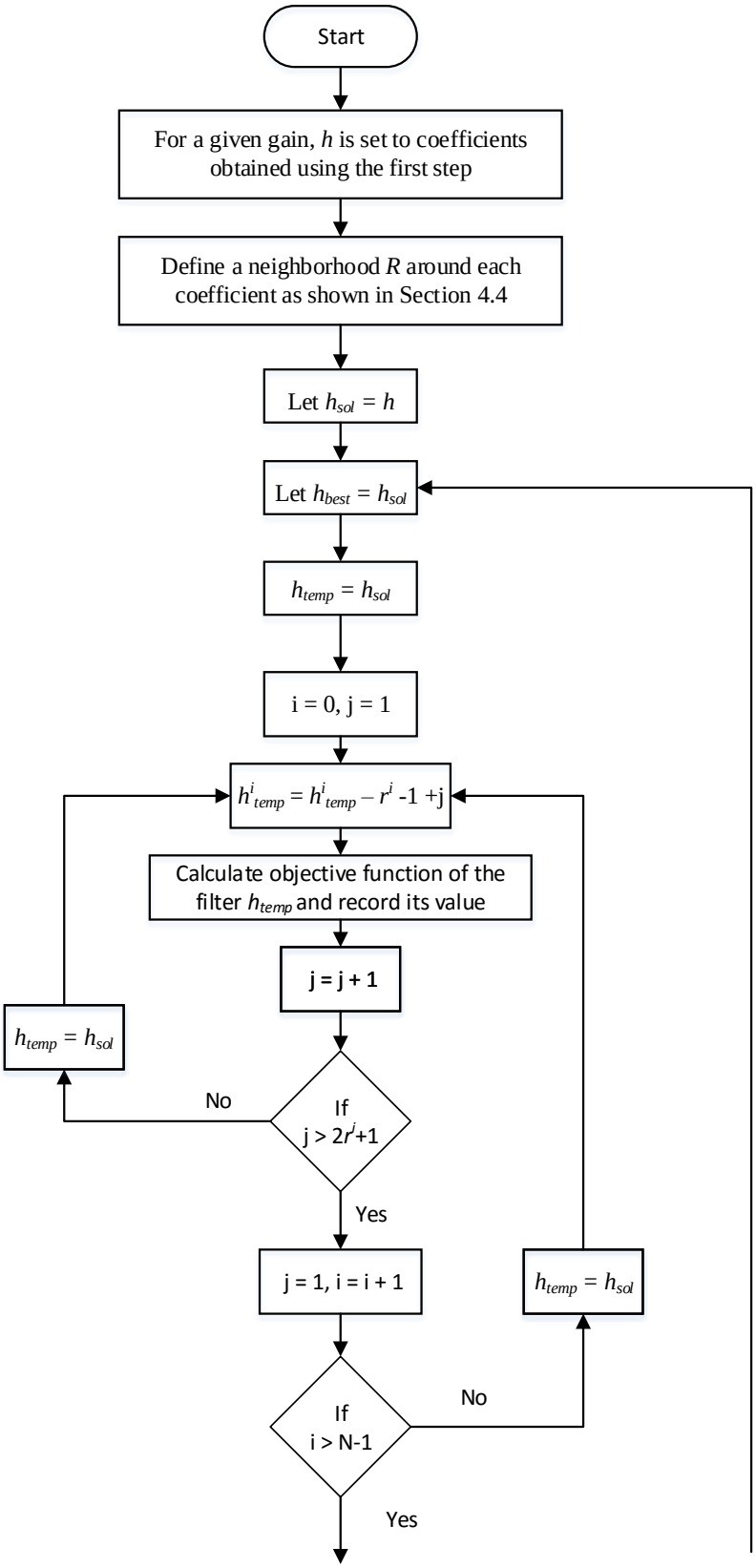


Fig. 4.2. Flowchart of the algorithm


Start
For a given gain, h is set to coefficients obtained using the first step
Define a neighborhood R around each coefficient as shown in Section 4.4
Let $h_{sol} = h$
Let $h_{best} = h_{sol}$
$h_{temp} = h_{sol}$
i = 0, j = 1
$h^i_{temp} = h^i_{temp} - r^i$ -1 +j
Calculate objective function of the filter h_{temp} and record its value
j = j + 1
If $j > 2r^i+1$
No
$h_{temp} = h_{sol}$
Yes
j = 1, i = i + 1
If i > N-1
No
$h_{temp} = h_{sol}$
Yes

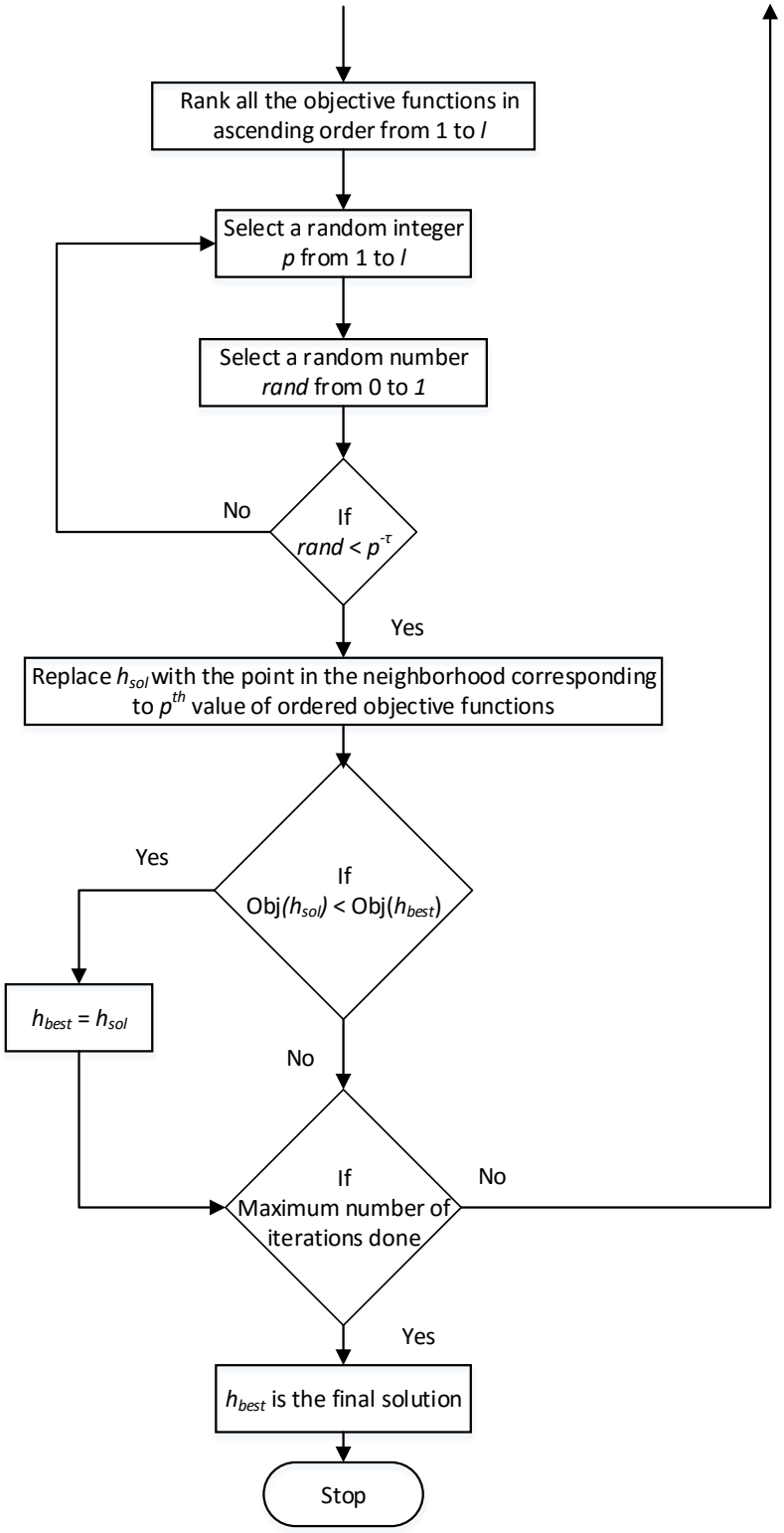


Fig. 4.3. Extremal optimization algorithm

In the flowchart shown in Fig. 4.2, the parts marked with A, B and C are explained in more details in sections 4.2, 4.3 and 4.5 respectively. The part D has been explained in previous section.

4.1 Finding the filter order and word-length

From the filter specifications, first a continuous coefficient filter is constructed using **firpm** function of MATLAB. This is done by successively trying higher orders until the specifications are met. The filter order for the final design is initially chosen to be 2-4 higher than the continuous one, although this can vary in the final design. The effective word-length is chosen to be as small as possible. Initially a value of 8 is chosen for the word-length. The first step of the algorithm is run after fixing the zeros as explained in the next section. If no feasible solution is found, the filter length is increased by one and the process repeated. If after increasing the filter-length several times no feasible solutions are found, then the effective word-length can be increased.

Once feasible solutions are obtained, the algorithm can be run to find the best solution in terms of adders. Next, a higher effective word-length can be tried, as it might be possible to achieve fewer adders by doing this. The procedure to fix the word-length and the filter order is explained in the flowchart in the Fig. 4.4.

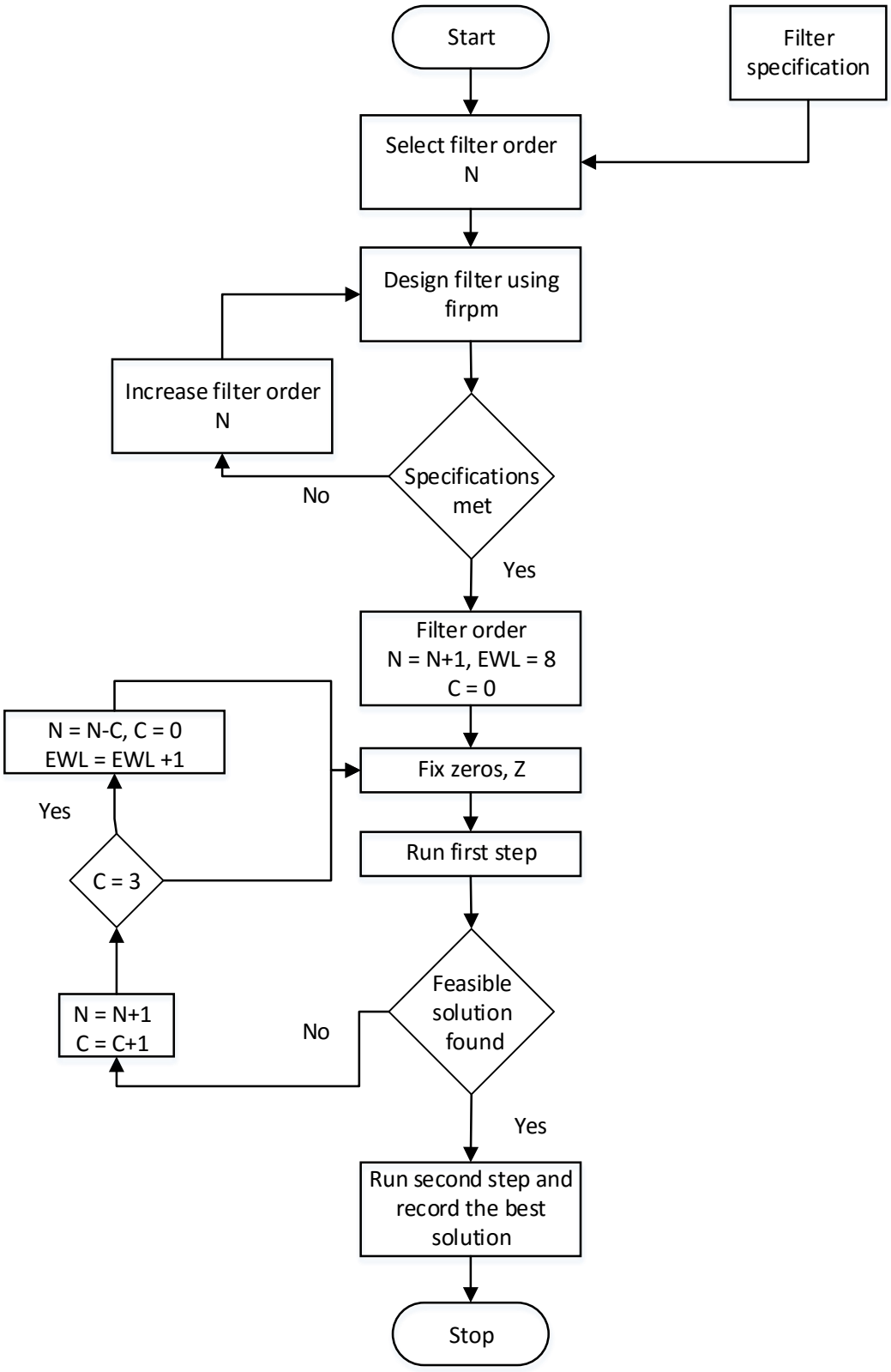


Fig. 4.4. Estimating filter order and word-length

4.2 Fixing some coefficients to zero

The fixing of coefficients to 0 is done as follows:

- The set Z, denoting the index of the coefficients whose value is 0, is set to empty.
- The following LP is solved to find the filter coefficients and the corresponding minimax error.

$$\begin{aligned} &\text{minimize: } \delta \\ &\text{such that: } 1-\delta \le H(\omega) \le 1+\delta, \text{for } \omega \in [0, \omega_p] \\ &-(\delta_s \delta)/\delta_p \le H(\omega) \le (\delta_s \delta)/\delta_p \text{ , for } \omega \in [\omega_{s,} \pi] \end{aligned} \qquad 4.2$$

Where

$$H(\omega) = \sum_{n=0,\ n \notin Z}^{\left\lfloor \frac{N-1}{2} \right\rfloor} h(n)\mathrm{Trig}(\omega, n) \qquad 4.3$$

And $\mathrm{Trig}(\omega, n)$ is an appropriate trigonometric function depending on the filter type and is given by equations 1.5-1.8.

- Find the coefficient with minimum absolute value and note its index k. Add the index in the set Z. Keep on repeating the second and the third step as long as the resulting error δ is less than the specified passband error δ_p.

The flowchart of the algorithm to fix the zero coefficients is shown in the Fig. 4.5.

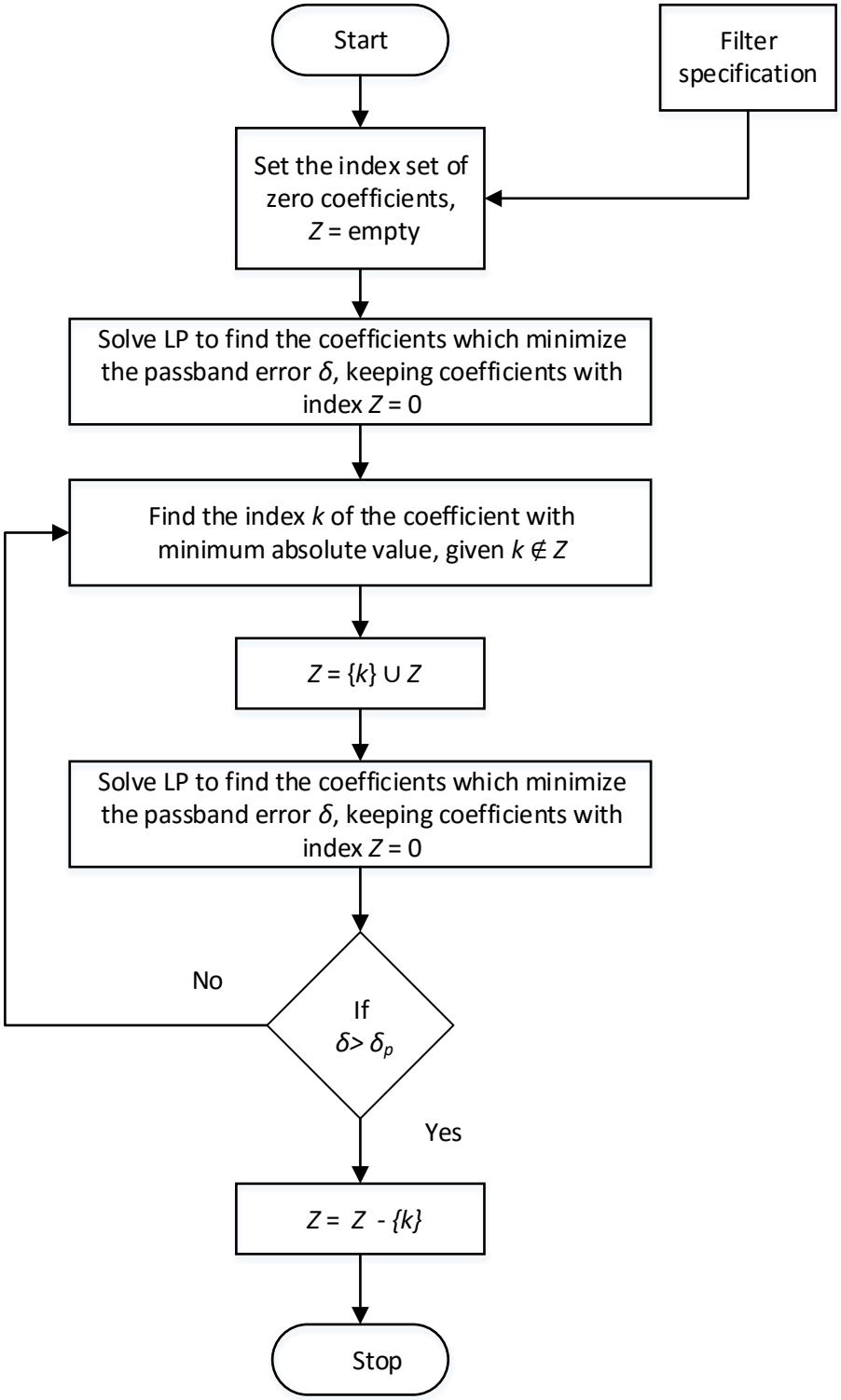


Fig. 4.5. Flowchart of the algorithm for fixing the zero coefficients

4.3 Partitioning the gain

To reduce the search space, the passband gain can be divided into smaller partitions in $\frac{2^{EWL-1}}{h_{max}}$ to $\frac{2^{EWL}}{h_{max}}$. Let's call $g_{min} = \frac{2^{EWL-1}}{h_{max}}$ and $g_{max} = \frac{2^{EWL}}{h_{max}}$. If the range is partitioned into M partitions, the i^{th} partition is defined by the range $[\gamma_{min}^{i}, \gamma_{max}^{i}]$.

$$\gamma_{min}^{i} = g_{min} + \frac{(i-1)(g_{max} - g_{min})}{M}, \text{and}$$

$$\gamma_{max}^{i} = g_{min} + \frac{(i)(g_{max} - g_{min})}{M} \qquad 4.4$$

If a large number of partitions are made, then large number of solutions can be achieved. This results in high probability of one of the solutions having less number of adders. But this also results in longer runtime for the algorithm. The number of partitions M is decided in a similar way as in [7]. If Q is the word-length of the filter the number of partitions is given by

$$M = 40 + 5(Q - 12) \qquad 4.5$$

This way, the number of partitions increase linearly with the increase in word-length, thus keeping the run-time of the algorithm tractable.

The flowchart of the algorithm for partitioning the gain is shown in the Fig. 4.6.

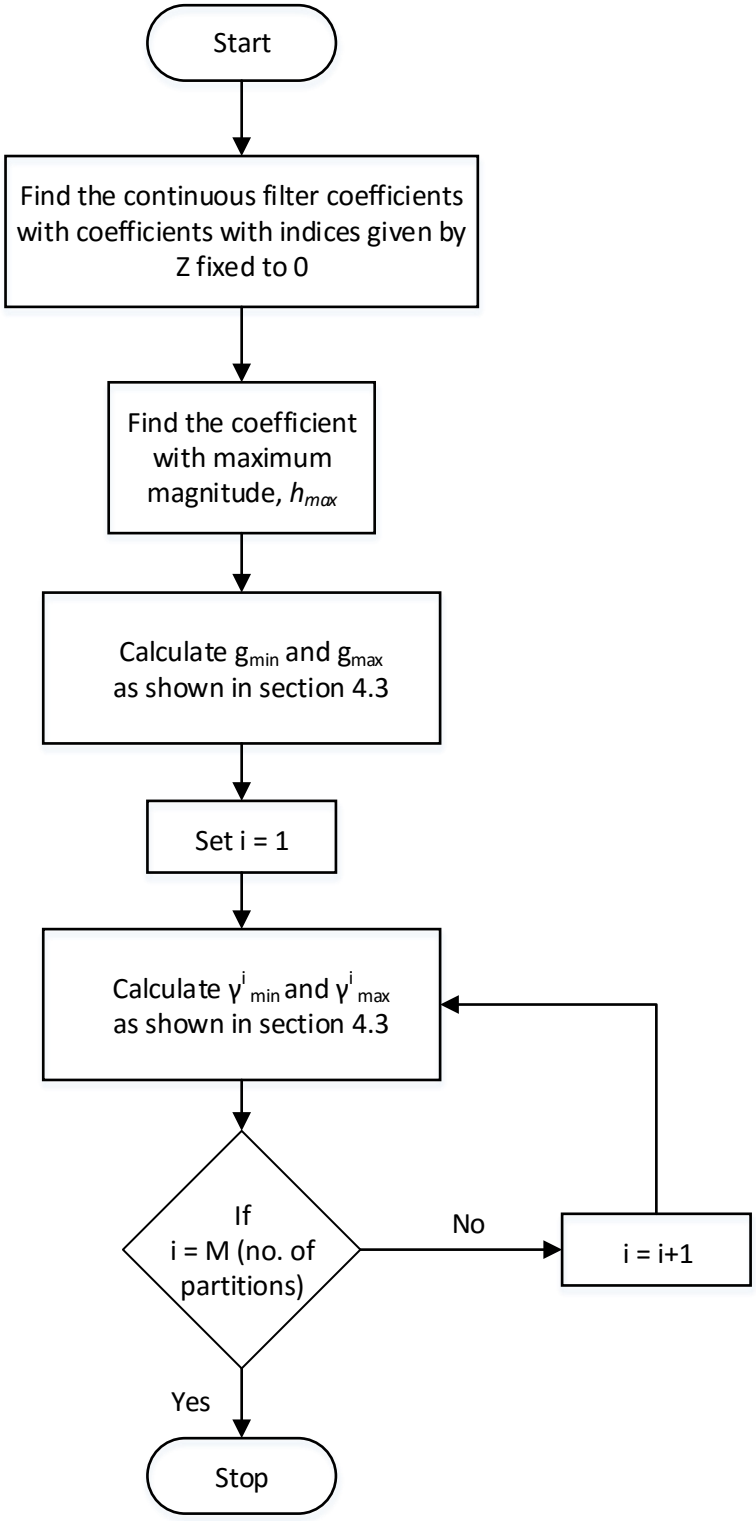


Fig. 4.6. Flowchart of the algorithm for partitioning the gain

4.4 Selecting the search neighborhood

It is very important to select the search neighborhood properly. One exact method [8] is to find the range of each coefficient by solving the following LP

$$\begin{aligned} &\text{minimize: } f = h(k) \\ &\text{such that: } b - \delta \le H(\omega) \le b + \delta, \text{for } \omega \in [0, \omega_p] \\ &-(\delta_s \delta)/\delta_p \le H(\omega) \le (\delta_s \delta)/\delta_p \text{ , for } \omega \in [\omega_{s,} \pi] \\ &b_l \le b \le b_u \end{aligned} \qquad 4.6$$

The above LP finds the lower limit of the k^{th} coefficient. Changing $f = h(k)$ to $f = -h(k)$, the upper limit of the coefficient can be found.

As the computation of LP is intensive therefore a fast way to find the neighborhood size is to take into consideration the magnitude of the coefficients. The smaller coefficients are assigned a narrower neighborhood and the larger, a wider neighborhood.

$$\begin{aligned} h_m^u &= \text{round}(h_m + K|h_m|) \\ h_m^l &= \text{round}(\, h_m - K|h_m|) \end{aligned} \qquad 4.7$$

Where h_m is the un-quantized m^{th} coefficient and h_m^u and h_m^l is the upper and lower limit of the coefficient respectively. *K* is a constant which decides the size of the neighborhood.

As in the first step of our algorithm the coefficients are fixed to the middle of their range one by one, therefore filter coefficients are already pre-optimized. It has been noticed that the initial few coefficients need not be optimized further by EO. Therefore in the actual algorithm, instead of using a constant multiplier *K* as in equation a more complicated function is used.

In the algorithm in this thesis the neighborhood is defined as shown in the following equation

$$h_m^u = \text{round}(h_m + \text{f}(m))$$
$$h_m^l = \text{round}(\ h_m - \text{f}(m))$$ 4.8

Where $\text{f}(m)$ is a function of the index of the coefficient.

4.5 Tree search algorithm to fix the coefficients

After fixing some coefficients to 0, the remaining un-quantized coefficients are quantized using a tree search algorithm, in which each coefficient is fixed to the value nearest to the middle of its feasible range. This is done in a similar way as in [7].

1. For a given passband range $\Gamma_m = [\text{l}_m, \text{u}_m]$, set $i = 0$.

2. If $i \in Z$, go to step 4, else find the feasible range of i^{th} coefficient by solving the following linear programming problem

Minimize $f_0 = h(i)$ and $f_1 = -h(i)$

Subject to.: $b_m(1 - \delta)2^Q \leq H(\omega) \leq b_m(1 + \delta)2^Q$, for $\omega \in [1, \omega_p]$

$-\delta_p b_m 2^Q \leq H(\omega) \leq \delta_p b_m 2^Q$, for $\omega \in [\omega_s, \pi]$ 4.9

$\text{l}_m \leq \text{b}_m \leq \text{u}_m$

Where

$$H(\omega) = \sum_{n=0,\notin Z}^{i-1} h'(n)\text{Trig}(\omega, n) + \sum_{n=i,\notin Z}^{\left\lfloor \frac{N-1}{2} \right\rfloor} h(n)\text{Trig}(\omega, n)$$ 4.10

Where $h'(n)$ are quantized coefficients and $h(n)$ are variable coefficients. The feasible range of the coefficient is therefore, $f_0 - f_1$.

3. If feasible range is empty, then feasible solution is not found for this passband range and go to step 5, otherwise set $h'(n)$ to the discrete value closest to the middle of the range.

4. Set $i = i + 1$. If $i = D$, a feasible solution is found for this gain.

5. Stop.

The flowchart of the algorithm is shown on the next page.

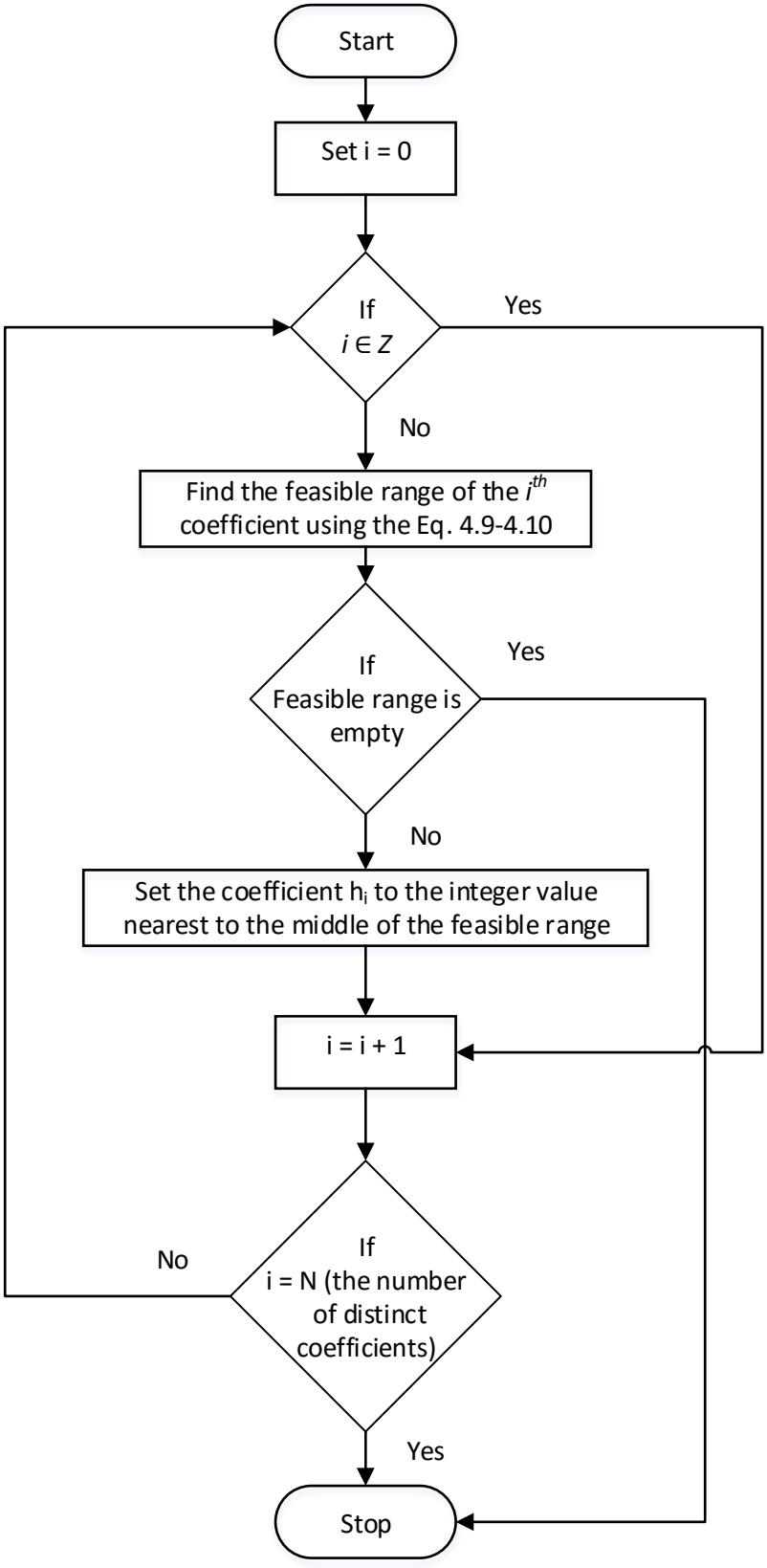


Fig. 4.7. Tree search algorithm of the first step

4.6 Speeding-up the frequency response calculations

Repeated cost function evaluations is the most computationally intensive step of the algorithm.

The frequency response of a linear phase FIR filter is given by the equations 1.5-1.8, for the four types of filters. The four equations can be written in a generalized for as below:

$$H(\omega) = \sum_{n=0}^{\lfloor (N-1)/2 \rfloor} h_n \mathrm{Trig}(\omega, n) \qquad 4.11$$

Where, $\mathrm{Trig}(\omega, n)$ is a sinusoidal function of ω and n depending on the type of the filter.

In the extremal optimization method proposed in this thesis, at each iteration, each coefficient is changed to all the values in its chosen range, keeping the others fixed to the previous iteration's solution. For all the realizations obtained, the objective function is computed. To compute the objective function, the frequency response has to be computed first. From the equation 4.11, we can see that re-computing the whole equation is unnecessary. The trigonometric functions can be pre-evaluated at all the frequency points on the grid and saved in a matrix as shown below:

$$\mathrm{Trig}_{MAT} = \begin{bmatrix} \mathrm{Trig}(\omega_1, 0) & \cdots & \mathrm{Trig}(\omega_K, 0) \\ \vdots & \ddots & \vdots \\ \mathrm{Trig}(\omega_1, \lfloor (N-1)/2 \rfloor) & \cdots & \mathrm{Trig}(\omega_K, \lfloor (N-1)/2 \rfloor) \end{bmatrix} \qquad 4.12$$

The frequency response vector can be then written in the following matrix form:

$$H = [h_0 \quad \dots \quad h_{\lfloor (N-1)/2 \rfloor}] \begin{bmatrix} \mathrm{Trig}(\omega_1, 0) & \cdots & \mathrm{Trig}(\omega_K, 0) \\ \vdots & \ddots & \vdots \\ \mathrm{Trig}(\omega_1, \lfloor (N-1)/2 \rfloor) & \cdots & \mathrm{Trig}(\omega_K, \lfloor (N-1)/2 \rfloor) \end{bmatrix}$$

$$H = \boldsymbol{h}^T . \text{Trig}_{MAT} \tag{4.13}$$

If the n[th] coefficient is increased by amount c, the new frequency response becomes

$$H_{new} = \begin{bmatrix} h_0 & \dots & h_n + c & \dots & h_{\left\lfloor \frac{(N-1)}{2} \right\rfloor} \end{bmatrix} .$$

$$\begin{bmatrix} \text{Trig}(\omega_1, 0) & \cdots & \text{Trig}(\omega_K, 0) \\ \vdots & \ddots & \vdots \\ \text{Trig}(\omega_1, \lfloor (N-1)/2 \rfloor) & \cdots & \text{Trig}(\omega_K, \lfloor (N-1)/2 \rfloor) \end{bmatrix}$$

$$= \begin{bmatrix} h_0 & \dots & h_n & \dots & h_{\left\lfloor \frac{(N-1)}{2} \right\rfloor} \end{bmatrix} . \tag{4.14}$$

$$\begin{bmatrix} \text{Trig}(\omega_1, 0) & \cdots & \text{Trig}(\omega_K, 0) \\ \vdots & \ddots & \vdots \\ \text{Trig}(\omega_1, \lfloor (N-1)/2 \rfloor) & \cdots & \text{Trig}(\omega_K, \lfloor (N-1)/2 \rfloor) \end{bmatrix}$$
$$+ c[\text{Trig}(\omega_1, n) \quad \dots \quad \text{Trig}(\omega_K, n)]$$

$$= H + c[\text{Trig}(\omega_1, n) \quad \dots \quad \text{Trig}(\omega_K, n)]$$

From the final result, we can notice that in a given iteration we need to compute only the second term, which requires significantly small computational time. H is fixed, and can be simply added to the second term to find the frequency response of the mutated filter.

Chapter 5

Results

5.1 Convergence analysis

The algorithm is run 60 times at different values of τ. The value of τ is varied from 1 to 3 in the steps of 0.1. It is noticed that at low values of τ the algorithm does not find good minima and the takes longer time to find filter with lower objective function. The plot of iterations vs the best objective function value found so far is plotted at $\tau = 1.0$, 1.8, 2.4 and 3.0 for different runs of the algorithm.

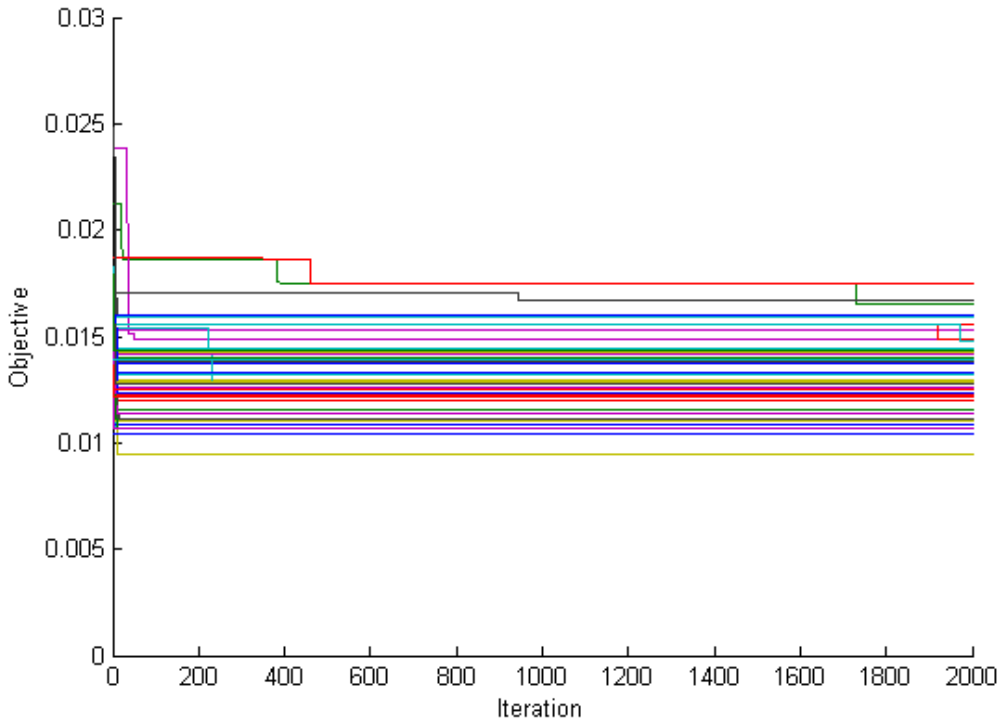


Fig. 5.1. Convergence at $\tau = 1.0$

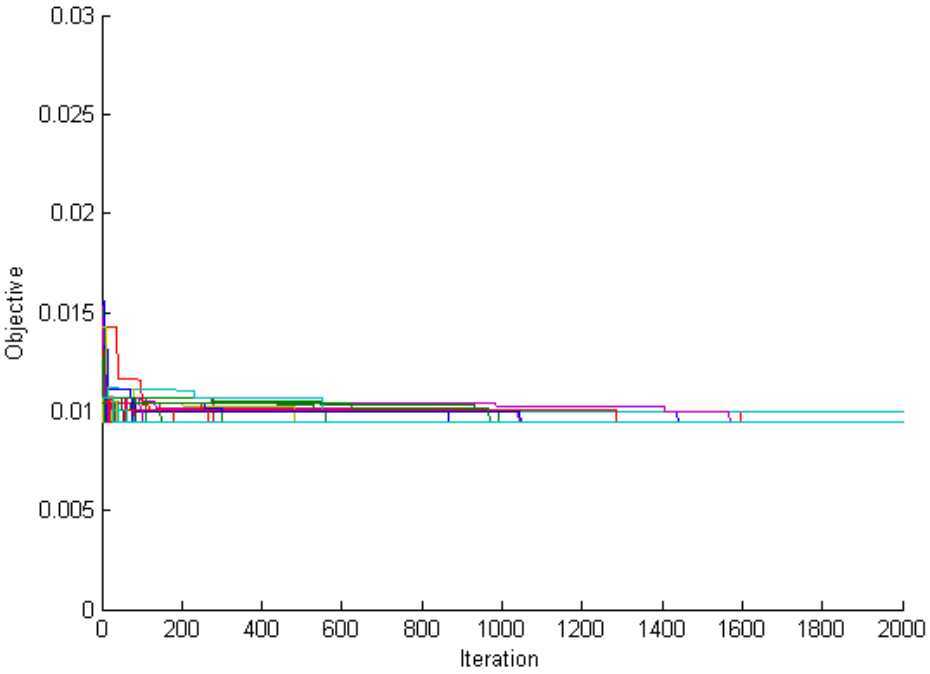


Fig. 5.2. Convergence at $\tau = 1.8$

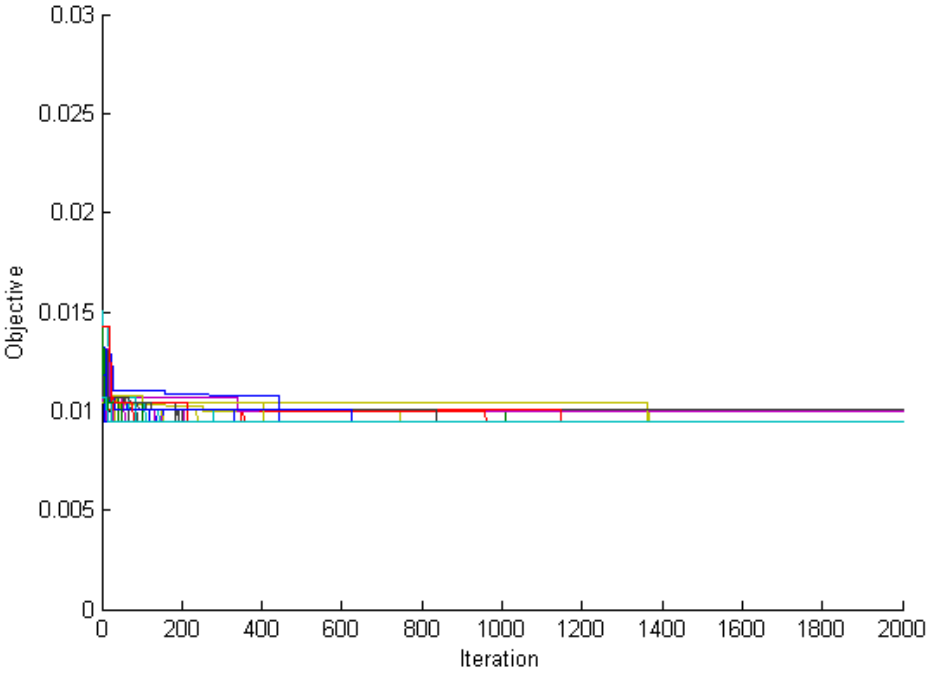


Fig. 5.3. Convergence at $\tau = 2.4$

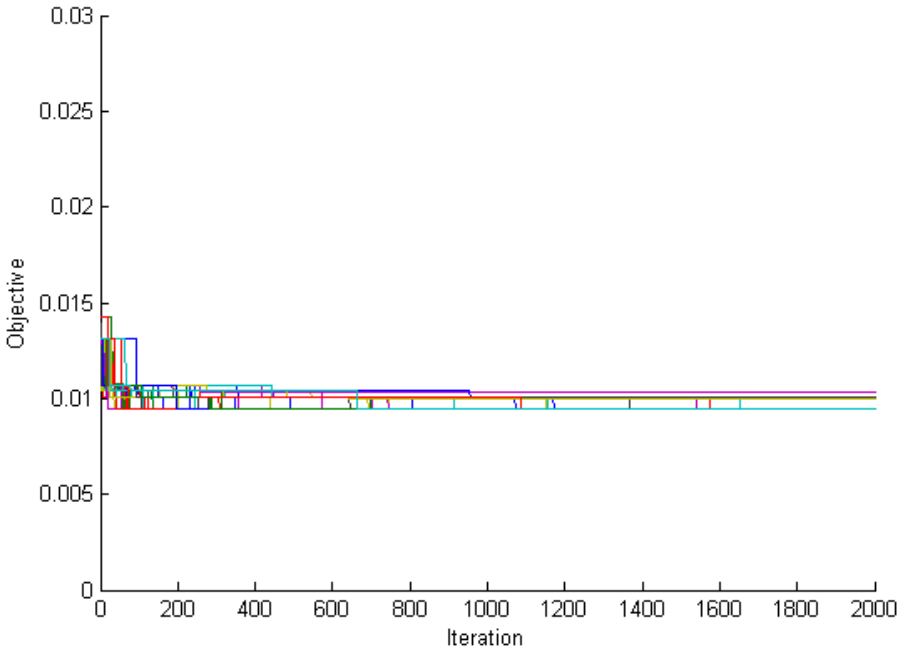


Fig. 5.4. Convergence at $\tau = 3.0$

It can be seen from the Fig. 5.1 that at very low value of τ, the convergence of the algorithm is not very fast. Each individual line in the plot represents a single run of the algorithm. It can be seen that different runs of the algorithm do not converge to the same final value after fixed number of iterations (which are 2000 in this case). At $\tau = 1.0$, the deviation of the final minima obtained in each run is very high, which can be seen from the spread of various lines at the extreme right of the figure.

At $\tau = 1.8$, most of the runs of the algorithm converge to the same final value. After $\tau = 2.0$, the spread of the final values again starts increasing slowly. Therefore, from the above discussion, it can be concluded that the optimum value of τ, lies in between these two extreme values.

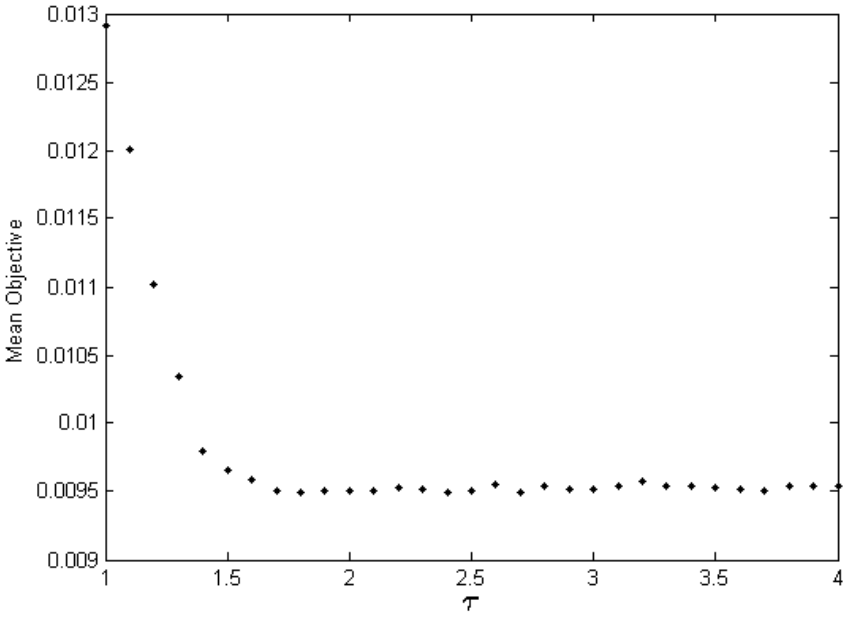


Fig. 5.5. Variation of mean value of objective with τ

Fig. 5.5 shows the variation of mean value of objective with τ. The mean value is obtained from the plots similar to Fig. 5.1-5.4 for different values of τ. The final values (after 2000 iterations) for each run at a given τ, are found and their mean is calculated. The lower value denotes that the final result obtained is on average closer to the global optima.

From the figure, it can be seen that the minimum occurs at around 1.7-1.8. After this, the mean value starts increasing again. Still, even at $\tau = 4.0$, the increase is comparatively small.

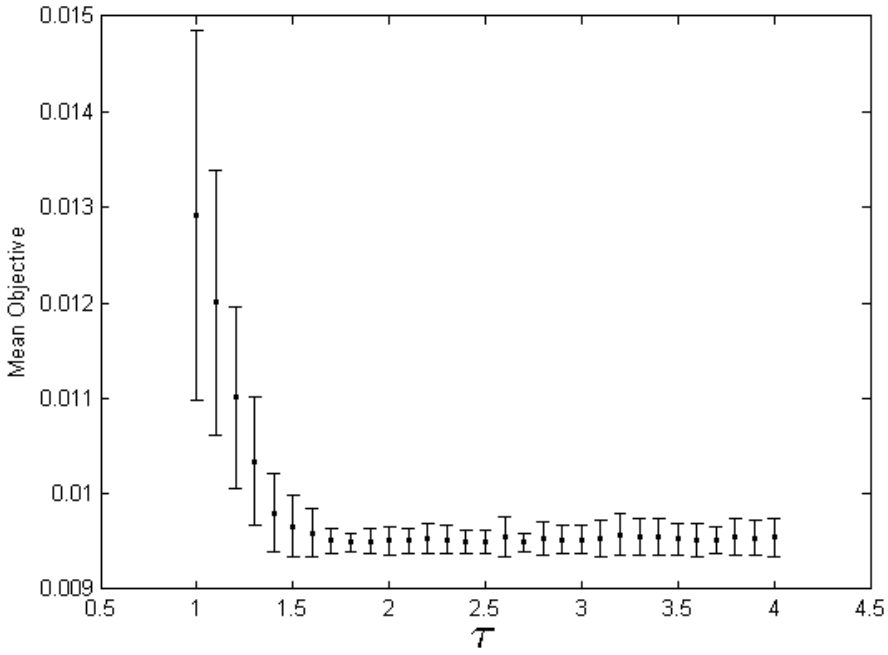


Fig. 5.6. The variance of the final objective with τ.

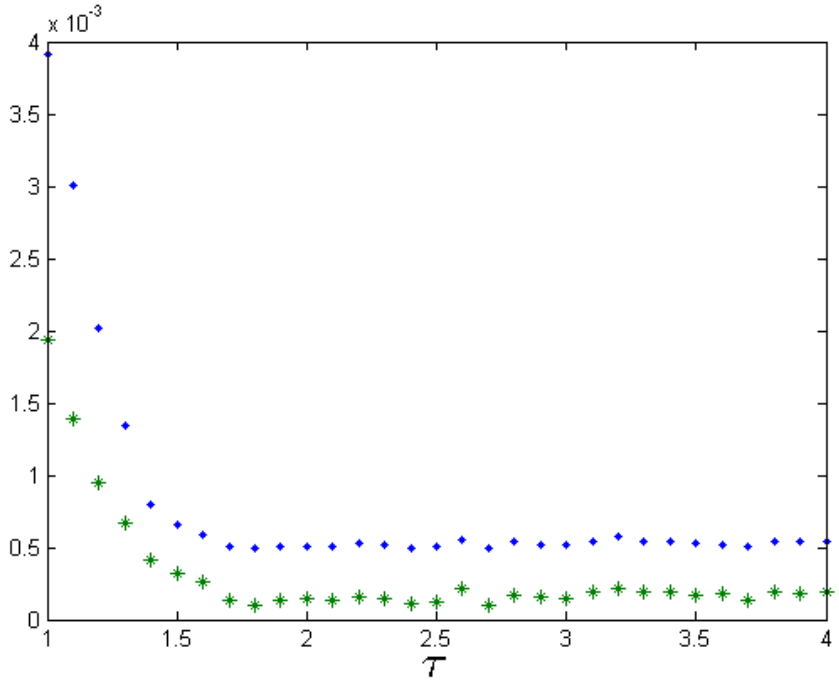


Fig. 5.7. Variance (dots) and mean value (stars, shifted vertically)

In Fig. 5.6 the mean value of the objective function is shown at various values of τ. The lines represent the variance of the objective function around the mean value. In Fig. 5.7, the mean value and the variance are compared side by side for easier comparison. It can be seen that there is a high degree of correlation between the 2 plots. When the mean value is high the variance is also high and vice versa.

In the next figure, the expected number of iterations for the objective function to fall below a specific value are calculated and plotted at different settings of τ.

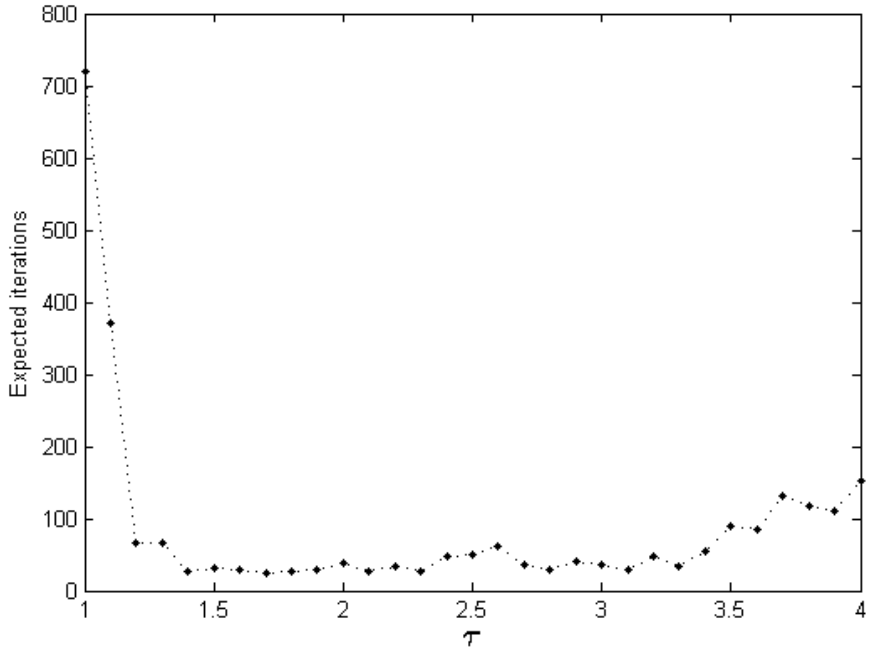


Fig. 5.8. Expected number of iterations at different τ.

The analysis in this section was done on the filter A, but it has been observed that taking the value of τ from 1.7 to 2.0 gives good results for filters with wide range of specifications. In the design examples shown in the section 5.2 the value of τ used is 1.7 unless stated otherwise.

5.2 Design examples

Eight design examples are shown in this section. The filters shown in Table 5.1 are standard benchmark filters taken from the literature [6], [7] and [12]. The filters A and S2 are medium order filters and the filters B and C are high order filters. ω_p and ω_s are the passband and stopband cut-off frequencies respectively and δ_p and δ_s are the maximum allowed passband and stopband errors. All the filters are low pass filters and without the loss of generality the design method can be extended to other types of filters as well.

The filters shown in Table 5.2 are designed to show the robustness of the design method. The first filter is a band-pass filter and the second one is a band-stop filter. Both of these filters are high order filters with filter orders 112 and 104 respectively.

Ω_p and Ω_s are the normalized passband and stopband frequencies respectively. From the given passband and stopband error specifications the filter order is decided by first designing feasible floating point filters using Parks-McClellan algorithm. Next the filter order and word-length is chosen, such that the resulting fixed point filter with minimum number of adders is achieved.

Filter	Type	Filter order	ω_p	ω_s	δ_p	δ_s
A	Low pass	58	0.125	0.225	0.01	0.001
S2	Low pass	59	0.042	0.14	0.01158	0.001
B	Low pass	104	0.2	0.24	0.01	0.01
C	Low pass	324	0.125	0.14	0.005	0.005
Y1	Low pass	29	0.3	0.5	0.00316	0.00316
G1	Low pass	15	0.2	0.5	0.01	0.01

Table 5.1. Specifications of the benchmark filters

Type	Filter order	Ω_p	Ω_s	δ_p	δ_s
Band pass	112	[0.24, 0.5]	[0, 0.2]U[0.55, 1]	0.01	0.005
Band stop	104	[0, 0.06]U[0.25, 1]	[0.1, 0.2]	0.01	0.01

Table 5.2. Specifications of the Band pass and Band stop filters

The design of each of these filters is shown below:

5.2.1 Example 1: Design of filter A

The filter A is an odd length filter. Initially some coefficients are fixed to 0. If $h(n)$ are the filter coefficients, such that $0 \leq n \leq 58$, then due to the symmetry, we need to design only the first 30 coefficients. The indices of the coefficients with 0 value are found to be $Z \in \{6,12\}$. In other words, the coefficients $h(5)$ and $h(11)$ are 0.

The gain is partitioned into 45 parts. After fixing the zero coefficients, at each partition of the gain a corresponding discrete filter is design as explained in the section (). The feasible solutions are selected and RAG-n algorithm is used to find the number of multiplier adders required for each of the solution. The feasible solution with minimum number of adders obtained in this step is found to be

2, 4, 6, 6, 4, 0, -7, -15, -21, -21, -15, 0, 21, 42, 57, 59, 42, 6, -43, -94, -131, -136, -97, -7, 128, 294, 466, 617, 719, 756

The minimax error is 0.00993 and the number of multiplier block adders obtained is 16.

In Fig. 5.9 the error of the filters quantized at different gains is shown. The Fig. 5.10 shows the number of adders achieved for different gains.

The general trend, as would be expected, is the decrease in the error as the gain increases. This is because when the gain is higher the coefficients are scaled to integers with larger values and the rounding error when rounding the coefficients is lesser. Therefore the

coefficients when normalized to unity passband gain, are closer to the original un-quantized coefficients.

On the other hand, the number of adders generally increases with the increase of gain. This is because with the increase in gain, the range of integer values which the coefficients can take increases, resulting in the decrease in the probability of achieving less distinct odd fundamentals. This is the same reason, why the filter with higher word-length has more probability of achieving higher number of adders. Due to this fact, it is preferable to design a filter with smaller word-length. This has an added advantage that the number of full adders is also less in the resulting hardware implementation. In addition it is easier for the algorithm to design a filter with smaller word-length and high order in comparison to a filter with low order and high word-length.

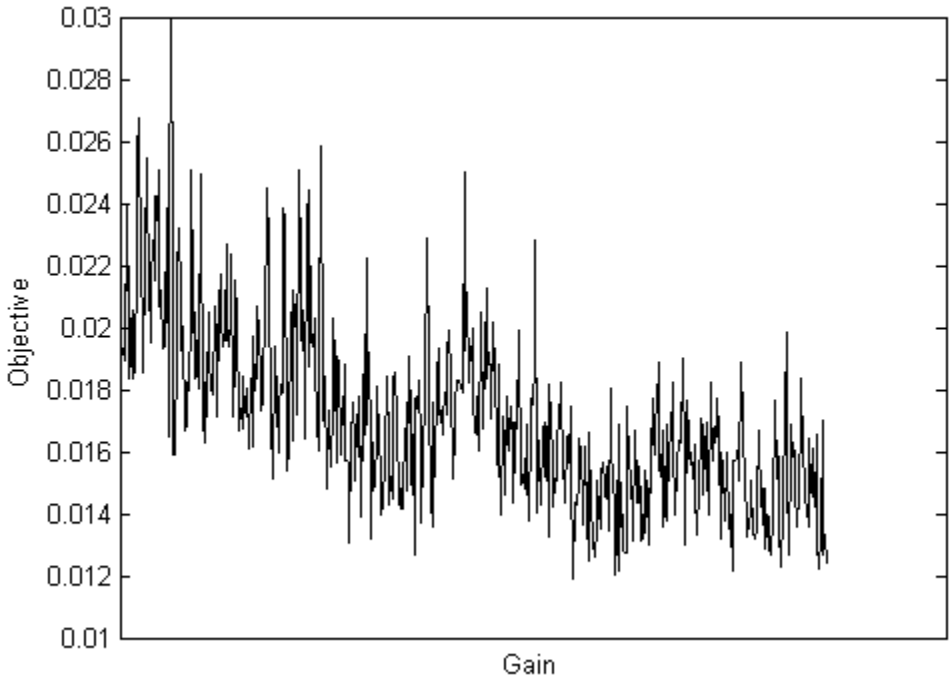


Fig. 5.9. Variation of error of the filter quantized at different gains

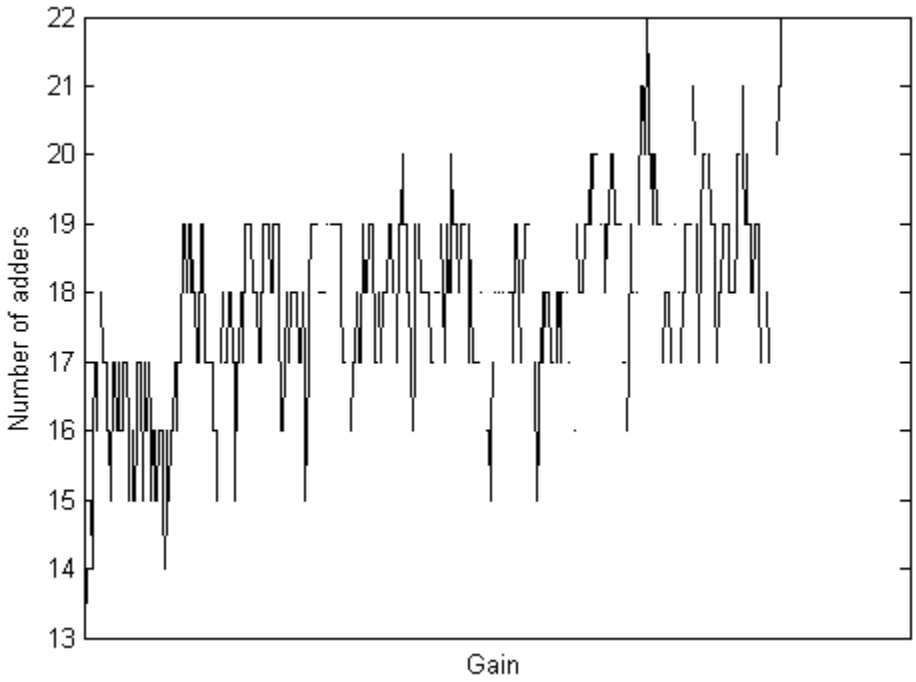


Fig. 5.10. Variation of number of multiplier block adders at different gains

For the second step involving EO algorithm, the solutions within 105% of δ_p are selected. The best solution is chosen as the final solution of the algorithm. The Table 5.3 shows the filter coefficients, the odd fundamentals (Basis set) and the synthesis of the filter coefficients. The number of adders achieved is 15.

The magnitude of the frequency response of the filter A is shown in the Figure 5.11 and 5.12.

Filter A, $h(n) = h(58 - n)$ for $0 \leq n \leq 29$ Passband gain: 5268.16004, $EWL = 10$ 3, 5, 7, 8, 6, 0, -8, -17, -24, -25, -17, 0, 24, 50, 68, 70, 50, 8, -50, -110, -154, -160, -115, -10, 149, 344, 546, 723, 844, 887		
Basis Set ={3, 5, 7, 17, 25, 35, 43, 55, 77, 115, 149, 211, 273, 723, 887}		
$3 = 1\times2^1+1$	$5 = 1\times2^2+1$	$7 = 1\times2^3-1$
$17 = 1\times2^4+1$	$25 = 1\times2^3+17$	$35 = 1\times2^5+3$
$43 = 5\times2^3+3$	$55 = 7\times2^3-1$	$77 = 5\times2^4-3$
$115 = 7\times2^4+3$	$149 = 77\times2^1-5$	$211 = 3\times2^5+115$
$273 = 1\times2^8+17$	$723 = 43\times2^4+35$	$887 = 55\times2^4+7$

Table 5.3. Filter A coefficients, basis set and its adder synthesis

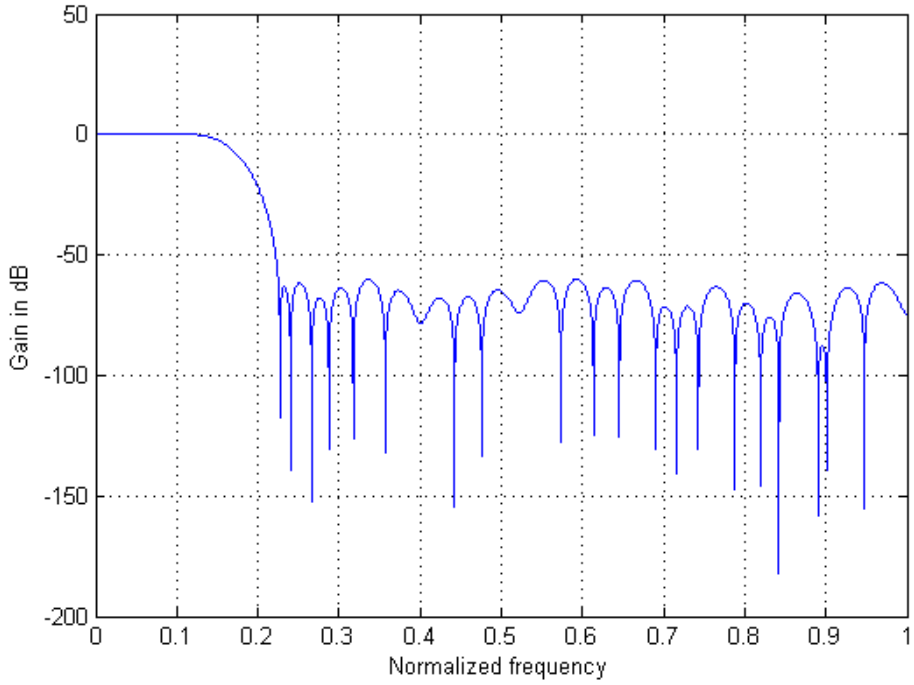


Fig. 5.11. Magnitude response of filter A

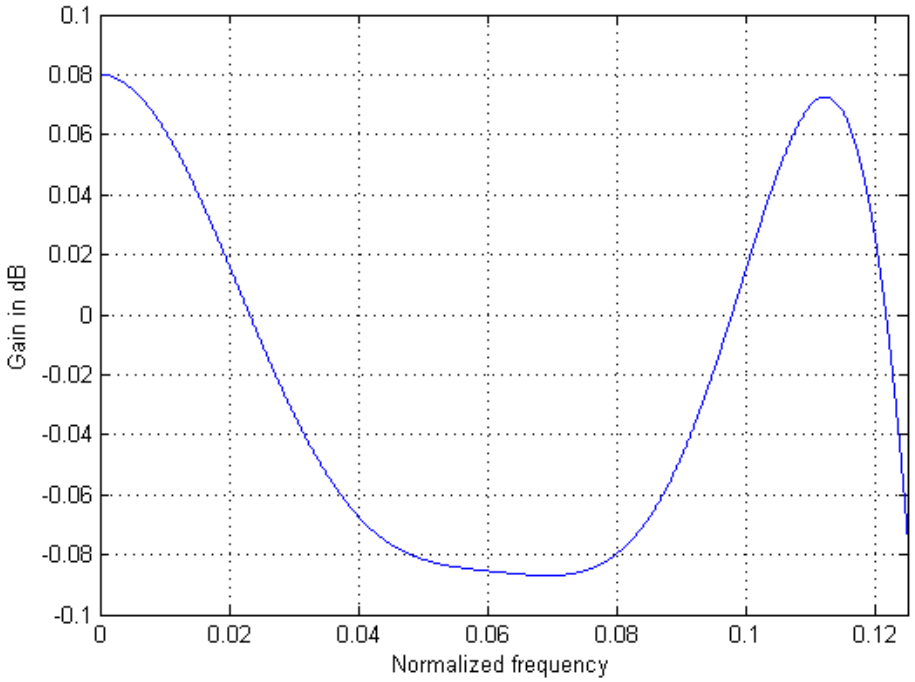


Fig. 5.12 Passband of the magnitude response of filter A

5.2.2 Example 2: Design of filter S2

The filter S2 is an even length filter. No coefficients are fixed to zero in this case.

The gain is partitioned into 50 parts. The feasible solution with minimum number of adders obtained in this step is found to be

5, 5, 6, 5, 2, -3, -10, -21, -34, -50, -66, -82, -95, -102, -102, -90, -66, -27, 27, 98, 182, 279, 383, 492, 600, 700, 789, 860, 910, 936

The minimax error is 0.01135 and the number of multiplier block adders obtained is 22.

For the second step involving EO algorithm, the solutions within 105% of δ_p are selected. The best solution is chosen as the final solution of the algorithm. The following table shows the filter coefficients, the odd fundamentals (Basis set) and the synthesis of the filter coefficients. The number of adders achieved is 18.

Filter *S2*, $h(n) = h(59 - n)$ for $0 \leq n \leq 29$ Passband gain: 1066.68719, $EWL = 8$ 5, 5, 5, 5, 2, -2, -10, -20, -32, -47, -63, -78, -90, -97, -97, -86, -63, -26, 26, 93, 173, 266, 365, 470, 571, 668, 752, 821, 868, 893		
Basis Set ={5, 13, 39, 43, 45, 47, 63, 93, 97, 133, 167, 173, 217, 235, 365, 571, 821, 893}		
$5 = 1\times2^1+1$	$13 = 1\times2^3+5$	$39 = 5\times2^3-1$
$43 = -5\times2^2+63$	$45 = 5\times2^3+5$	$47 = -1\times2^4+63$
$63 = 1\times2^6-1$	$93 = -1\times2^2+97$	$97 = 5\times2^5-63$
$133 = 1\times2^7+5$	$167 = 1\times2^7+39$	$173 = 1\times2^7+45$
$217 = 7\times2^5-7$	$235 = 43\times2^2+63$	$365 = 5\times2^7+45$
$571 = 133\times2^2+39$	$821 = 43\times2^4+133$	$893 = 45\times2^4+173$

Table 5.4. Filter S2 coefficients, basis set and its adder synthesis

The magnitude of the frequency response of the filter S2 is in the Figure 5.13 and 5.14.

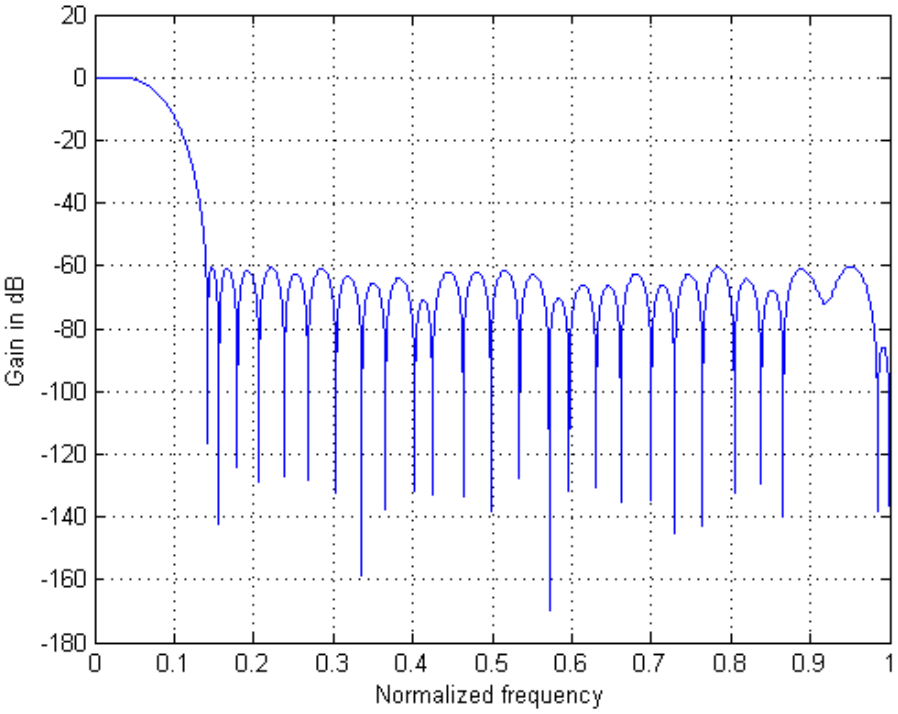


Fig. 5.13. Magnitude response of filter S2

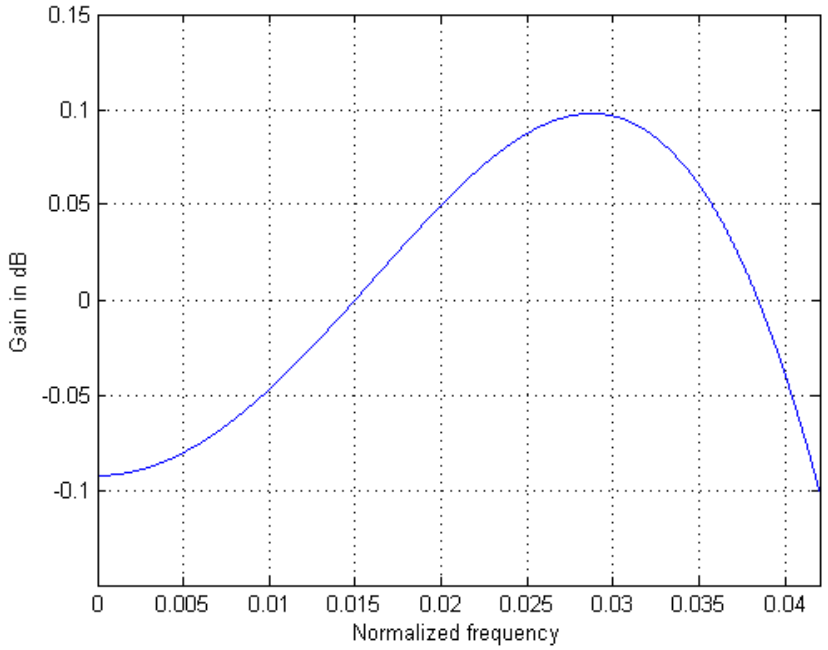


Fig. 5.14. Passband of the magnitude response of filter S2

5.2.3 Example 3: Design of filter B

The filter B is an odd length filter. The indices of the coefficients with 0 value are found to be $Z \in \{2, 3, 8\ 12, 21\}$.

The gain is partitioned into 40 parts. No feasible solution is found in the first step for this filter.

For the second step involving EO algorithm, the solutions within 105% of δ_p are selected. The best solution is chosen as the final solution of the algorithm. The following table shows the filter coefficients, the odd fundamentals (Basis set) and the synthesis of the filter coefficients. The number of adders achieved is 12.

<table>
<tr><td colspan="3">Filter B, $h(n) = h(104 - n)$ for $0 \leq n \leq 52$

Passband gain: 1042.71367, $EWL = 8$

-2, 0, 0, 2, 3, 2, 1, 0, -2, -3, -2, 0, 2, 4, 4, 2, -1, -4, -6, -4, 0, 3, 7, 7, 4, -2, -7, -10, -8, -2, 6, 12, 13, 8, -2, -12, -18, -16, -6, 10, 24, 27, 19, -2, -28, -46, -46, -20, 30, 96, 164, 211, 228</td></tr>
<tr><td colspan="3">Basis Set ={ 3, 5, 7, 9, 13, 15, 19, 23, 27, 41, 57, 211}</td></tr>
<tr><td>$3 = 1\times2^1+1$</td><td>$5 = 1\times2^2+1$</td><td>$7 = 1\times2^3-1$</td></tr>
<tr><td>$9 = 1\times2^3+1$</td><td>$13 = 3\times2^2+1$</td><td>$15 = 2^4-1$</td></tr>
<tr><td>$19 = 9\times2^1+1$</td><td>$23 = 3\times2^3-1$</td><td>$27 = 7\times2^2-1$</td></tr>
<tr><td>$41 = 5\times2^3+1$</td><td>$57 = 7\times2^3+1$</td><td>$211 = 7\times2^5-13$</td></tr>
</table>

Table 5.5. Filter B coefficients, basis set and its adder synthesis

The magnitude of the frequency response of the filter B is shown in the Figure 5.15 and 5.16.

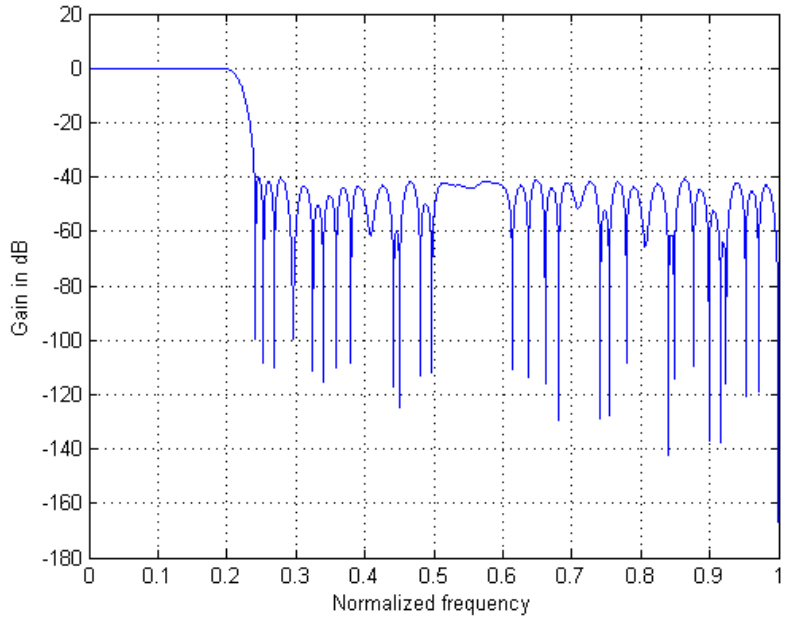


Fig. 5.15. Magnitude response of filter B

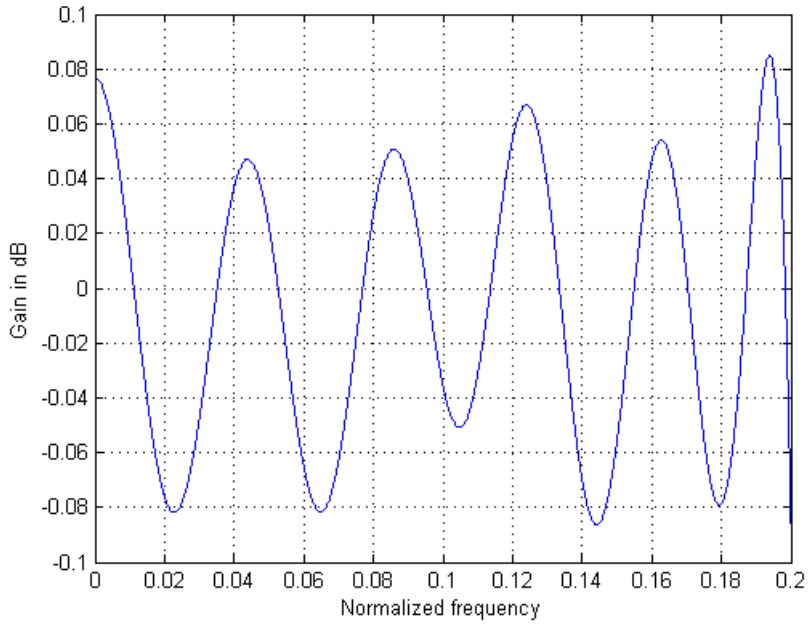


Fig. 5.16. Passband of the magnitude response of filter B

5.2.4 Example 4: Design of filter C

The filter C is an odd length filter. The indices of the coefficients with 0 value are found to be $Z \in \{3, 4, 8, 11, 12, 13, 14, 18, 19, 20, 27, 28, 34, 35, 42, 43, 50, 57, 65, 80, 95\}$.

The gain is partitioned into 45 parts.

For the second step involving EO algorithm, the solutions within 105% of δ_p are selected. The best solution is chosen as the final solution of the algorithm. The following table shows the filter coefficients, the odd fundamentals (Basis set) and the synthesis of the filter coefficients. The number of adders achieved is 28. The magnitude of the frequency response of the filter C is shown in Figure 5.17 and 5.18.

In the second step of the algorithm, the coefficients are optimized in groups of 40 each. Each group is optimized one by one. Once all groups has been optimized the process is repeated again until there is no further decrease in the number of adders in the resulting filter. This is done in order to avoid the exponential increase in the runtime of extremal optimization due to large number of coefficients to be optimized.

The first step solution which results in the best solution after the EO algorithm is shown below.

-4, -3, 0, 0, 1, 2, 3, 0, 3, 3, 0, 0, 0, 0, -3, -3, -3, 0, 0, 0, 2, 3, 4, 3, 2, 3, 0, 0, -3, -4, -4, -3, -3, 0, 0, 3, 4, 5, 5, 4, 3, 0, 0, -4, -5, -6, -5, -4, -2, 0, 3, 5, 7, 7, 6, 4, 0, -2, -5, -7, -8, -8, -6, -3, 0, 4, 7, 9, 10, 8, 6, 2, -3, -7, -10, -12, -11, -9, -5, 0, 5, 10, 13, 14, 13, 9, 3, -3, -9, -14, -16, -16, -13, -8, 0, 7, 14, 18, 20, 18, 13, 6, -3, -12, -19, -23, -23, -19, -12, -2, 9, 18, 26, 28, 26, 20, 9, -4, -16, -28, -34, -35, -30, -19, -4, 12, 28, 39, 44, 42, 32, 16, -4, -25, -43, -55, -59, -51, -35, -10, 18, 47, 70, 83, 82, 67, 37, -4, -50, -94, -128, -144, -136, -102, -37, 52, 160, 281, 399, 509, 594, 650, 669

The passband error of this solution is 0.0049879 and the number of adders is 29.

Filter *C*, $h(n) = h(324 - n)$ for $0 \leq n \leq 162$

Passband gain: 5057.40188, $EWL = 10$

-4, -3, 0, 0, 1, 2, 3, 0, 3, 3, 0, 0, 0, 0, -3, -3, -3, 0, 0, 0, 2, 3, 4, 3, 2, 3, 0, 0, -3, -4, -4, -3, -3, 0, 0, 3, 4, 5, 5, 4, 3, 0, 0, -4, -5, -6, -5, -4, -2, 0, 3, 5, 7, 7, 6, 4, 0, -2, -5, -7, -8, -8, -6, -3, 0, 4, 7, 9, 10, 8, 6, 2, -3, -7, -10, -12, -11, -9, -5, 0, 5, 10, 13, 14, 13, 9, 3, -3, -9, -14, -17, -16, -13, -8, 0, 7, 14, 18, 19, 18, 13, 6, -4, -12, -19, -23, -23, -19, -12, -2, 8, 19, 26, 28, 26, 19, 9, -4, -17, -27, -34, -35, -31, -19, -5, 12, 27, 39, 44, 42, 32, 17, -4, -25, -44, -56, -58, -51, -34, -10, 19, 47, 71, 83, 82, 67, 36, -4, -50, -94, -128, -144, -137, -101, -38, 52, 160, 280, 400, 509, 595, 651, 670

Basis Set ={3, 5, 7, 9, 11, 13, 17, 19, 21, 23, 25, 27, 29, 31, 35, 39, 41, 47, 51, 67, 71, 83, 101, 137, 335, 509, 595, 651}

$3 = 1\times2^1+1$	$5 = 1\times2^2+1$	$7 = 1\times2^3-1$
$9 = 1\times2^3+1$	$11 = 1\times2^3+3$	$13 = 1\times2^4-3$
$17 = 1\times2^4+1$	$19 = 1\times2^4+3$	$21 = 1\times2^4+5$
$23 = 1\times2^4+7$	$25 = 1\times2^4+9$	$27 = -1\times2^2+31$
$29 = -1\times2^1+31$	$31 = 1\times2^5-1$	$35 = 1\times2^5+3$
$39 = 1\times2^5+7$	$41 = 1\times2^5+9$	$47 = 1\times2^6-17$
$51 = 5\times2^2+31$	$67 = 1\times2^6+3$	$71 = 1\times2^6+7$
$83 = 5\times2^4+3$	$101 = 7\times2^4-1$	$137 = 1\times2^7+9$
$335 = 83\times2^2+3$	$509 = 1\times2^9-3$	$595 = 137\times2^4+47$
$651 = 71\times2^1+509$		

Table 5.6. Filter C coefficients, basis set and its adder synthesis

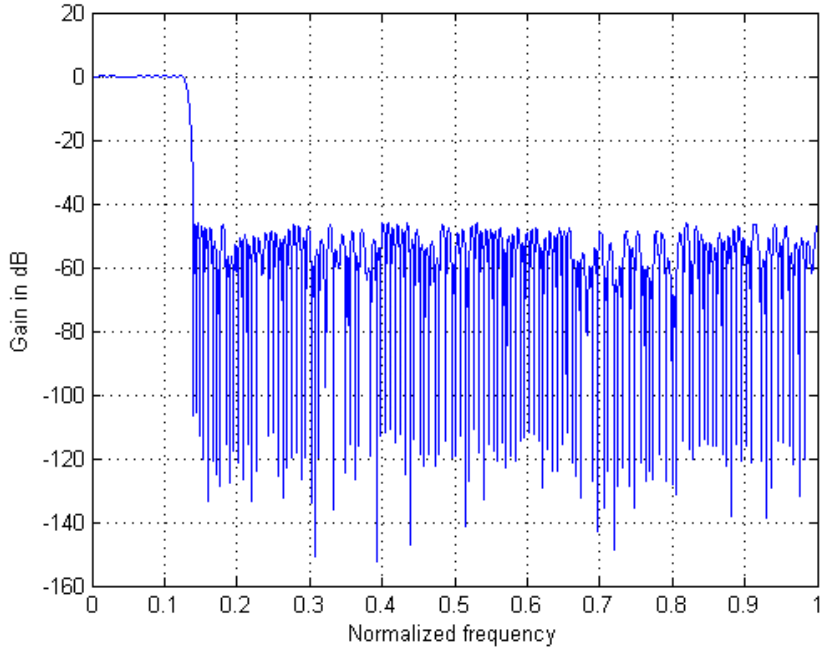


Fig. 5.17. Magnitude response of filter C

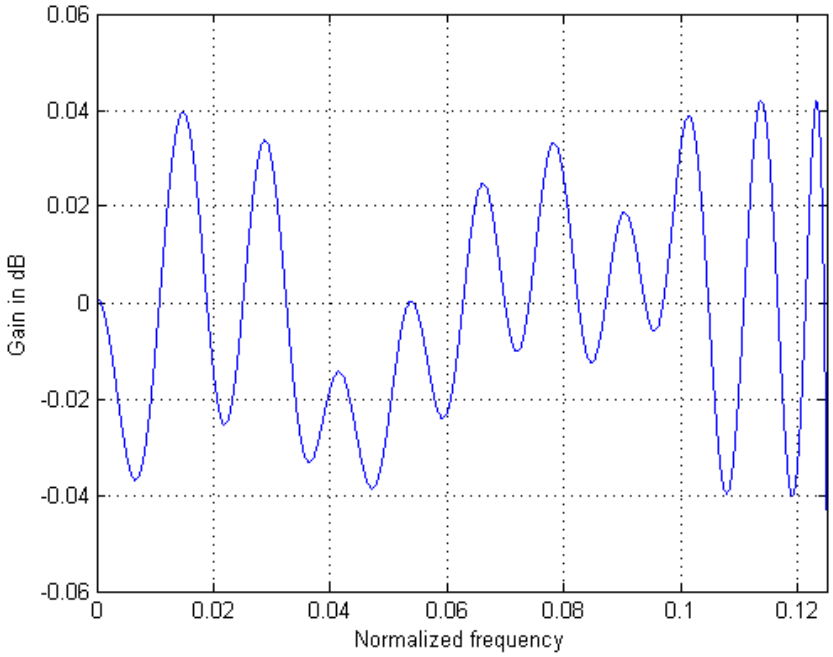


Fig. 5.18. Passband of the magnitude response of filter C

5.2.5 Example 5: Design of filter Y1

The filter Y1 is a short even length filter. The indices of the coefficients with 0 value are found to be $Z \in \{3, 8, 13\}$.

The gain is partitioned into 40 parts.

For the second step involving EO algorithm, the solutions within 105% of δ_p are selected. The best solution is chosen as the final solution of the algorithm. The following table shows the filter coefficients, the odd fundamentals (Basis set) and the synthesis of the filter coefficients. The number of adders achieved is 6 and the passband error is 0.003029. The magnitude of the frequency response of the filter Y1 is shown on the next page.

<table>
<tr><td colspan="3">$h(n) = h(29 - n)$ for $0 \leq n \leq 14$

Passband gain: 1400.65740, $EWL = 10$

-1, -4, 0, 9, 8, -11, -24, 0, 44, 36, -48, -108, 0, 277, 523</td></tr>
<tr><td colspan="3">Basis Set ={3, 9, 11, 27, 277, 523}</td></tr>
<tr><td>$3 = 1\times2^1+1$</td><td>$9 = 1\times2^3+1$</td><td>$11 = 1\times2^3+3$</td></tr>
<tr><td>$27 = 3\times2^3+3$</td><td>$277 = 9\times2^5-11$</td><td>$523 = 1\times2^9+11$</td></tr>
</table>

Table 5.7. Filter Y1 coefficients, basis set and its adder synthesis

In the following figure, the hardware implementation of the above designed filter is shown. For the remaining examples the hardware implementation is not shown. In the figure, the sign '<' denotes left shifting the binary value, which is equivalent to multiplying by factor of 2. The part within the light dotted rectangle is the multiplier block of the filter.

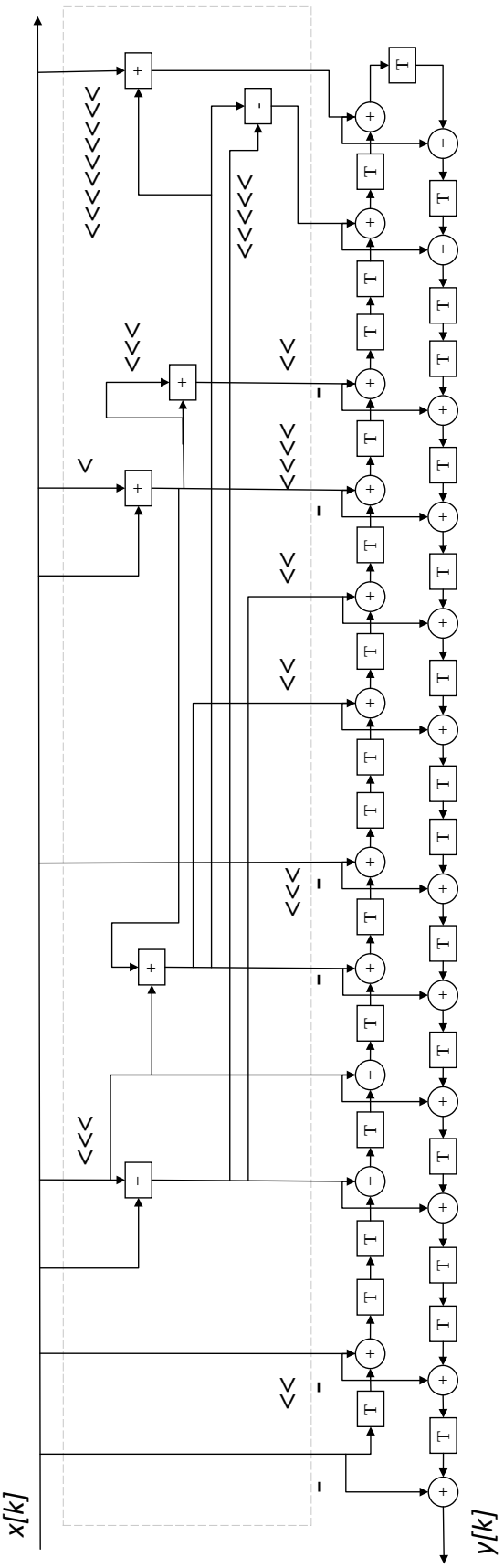


Fig. 5.19. Hardware implementation of filter Y1

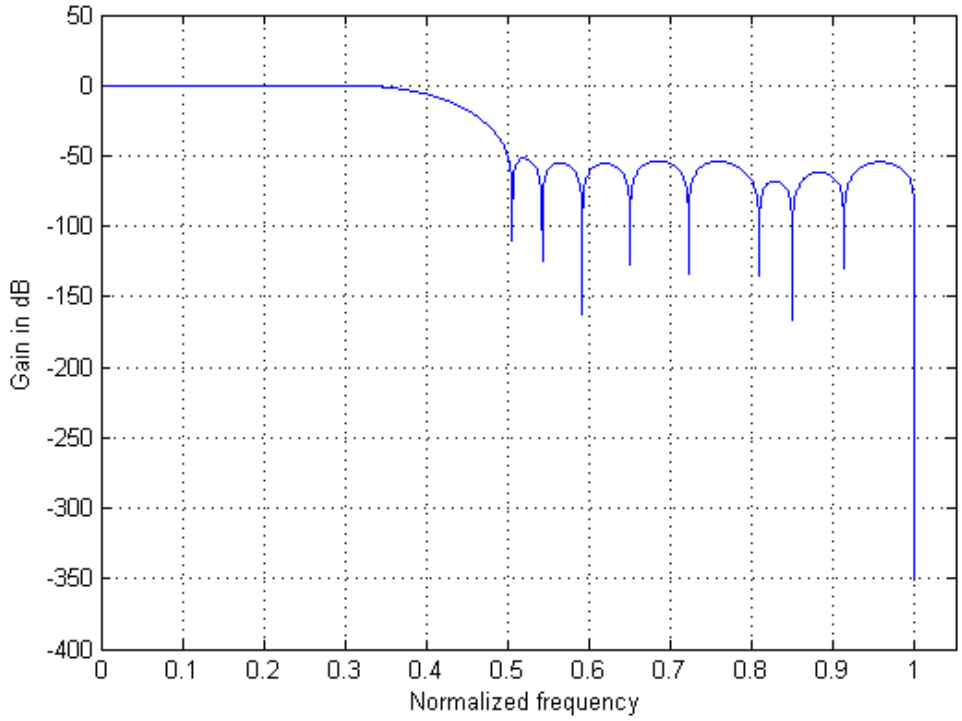


Fig. 5.20. Magnitude response of filter Y1

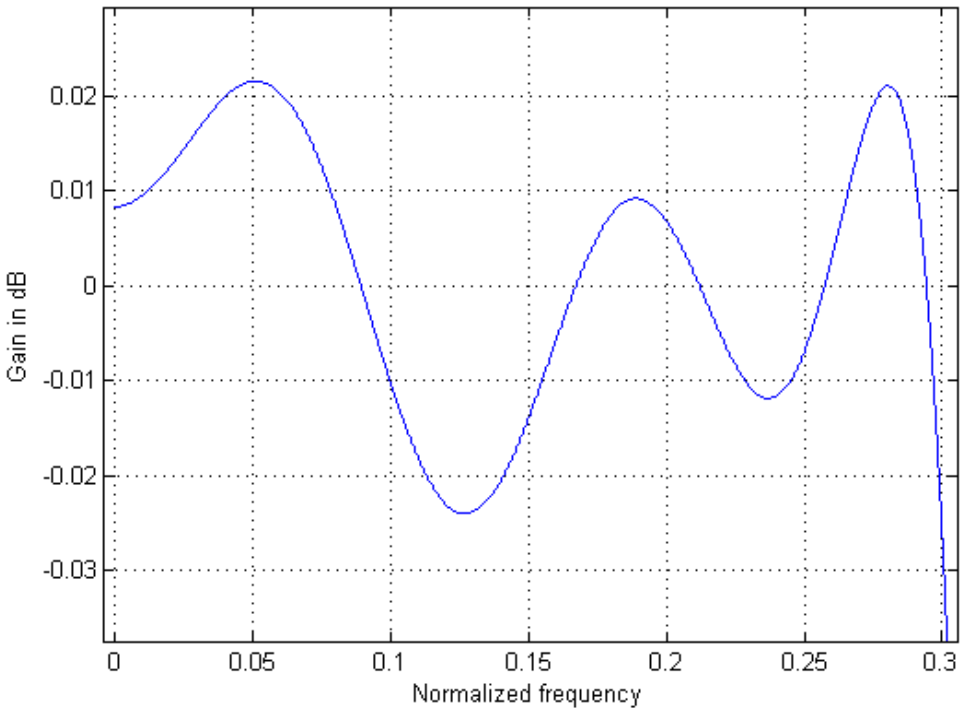


Fig. 5.21. Passband of the magnitude response of filter Y1

5.2.6 Example 6: Design of filter G1

The filter G1 is an even length short filter. The indices of the coefficients with 0 value are found to be $Z \in \{3\}$.

The gain is partitioned into 40 parts. The feasible solution with minimum number of adders in the first step is shown below.

3, 6, 0, -16, -19, 12, 75, 127

The number of adders achieved for this solution is 4 and the minimax error is 0.0091679.

For the second step involving EO algorithm, the solutions within 105% of δ_p are selected. The best solution is chosen as the final solution of the algorithm. The following table shows the filter coefficients, the odd fundamentals (Basis set) and the synthesis of the filter coefficients. The number of adders achieved is 12.

Filter *G1*, $h(n) = h(15 - n)$ for $0 \leq n \leq 7$ Passband gain: 377.00004, $EWL = 7$ 3, 6, 0, -16, -19, 12, 76, 128
Basis Set ={ 3, 19}
$3 = 1\times2^1+1$ $19 = 1\times2^4+3$

Table 5.8. Filter G1 coefficients, basis set and its adder synthesis

The magnitude of the frequency response of the filter G1 is shown in the Figure 5.22 and 5.23.

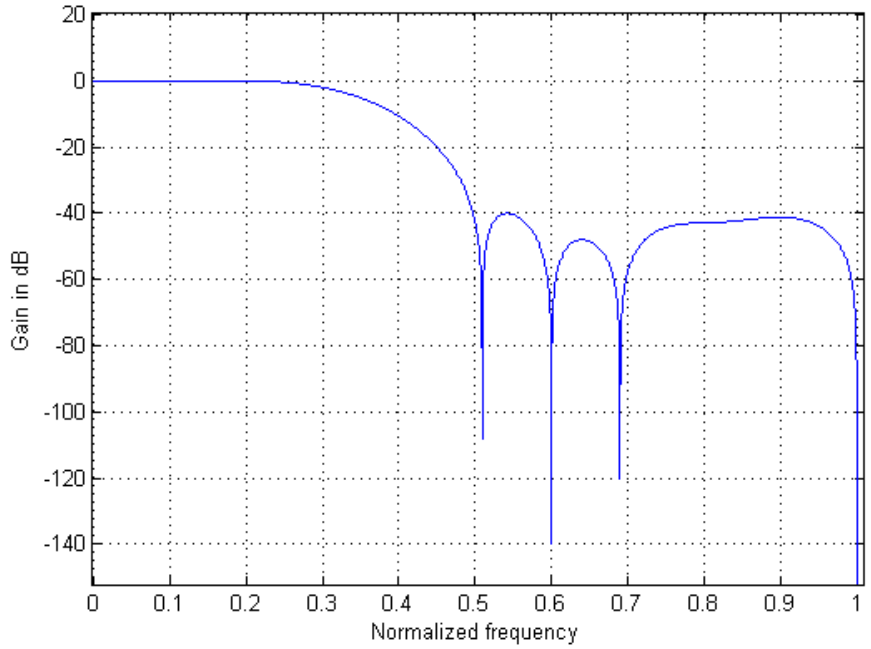


Fig. 5.22. Magnitude response of filter G1

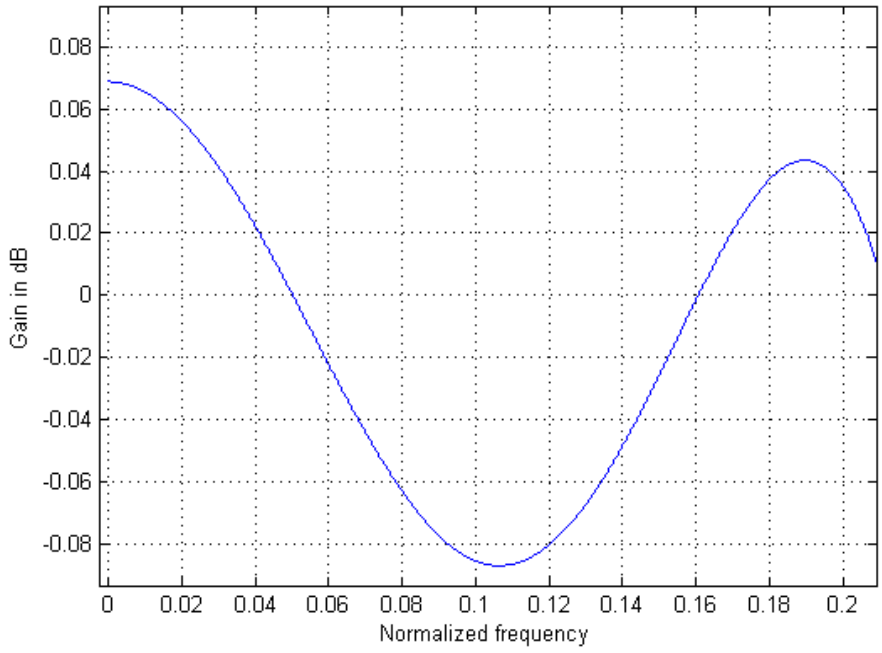


Fig. 5.23. Passband of the magnitude response of filter G1

5.2.7 Example 7: Band-pass filter

The filter order is found to be 112. This is found by first finding the filter order of continuous filter which meets the specifications using the Parks-McClellan method. The indices of the coefficients with 0 value are found to be $Z \in \{11, 24, 29, 31, 37\}$.

The gain is partitioned into 40 parts.

For the second step involving EO algorithm, the solutions within 105% of δ_p are selected. The best solution is chosen as the final solution of the algorithm. The following table shows the filter coefficients, the odd fundamentals (Basis set) and the synthesis of the filter coefficients. The number of adders achieved is 16 and the minimax error is 0.009969. The magnitude of the frequency response of the filter is shown on the next page.

$h(n) = h(112 - n)$ for $0 \leq n \leq 56$ Passband gain: 2101.65208, $EWL = 10$ -4, -2, 1, 2, 2, 4, 3, -4, -7, -3, 0, -2, 2, 11, 8, -4, -5, -1, -7, -13, 2, 17, 9, 0, 7, 4, -20, -26, 0, 12, 0, 12, 35, 13, -32, -28, 0, -17, -28, 29, 72, 22, -26, 3, -4, -91, -86, 55, 104, 14, 41, 164, -4, -409, -384, 239, 643		
Basis Set ={3, 5, 7, 9, 11, 13, 17, 29, 35, 41, 43, 55, 91, 239, 409, 643}		
$3 = 1\times2^1+1$	$5 = 1\times2^2+1$	$7 = 1\times2^3-1$
$9 = 1\times2^3+1$	$11 = 1\times2^3+3$	$13 = 1\times2^4-3$
$17 = 1\times2^4+1$	$29 = 1\times2^5-3$	$35 = 1\times2^5+3$
$41 = 1\times2^5+9$	$43 = 5\times2^3+3$	$55 = 1\times2^6-9$
$91 = 3\times2^5-5$	$239 = 1\times2^8-17$	$409 = 29\times2^4-55$
$643 = 5\times2^7+3$		

Table 5.9. Band pass filter coefficients, basis set and its adder synthesis

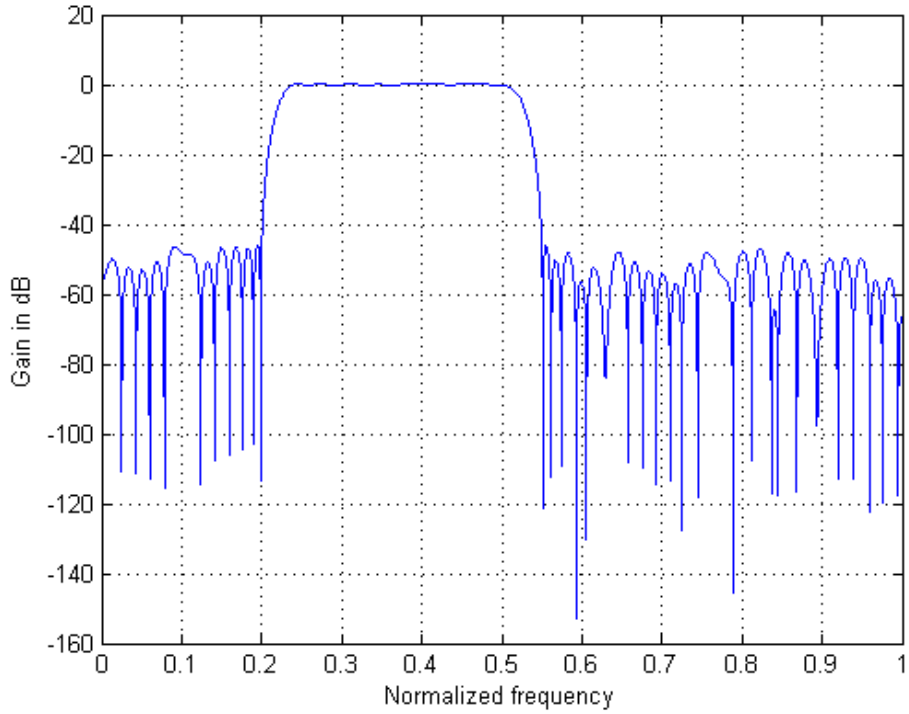


Fig. 5.24. Magnitude response of Bandpass filter

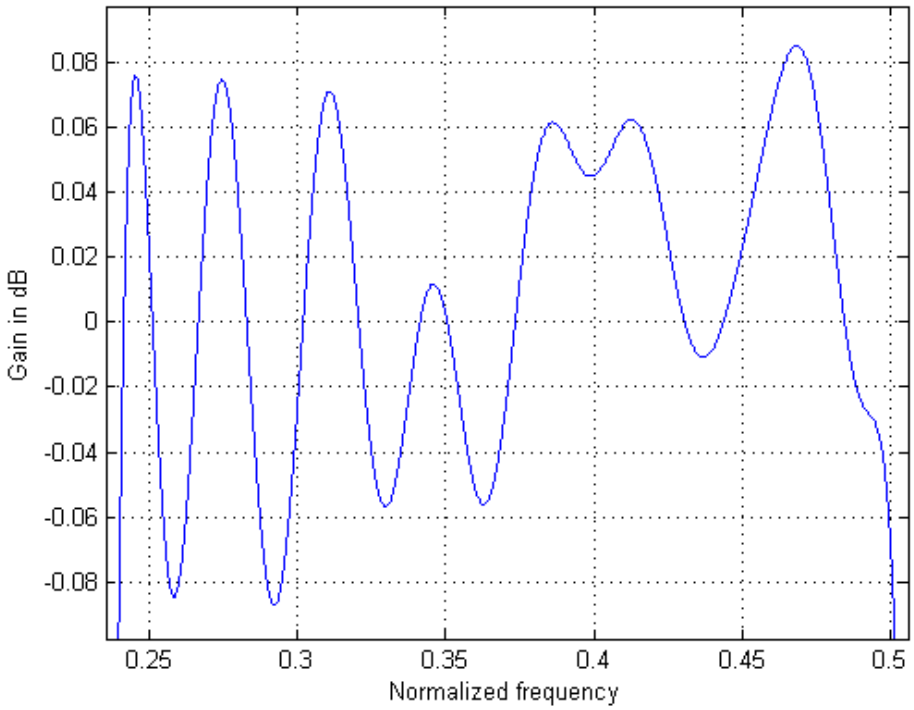


Fig. 5.25. Passband of the magnitude response of Bandpass filter

5.2.8 Example 8: Band-stop filter

The filter order is found to be 104. This is found by first finding the filter order of continuous filter which meets the specifications using the Parks-McClellan method. No coefficient is fixed to zero initially

The gain is partitioned into 40 parts.

For the second step involving EO algorithm, the solutions within 105% of δ_p are selected. The best solution is chosen as the final solution of the algorithm. The following table shows the filter coefficients, the odd fundamentals (Basis set) and the synthesis of the filter coefficients. The number of adders achieved is 12 and the minimax error achieved is 0.0098798. The magnitude of the frequency response of the filter is shown on the next page.

$h(n) = h(104 - n)$ for $0 \leq n \leq 52$ Passband gain: 1122.05977, $EWL = 10$ 3, 2, 1, 0, -3, -4, -4, -4, -2, -1, 0, -1, -2, -4, -5, -4, 0, 5, 10, 12, 10, 7, 2, -1, -1, 2, 7, 10, 8, 0, -10, -22, -28, -28, -20, -7, 3, 7, 2, -9, -19, -19, -4, 26, 64, 97, 111, 94, 48, -19, -90, -144, 960		
Basis Set ={3, 5, 7, 9, 11, 13, 15, 19, 45, 47, 97, 111}		
$3 = 1\times2^1+1$	$5 = 1\times2^2+1$	$7 = 1\times2^3-1$
$9 = 1\times2^3+1$	$11 = 1\times2^3+3$	$13 = 1\times2^4-3$
$15 = 1\times2^4+1$	$19 = 1\times2^4+3$	$45 = 5\times2^3+5$
$47 = 5\times2^3+7$	$97 = 7\times2^4-15$	$111 = 3\times2^5+15$

Table 5.10. Band stop filter coefficients, basis set and its adder synthesis

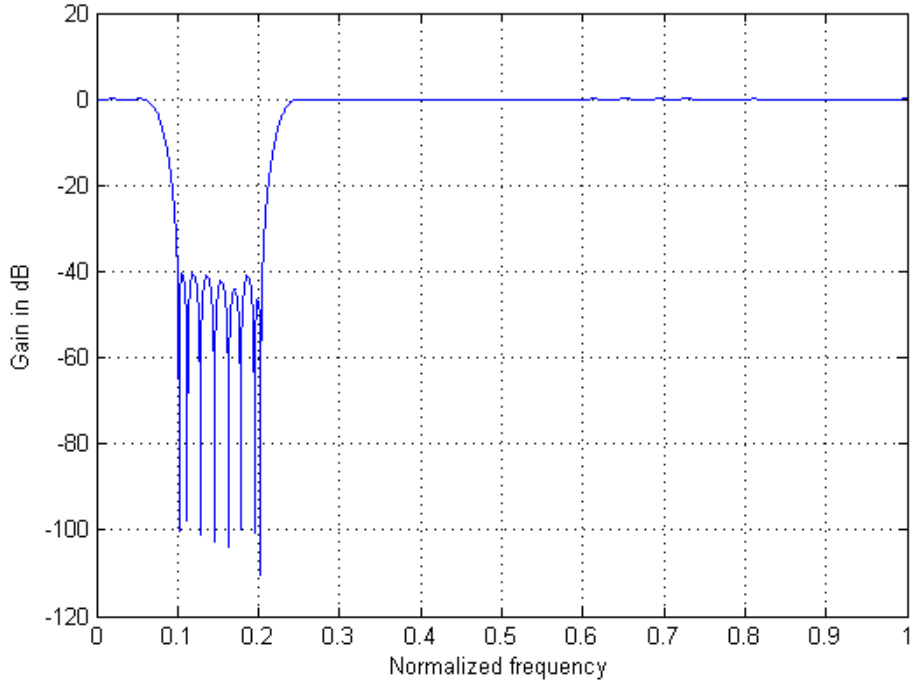


Fig. 5.26. Magnitude response of Band stop filter

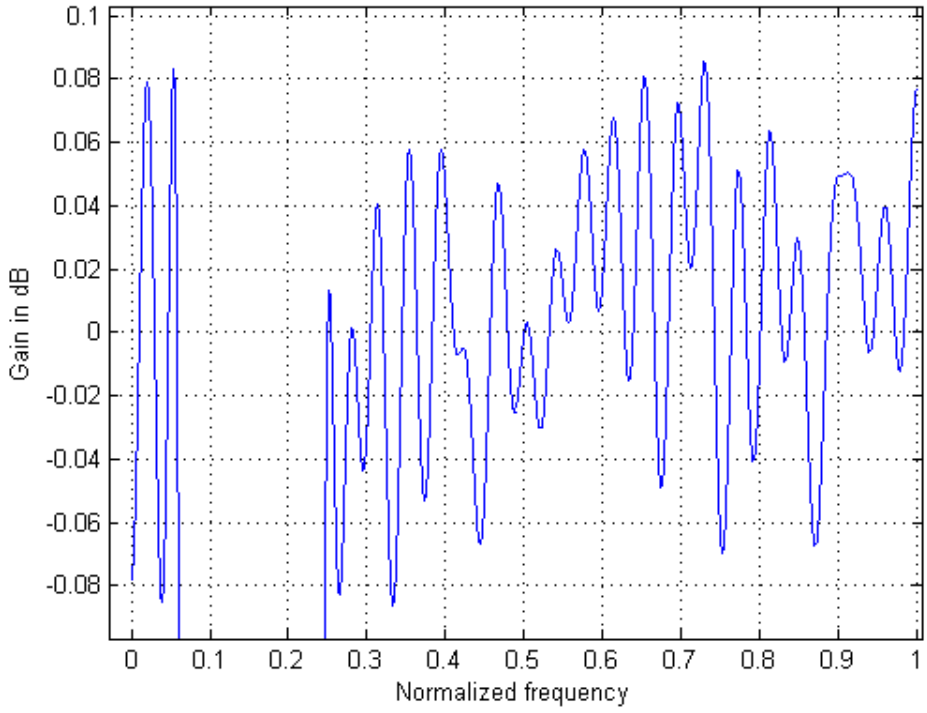


Fig. 5.27. Passband of the magnitude response of Band stop filter

5.3 Summary of the results

The results obtained with the algorithm are compared with other methods from the literature. **Note: NA means not available.**

Filter	Q	EWL	Method	MA	SA	Minimax error
A	14	10	FIRGAM [8]	18	58	NA
			Ye and Yu [9]	14	54	0.009885
			Proposed	15	54	0.009618
S2	14	10	FIRGAM [8]	27	59	NA
			Ye and Yu [9]	16	59	0.01130
			Proposed	18	59	0.01145
B	11	8	FIRGAM [8]	11	100	NA
			Ye and Yu [9]	12	94	0.009965
			GA [12]	10	98	NA
			Proposed	12	94	0.009906
C	13	10	FIRGAM [8]	22	306	NA
			Ye and Yu [9]	28	282	0.004993
			GA [12]	31	306	NA
			Proposed	28	282	0.004995
Y1	12	10	FIRGAM [8]	NA	NA	NA
			Ye and Yu [9]	6	23	0.002955

			Proposed	6	23	0.003029
G1	9	7	FIRGAM [8]	NA	NA	NA
			Shi and Yu [6]	2	13	0.009986
			Proposed	2	13	0.009986

Table 5.11. Comparison of the results with other methods

It can be shown [8] that the size of a structural adder (SA) is less than the size of a multiplier block adder (MA). Therefore for comparing the complexity of two designs, the number of SA has higher priority than MA. If two designs have same number of SA, then MA can be compared to compare the complexities.

Comparing the proposed method with GA [12] and FIRGAM [8], it can be seen that the number of SA is less. Therefore, the proposed method achieves significant reduction in complexity compared to GA [12] and FIRGAM [8]. Comparing the method with [6] and [9], it is seen that the number of SA achieved are same in all cases and number of MA are comparable in most of the cases.

Comparing filter B and C with state of the art GA based method [12]. It can be seen that the algorithm clearly outperforms [12]. The limitation which is noticed is that when the filter word-length is high, as in L1 filter designed in [9] and [12], the proposed method is not able to further improve the solutions obtained from the first step. This is because the neighborhood size required is comparatively large in this case and the optimum solutions are very sparsely distributed. With EWL less than or equal to 10 the proposed method can easily find good solutions.

In case of deterministic methods such as [6], [8] and [9], the proposed method achieves similar results in many cases, though in a significantly less time.

In Table 5.12, the results of a band-pass and a band-stop filter are shown. These filters have not been taken from the literature and it was felt that an example involving a band-pass and –stop filter should be included in this report.

Filter	EWL	Method	MA	SA	Minimax error
Band-pass	10	Proposed	16	102	0.009618
Band-stop	10	Proposed	12	96	0.009880

Table 5.12. Results of Band-pass and Band-stop filters

5.4 Run-time of the algorithm

As the initial solutions are obtained using the tree search method as in [9], therefore the time taken by the algorithm is similar for this step. In the following table the run time of the algorithm for different filters is compared with the other methods.

For low order filters, such as G1 and Y1, although the proposed algorithm can achieve optimum results in a very short time, yet it is impossible for it to do it in less time than the deterministic methods such as [6], [8] and [9]. This is because the search space is very small in these cases and the above methods can exhaustively finish the search very fast.

On the other hand for larger filters with moderate word-lengths the proposed method can significantly improve the total runtime while achieving near optimum results.

Filter	Method	Run-time	MA	SA	Filter	Method	Run-time	MA	SA
A	FIRGAM [8]	4h12m	18	58	Y1	FIRGAM [8]	23s	NA	NA
	Ye and Yu [9]	37m	14	54		Ye and Yu [9]	28s	6	23
	Proposed	15m	15	54		Proposed	2m	6	23
S2	FIRGAM [8]	27m	27	59	G1	FIRGAM [8]	NA	NA	NA
	Ye and Yu [9]	1h47m	16	59		Ye and Yu [6]	<1s	2	13
	Proposed	25m	18	59		Proposed	2m	2	13
B	FIRGAM [8]	24h	11	100	Band-pass	Proposed	~14m	16	102
	Ye and Yu [9]	39m	12	94					
	Proposed	12m	12	94					
C	FIRGAM [8]	24h	22	306	Band-stop	Proposed	~14m	12	96
	Ye and Yu [9]	6h24m	28	282					
	Proposed	3h	28	282					

Table 5.13. Comparison of the run-time with different algorithms

Chapter 6

Conclusion

The algorithm proposed in this paper is able to find the filter coefficients with the number of adders which are on par with the state of the art methods. With other evolutionary methods like GA and ACO it is very difficult to achieve global optimum solutions in a reasonable amount of time. In addition, the major advantage of EO is that it does not require a large number of adjustable parameters compared to other evolutionary methods. The design examples show that feasible results with similar number of adders can be achieved in a much shorter time compared to other methods. The simplicity of the algorithm with just one adjustable parameter, means that the algorithm can be implemented with ease with very less expert intervention when near optimum results are required.

Although the scope of the thesis is to minimize the number of adders (structural and multiplier block), but the proposed algorithm can be used to optimize the filter based on other criteria, such as , number of full adders, total chip area, etc.

The major limitation, which needs to be further addressed is to efficiently deal with higher word-length designs as the performance of the algorithm deteriorates with higher word-lengths (greater than 10). At moderate EWL up to 10 bits the algorithm can achieve good results most of the times.

REFERENCES

[1] T. Ciloglu, “An efficient local search method guided by gradient information for discrete coefficient FIR filter design,” *Signal Processing,* vol. 82, pp. 1337–1350, 2002.

[2] A. Yurdakul and G. Dundar, “Fast and efficient algorithm for the multiplierless realization of linear DSP transforms," *Circuits, Devices and Systems, IEE Proceedings on,* vol. 149, no. 4, pp. 205-211, Aug 2002.

[3] A. Dempster and M. Macleod, “Constant integer multiplication using minimum adders,” *Circuits, Devices and Systems, IEE Proceedings on,* vol. 141, no. 5, pp. 407-413, Oct 1994.

[4] A. Dempster and M. Macleod, “Use of minimum-adder multiplier blocks in FIR digital filters,” *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on,* vol. 42, no. 9, pp. 569-577, Sep 1995.

[5] A. Dempster, S. Dimirsoy and I. Kale, “Designing multiplier blocks with low logic depth,” *Circuits and Systems (ISCAS), IEEE International Symposium on,* vol. 5, pp. V-773-V-776, 2002.

[6] D. Shi and Y. J. Yu, “Design of linear phase FIR filters with high probability of achieving minimum number of adders,” *Circuits and Systems I: Regular Papers, IEEE Transactions on,* vol. 58, no. 1, pp. 126-136, Jan. 2011.

[7] W. B. Ye and Y. J. Yu, “A polynomial-time algorithm for the design of multiplierless linear-phase FIR filters with low hardware cost,” *Circuits and Systems (ISCAS), IEEE International Symposium on,* pp. 970-973, 1-5 June 2014.

[8] M. Aktan, A. Yurdakul and G. Dundar, “An Algorithm for the Design of Low-Power Hardware-Efficient FIR Filters,” *Circuits and Systems I: Regular Papers, IEEE Transactions on,* vol. 55, no. 6, pp. 1536, 1545, July 2008.

[9] W. B. Ye and Y. J. Yu, “Two-Step Optimization Approach for the Design of Multiplierless Linear-Phase FIR Filters,” *Circuits and Systems I: Regular Papers, IEEE Transactions on,* vol. 62, no. 5, pp. 1279-1287, May 2015.

[10] P. Gentili, F. Piazza and A. Uncini, “Efficient genetic algorithm design for power-of-two FIR filters,” *Acoustics, Speech, and Signal Processing, International Conference on,* vol. 2, pp. 1268-1271, 1995.

[11] W. B. Ye and Y. J. Yu, “Design of high order and wide coefficient wordlength multiplierless FIR filters with low hardware cost using genetic algorithm,” *Circuits and Systems (ISCAS), IEEE International Symposium on,* pp. 45-48, 20-23 May 2012.

[12] W. B. Ye and Y. J. Yu, “Single-Stage and Cascade Design of High Order Multiplierless Linear Phase FIR Filters Using Genetic Algorithm,” *Circuits and Systems I: Regular Papers, IEEE Transactions on,* vol. 60, no. 11, pp. 2987-2997, Nov. 2013.

[13] Chao-Liang Chen and A. N. Willson, “A trellis search algorithm for the design of FIR filters with signed-powers-of-two coefficients,” *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on,* vol. 46, no. 1, pp. 29-39, Jan 1999.

[14] H. Samueli, “On the design of optimal equiripple FIR digital filters for data transmission applications,” *Circuits and Systems, IEEE Transactions on,* vol. 35, no. 12, pp. 1542-1546, Dec 1988.

[15] H. Zhao, W. B. Ye and Y. J. Yu, “Sparse FIR filter design based on Genetic Algorithm,” *Circuits and Systems (ISCAS), IEEE International Symposium on,* pp. 97-100, 19-23 May 2013.

[16] L. Cen and Y. Lian, “A modified micro-genetic algorithm for the design of multiplierless digital FIR filters,” *Region 10 Conference, Chiang Mai, Thailand, IEEE Proceedings of,* vol. 1, pp. 52–55, Nov. 2004.

[17] L. Cen, “A hybrid genetic algorithm for the design of FIR filters with SPoT coefficients,” *Signal Processing,* vol. 87, pp. 528–540, 2007.

[18] S. J. Gould and N. Eldridge, “Punctuated Equilibria: The Tempo and Mode of Evolution Reconsidered,” *Paleobiology,* vol. 3, pp. 115–151, 1977.

[19] P. Bak and K. Sneppen, “Punctuated equilibrium and criticality in a simple model of evolution,” *Physical Review Letters,* vol. 71, no. 24, pp. 4083–4086, 1993.

[20] S. Boettcher and A. G. Percus, “Extremal Optimization: Methods derived from Co-Evolution,” *Genetic and Evolutionary Computation Conference, Proceedings of,* pp. 825-832, 1999.

[21] S. Boettcher and A. G. Percus, “Optimization with Extremal Dynamics,” *Physical Review Letters,* vol. 86, no. 23, pp. 5211–5214, 2001.

[22] F. L. Sousa and F. M. Ramos, “Function Optimization using Extremal Dynamics,” *4th International Conference on Inverse Problems in Engineering, Proceedings of,* Rio de Janeiro, Brazil, 2002.

[23] M. r. Chen, Y. z. Lu and G. Yang, “Population-Based Extremal Optimization with Adaptive Lévy Mutation for Constrained Optimization,” *Computational Intelligence and Security, International Conference on,* vol. 1, pp. 258-261, 3-6 Nov. 2006.

VITA AUCTORIS

Manpreet Singh Malhi was born in India in 1985. He received his Bachelors in Technology in 2010 from Indian Institute of Technology, Roorkee, India. He received Master of Science in Communication Engineering in 2012 from University of Technology, Munich, Germany. In 2014 he started his master's program in ECE department at University of Windsor, Canada.