Design of Digital Filters Using Genetic Algorithms

by

Sabbir U. Ahmad

B.Sc. Eng., Chittagong University of Engineering and Technology, Bangladesh, 1992
M.Eng., Nanyang Technological University, Singapore, 2001

A Dissertation Submitted in Partial Fullfillment of the
Requirements for the Degree of

Doctor of Philosophy

in the Department of Electrical and Computer Engineering

Design of Digital Filters Using Genetic Algorithms

by

Sabbir U. Ahmad

B.Sc. Eng., Chittagong University of Engineering and Technology, Bangladesh, 1992
M.Eng., Nanyang Technological University, Singapore, 2001

Supervisory Committee

Dr. Andreas Antoniou, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Wu-Sheng Lu, Department Member
(Department of Electrical and Computer Engineering)

Dr. Pan Agathoklis, Department Member
(Department of Electrical and Computer Engineering)

Dr. Zuomin Dong, Outside Member
(Department of Mechanical Engineering)

Supervisory Committee

Dr. Andreas Antoniou, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Wu-Sheng Lu, Department Member
(Department of Electrical and Computer Engineering)

Dr. Pan Agathoklis, Department Member
(Department of Electrical and Computer Engineering)

Dr. Zuomin Dong, Outside Member
(Department of Mechanical Engineering)

Abstract

In recent years, genetic algorithms (GAs) began to be used in many disciplines such as pattern recognition, robotics, biology, and medicine to name just a few. GAs are based on Darwin's principle of natural selection which happens to be a slow process and, as a result, these algorithms tend to require a large amount of computation. However, they offer certain advantages as well over classical gradient-based optimization algorithms such as steepest-descent and Newton-type algorithms. For example, having located local suboptimal solutions they can discard them in favor of more promising local solutions and, therefore, they are more likely to obtain better solutions in multimodal problems. By contrast, classical optimization algorithms though very efficient, they are not equipped to discard inferior local solutions in favour of more optimal ones.

This dissertation is concerned with the design of several types of digital filters by using GAs as detailed bellow.

In Chap. 2, two approaches for the design of fractional delay (FD) filters based on a GA are developed. The approaches exploit the advantages of a global search technique to determine the coefficients of FD FIR and allpass-IIR filters based on the so-called Farrow structure. The GA approach was compared with a least-squares approach and was found to lead to improvements in the amplitude response and/or delay characteristic.

In Chap. 3, a GA-based approach is developed for the design of delay equalizers. In this approach, the equalizer coefficients are optimized using an objective function based on the passband filter-equalizer group delay. The required equalizer is built by adding new second-order sections until the desired accuracy in terms of the flatness of the group delay with respect to the passband is achieved. With this approach stable delay equalizers satisfying arbitrary prescribed specifications with the desired degree of group-delay flatness can easily be obtained.

In Chap. 4, a GA-based approach for the design of multiplierless FIR filters is developed. A recently-introduced GA, called orthogonal GA (OGA) based on the so-called *experimental design* technique, is exploited to obtain fixed-point implementations of linear-phase FIR filters. In this approach, the effects of finite word length are minimized by considering the filter as a cascade of two sections. The OGA leads to an improved amplitude response relative to that of an equivalent direct-form cascade filter obtained using the Remez exchange algorithm.

In Chap. 5, a multiobjective GA for the design of asymmetric FIR filters is proposed. This GA uses a specially tailored *elitist nondominated sorting GA* (ENSGA) to obtain so-called *Pareto-optimal* solutions for the problem at hand. Flexibility is introduced in the design by imposing phase-response linearity only in the passband instead of the entire baseband as in conventional designs. Three objective

functions based on the amplitude-response error and the flatness of the group-delay characteristic are explored in the design examples considered. When compared with a WLS design method, the ENSGA was found to lead to improvements in the amplitude response and passband group-delay characteristic.

In Chap. 6, a hybrid approach for the design of IIR filters using a GA along with a quasi-Newton (QN) algorithm is developed. The hybrid algorithm, referenced to as the *genetic quasi-Newton* (GQN) algorithm combines the flexibility and reliability inherent in the GA with the fast convergence and precision of the QN algorithm. The GA is used as a global search tool to explore different regions in the parameter space whereas the QN algorithm exploits the efficiency of a gradient-based algorithm in locating local solutions. The GQN algorithm works well with an arbitrary random initialization and filters that would satisfy prescribed amplitude-response specifications can easily be designed.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AIFS	allpass IIR Farrow structure
CPU	central processing unit
EA	evolutionary algorithm
ENSGA	elitist nondominated sorting genetic algorithm
EOA	extended orthogonal array
FD	fractional delay
FDFS	fractional-delay FIR filters based on Farrow structure
FIR	finite-duration impulse response
FPGA	field-programmable gate arrays
FRM	frequency-response masking
FS	Farrow structure
GA	genetic algorithm
GC	global cycle
GQN	genetic quasi-Newton
IIR	infinite-duration impulse response
LES	local elite solution
LS	least-squares
MILP	mixed-integer linear programming
NDL	nondominated level
OA	orthogonal array
OGA	orthogonal genetic algorithm
PLD	programmable logic devices
PM	polynomial mutation
POT	powers of two
QN	quasi-Newton

SBX	simulated binary crossover
sGA	structured genetic algorithm
SOPOT	sums of powers of two
WLS	weighted least-squares

Acknowledgments

First and foremost I would like to express my gratitude to my supervisor Professor Andreas Antoniou for his superb mentorship, encouragement, and support during the course of my graduate studies at the University of Victoria. Without his guidance and willingness to help, I would not have completed this work. At the same time, I would like to thank the members of my supervisory committee for their time and effort in reviewing the dissertation. The friendly and supportive environment of the Digital Signal Processing Group at UVic has enlightened me and contributed essentially to the final outcome of my studies. I would like to thank my fellow students, especially Drs. Nanyan Wang and Stuart Bergen. I would also take this opportunity to thank the staff of the Department of Electrical and Computer Engineering for their support especially Steve Campbell, Erik Laxdal, Vicky Smith, and Lynne Barrett.

For lack of space, it is not possible to mention many other people who have in some way influenced my work for this dissertation. However, it is impossible to leave out my wife Rownak Afroze who contributed through her continuous support in every aspect of my work. I am deeply indebted and thankful for her extraordinary passion and care without which it would have been a rather difficult path to traverse. I am also very thankful to my two lovely kids Shuhrat and Orrin for their unusual patience while deprived from having fun with their dad during these years. I would also like to thank my friends Stephen Boppart, Jorge Flaminman, and Dr. Jahangir Hossain for their great encouragement to finish this work. Last, but not the least, the greatest thanks go to my mother Syeda Layeka Begum whose extreme sacrifice has made my journey possible this far.

Dedication

Two most important persons in my life - my mother and wife

Chapter 1

Introduction

Digital filters are used in numerous applications from control systems, systems for audio and video processing, and communication systems to systems for medical applications to name just a few. They can be implemented in hardware or software and can process both real-time and off-line (recorded) signals. Digital filters in hardware form can now routinely perform tasks that were almost exclusively performed by analog systems in the past whereas software digital filters can be implemented using low-level or user-friendly high-level programming languages.

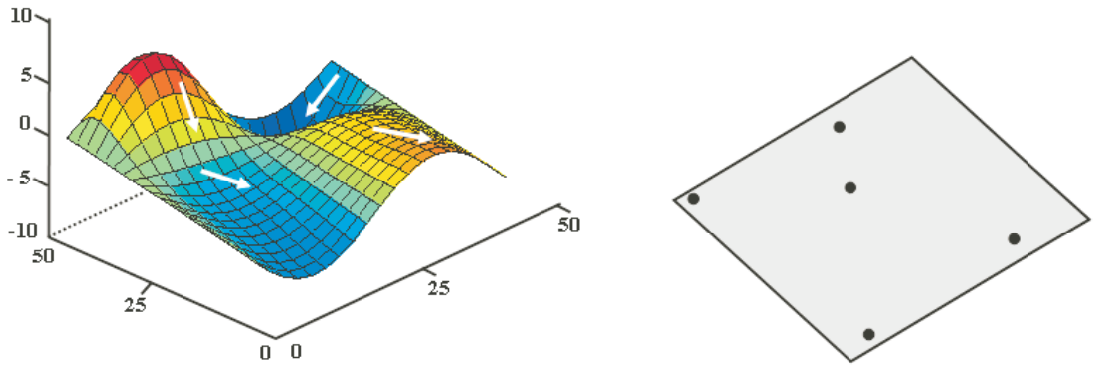
Nowadays digital filters can be used to perform many filtering tasks which in the not so distant past were performed almost exclusively by analog filters and are replacing the traditional role of analog filters in many applications. Beside the inherent advantages, such as, high accuracy and reliability, small physical size, and reduced sensitivity to component tolerances or drift, digital implementations allow one to achieve certain characteristics not possible with analog implementations such as exact linear phase and multirate operation. Digital filtering can be applied to very low frequency signals, such as those occurring in biomedical and seismic applications very efficiently. In addition, the characteristics of digital filters can be changed or adapted by simply changing the content of a finite number of registers, thus multiple

filtering tasks can be performed by one *programmable* digital filter without the need to replicate the hardware. With the ever increasing number of applications involving digital filters, the variety of requirements that have to be met by digital fitters has increased. As a result, state-of-the-art design techniques that are capable of satisfying sophisticated design requirements are becoming an impregnable necessity. In what follows, we provide a general background and motivation for the specific research work reported in this dissertation.

1.1 General Background and Motivation

Like most other engineering problems, the design of digital filters involves multiple, often conflicting, design criteria and specifications, and finding an optimum design is, therefore, not a simple task. Analytic or simple iterative methods usually lead to sub-optimal designs. Consequently, there is a need for optimization-based methods that can be used to design digital filters that would satisfy prescribed specifications [1–3]. However, optimization problems for the design of digital filters are often complex, highly nonlinear, and multimodal in nature (see p. 725 of [1] and [4–6]). The problems usually exhibit many local *minima*. A view of the solution space in a typical multimodal problem is illustrated in Fig. 1.1. Ideally, the optimization method should lead to the global optimum of the objective function with a minimum amount of computation. Classical optimization methods are generally fast and efficient, and have been found to work reasonably well for the design of digital filters [1]. These methods are very good in locating local minima but unfortunately, they are not designed to discard inferior local solutions in favor of better ones. Therefore, they tend to locate minima in the locale of the initialization point.

In recent years, a variety of algorithms have been proposed for global optimization including *stochastic* or *heuristic* algorithms [7]. Stochastic algorithms involve



(a) (b)

Figure 1.1: (a) Two-variable optimization problem, (b) local minima.

randomness and/or statistical arguments and in some instances are based on analogies with natural processes [8]. The algorithms based on the mechanics of natural selection and genetics have come to be known collectively as evolutionary algorithms (EAs) [9]. Well-known examples of such algorithms are *genetic algorithms* (GAs), *evolutionary strategies*, *genetic programming*, *ant colony optimization*, and *particle swarm optimization*. Among these algorithms, GAs are perhaps the most widely known types of EAs today [10].

GAs received considerable attention about their potentials as novel optimization technique for complex problems, especially for the problem with nondifferentiable solution space [11]. While these algorithms tend to require a large amount of computation, they also offer certain unique features with respect to classical gradient-based algorithms. For example, having located local suboptimal solutions, GAs can discard them in favour of more promising subsequent local solutions and, therefore, in the long run they are more likely to obtain better solutions for multimodal problems [12]. GAs are also very flexible, nonproblem specific, and robust [13]. Furthermore, owing to their heuristic nature, arbitrary constraints can be imposed on the objective

function without increasing the mathematical complexity of the problem.

Because of their inherent characteristics, GAs have been suggested for numerous applications such as pattern recognition, robotics, biology, and medicine. These algorithms have also been suggested for various digital signal processing applications, for example, in adaptive estimation of time delay between sampled signals [14, 15], fingerprint matching [16], pattern recognition [17], and speech recognition [18].

The work described in this dissertation explores the use of genetic algorithms for the design of several types of digital filters.

1.2 Genetic Algorithms - Concept and Configurations

1.2.1 Introduction

GAs are stochastic search methods that can be used to search for an optimal solution to the evolution function of an optimization problem [19]. Holland proposed GAs in the early seventies [20] as computer programs that mimic the natural evolutionary process. De Jong extended GAs to functional optimization [21] and a detailed mathematical model of a GA was presented by Goldberg in [9].

GAs differ from classical optimization and search methods in several respects. Rather than focusing on a single solution, GAs operate on group of trial solutions in parallel where they manipulate a *population* of individuals in each *generation* (iteration) where each individual, termed as the *chromosome*, represents one candidate solution to the problem. Within the population, fit individuals survive to reproduce and their genetic materials are recombined to produce new individuals as *offsprings*. The genetic material is modeled by some finite-length data structures. As in nature, *selection* provides the necessary driving mechanism for better solutions to survive. Each solution is associated with a *fitness* value that reflects how good

it is compared with other solutions in the population. The recombination process is simulated through a *crossover* mechanism that exchanges portions of data strings between chromosomes. New genetic material is also introduced through *mutation* that causes random alterations of the strings. The frequency of occurrence of these genetic operations is controlled by certain pre-set probabilities. The selection, crossover, and mutation processes constitute the basic GA cycle or generation, which is repeated until some pre-determined criteria are satisfied. Through this process, successively better and better individuals of the species are generated.

In a nutshell, a GA entails four fundamental steps as follows:

- *Step 1*: Create an initial population of random solutions (chromosomes) by some means.
- *Step 2*: Assess the chromosomes for fitness using the criteria imposed on the required solution and create an elite set of chromosomes by selecting a number of chromosomes that best satisfy the requirements imposed on the solution.
- *Step 3*: If the top-ranking chromosome in the elite set satisfies fully the requirements imposed on the solution, output that chromosome as the required solution, and stop. Otherwise, continue to *Step 4*.
- *Step 4*: Apply crossover between pairs of chromosomes in the elite set to generate more chromosomes and subject certain chromosomes chosen at random to mutations, and repeat from *Step 2*.

A schematic representation of the genetic search approach is presented in Fig. 1.2. In the reminder of this section, the aspects associated to the fundamental steps of GAs, described above, such as *chromosome representation*, *encoding schemes*, *population initialization*, *fitness function*, *genetic operators*, and *selection methods*

are discussed. We avoid presenting a detailed study on GAs including the theoretical analysis based on so-called *schema theorem* since a rich literature is available on that subject. Instead, we offer a general overview to the important aspects that are necessary to the configuration of GAs.

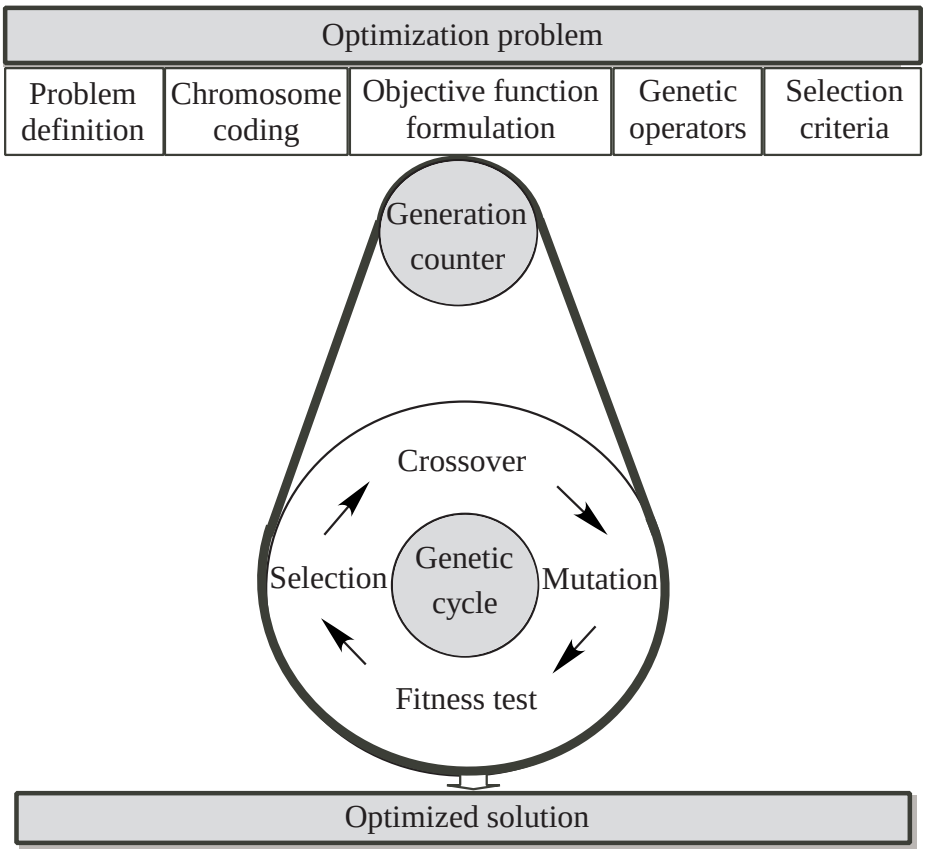


Figure 1.2: Conceptual representation of the optimization process through a genetic algorithm.

1.2.2 Chromosome Representation

In the most basic form, GA works as a function optimizer with a given objective function

$$minimize\ f(\mathbf{a}) \quad \text{where } \mathbf{a} = [a_1\ a_2\ \cdots\ a_M]^T \tag{1.1}$$

In general, GAs operate on a symbolic representation of the design variables known as a chromosome. This requires an encoding function of the form

$$T\ :\ S_a\ \mapsto\ X \tag{1.2}$$

to map the solution space S_a of the problem onto the chromosome space X [22]. By analogy with the biological terminology, the encoded chromosomes are called the *genotype* representation and the corresponding solutions in the search space are called the *phenotype* representation. A chromosome $\mathbf{x}$ is the encoded version, i.e., genotype, of a solution with phenotype $\mathbf{a}$ and they are related by

$$\mathbf{a} = T(\mathbf{x}) \tag{1.3}$$

where

$$\mathbf{x} = [\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_M]^T \tag{1.4}$$

Each element $\mathbf{x}_i$ in a chromosome $\mathbf{x}$ is often referred to as a *gene*. In turn, a gene is usually constructed from a number of elements called *alleles*, e.g. $\mathbf{x}_i = [x_{i1}\ x_{i2}\ \cdots\ ]^T$. If the number of alleles in a gene is J, vector $\mathbf{x}$ can be further expanded as

$$\mathbf{x} = [x_{11}\ x_{12}\ \cdots\ x_{1J}\ x_{21}\ x_{22}\ \cdots\ x_{2J}\ \cdots\ x_{M1}\ x_{M2}\ \cdots\ x_{MJ}]^T \tag{1.5}$$

and if we let $M \times J = N_x$, the chromosome vector can be expressed as

$$\mathbf{x} = [g_1\ g_2\ \cdots\ g_{N_x}]^T \tag{1.6}$$

1.2.3 Encoding Schemes

GAs use various encoding schemes, such as, binary encoding, integer encoding, Gray encoding, and decimal encoding. In binary encoding scheme, each variable is encoded into a bit string of predefined length whereas integers are used as the elements of chromosome vectors in integer encoding. Gray encoding is a variant of binary encoding scheme except it maintains a minimum *Hamming distance* between adjacent numbers where the adjacent numbers differ in one bit. The elements of chromosome vectors are represented using decimal numbers in decimal encoding. This scheme is also called *real encoding*. The choice of encoding is the most important factor in designing a genetic algorithm. The encoding has profound implications on the performance of the GA. Several strategies have been suggested for selecting an encoding scheme but until there is more rigorous theory on GAs and the different encodings, the best strategy seems to be to choose an encoding that is naturally suited for the problem at hand and then design a genetic algorithm that can handle this encoding [23]. For example, binary encoding is useful if the variables of the problem are discrete whereas decimal encoding might be necessary when high-precision is required.

As in Holland's original genetic algorithm, binary encoding is the traditional way to represent parameters in most GAs. To use binary encoding with numeric domains, the binary representation of a gene $\mathbf{x}_m = [x_{m1}\ x_{m2}\ \cdots\ x_{mJ}]^T$ can be mapped (decoded) onto a real number a_m through a simple linear transformation

$$a_m \;=\; a_{min} \;+\; \frac{a_{max} - a_{min}}{2^J - 1} \left(\sum_{n=1}^{J-1} x_{mn}\ 2^{J-1-n} \right) \quad \text{for } m = 1, 2, \ldots, M \qquad (1.7)$$

where a_m takes values ranging from a_{min} to a_{max}, and x_{mn} represents the nth bit in the mth gene in the binary encoding. However, the decoding operation can be entirely avoided in decimal-encoded GAs since they use real-valued genes $\mathbf{x}_m$ to construct

chromosome $\mathbf{x}$. The genotype to phenotype mappings are done through the simple relation $a_m = x_m$ so that $J = 1$.

1.2.4 Population Initialization

Conceptually, GAs maintain a population of N_p chromosomes that are selected and created in an iterative process. The population size can be variable but is usually fixed. The population $\mathbf{P}^t$ at generation t can be denoted as a set of chromosomes as

$$\mathbf{P}^t = \left\{\mathbf{x}^t(1), \mathbf{x}^t(2), \ldots, \mathbf{x}^t(N_p)\right\} \tag{1.8}$$

To commence the iteration, the GA usually generates a random initial population $\mathbf{P}^0$. Other initialization schemes are possible. The initialization does not need to be purely random. A priori knowledge of the problem domain is sometimes invoked to seed $\mathbf{P}^0$ with good chromosomes. The seed can be obtained by using a classical optimization method. Then by applying some heuristic technique or through perturbations of an initial population $\mathbf{P}^0$ can be generated. The population can also be initialized through a deterministic uniform distribution or by using a combination of two or more of the stated schemes. Once an initial population $\mathbf{P}^0$ is created, the main GA cycle can begin. The iteration process is illustrated in terms of pseudocode in Table 1.1.

1.2.5 Fitness Function

The GA produces a succession of populations whose members will have generally improving adaptability to the environment. In order to drive the search of the GA, the fitness levels of the individuals in the population is evaluated by using a fitness function.

The fitness function is usually an objective or cost function but anything will suffice as long as it can successfully quantify the quality of all possible phenotype

Table 1.1: GA Iteration on Successive Population

GA Main
$t = 0$
initialize population $\mathbf{P}^0$
evaluate population $\mathbf{P}^0$
while (! 'termination condition') {
$t = t + 1$
select population $\mathbf{P}_1^t$ from $\mathbf{P}^{(t-1)}$
generate population $\mathbf{P}^t$ from $\mathbf{P}_1^t$
evaluate population $\mathbf{P}^t$
}

solutions. The fitness function is dependent on the environment and application of the system that is undergoing the genetic search process, and it is the only connection between the physical problem being optimized and the genetic algorithm itself [24].

Given a population $\mathbf{P}^t$ at generation t, the GA iteration starts by evaluating the set

$$\mathbf{F}^t = \left\{ f^t(1), f^t(2), \ldots, f^t(N_p) \right\} \tag{1.9}$$

of objective function values associated with the chromosomes $\{\mathbf{x}^t(k)\}$ with $k = 1, 2, ..., N_p$. The GA then applies the genetic operators and selection to produce population $\mathbf{P}^{t+1}$ for the next generation.

The objective function for GAs is formulated as in classical optimization algorithms. However, the GAs do not need gradient information. Therefore, the mathematical structure of these algorithms is simple and flexible. Multiobjective variants of GAs can handle problems with multiple, often conflicting, optimization goals.

1.2.6 Genetic Operators

Evolution from generation to generation is simulated by preserving, redistributing or altering genetic material contained in the chromosome strings of fit individuals. These basic functionalities of the genetic algorithm are provided by the genetic operators. The basic GA operators, crossover and mutation, constitute the main algorithm whereas the population and fitness function can be viewed as external entities. Both crossover and mutation are probabilistic operations and their frequencies of occurrence are controlled by predefined probabilities, P_x and P_m, respectively. As crossover plays the key role in improving the solution, it is assigned a high frequency of occurrence, typically 80-90%. The frequency of occurrence of mutation is kept fairly low, typically 5-10%, to prevent the GA from producing a large number of random solutions.

Crossover

Crossover recombines genetic material from selected individuals to form one or more offspring where some of the useful traits of the parents are preserved. The goal is to generate new chromosomes that are more fit than their ancestors, thereby contributing to the overall convergence of the population. There are many ways of performing crossover. One-point, two-point, or uniform crossover is used with binary encoding. Arithmetic crossover, perturbation or simulated binary crossover is used with decimal or real encoding.

During a one-point crossover, two individuals $\mathbf{x}(1) = [g_1\ g_2\ \cdots\ g_{N_x}]^T$ and $\mathbf{x}(2) = [g'_1\ g'_2\ \cdots\ g'_{N_x}]^T$ selected randomly from $\mathbf{P}$ undergo crossover if a random number u generated usually from a uniform range of numbers $U \in [0, 1]$ is smaller than the probability threshold P_x. Parts of the stings from each individual are swapped at the same location called the *crossover point* to create two offspring chromosomes $\mathbf{x}^c(1)$

and $\mathbf{x}^c(2)$ as follows

$$\begin{aligned} \mathbf{x}^c(1) &= \left[g_1\ g_2\ \cdots\ g_i\ g'_{i+1}\ \cdots\ g'_{N_x}\right]^T \\ \mathbf{x}^c(2) &= \left[g'_1\ g'_2\ \cdots\ g'_i\ g_{i+1}\ \cdots\ g_{N_x}\right]^T \end{aligned} \tag{1.10}$$

The crossover point i in Eqn. 1.10 is chosen randomly from a set of integers

$$I = \{i \in \mathbf{R} :\ 1 \leq i \leq N_x - 1\}$$

One-point crossover is illustrated in Fig. 1.3.

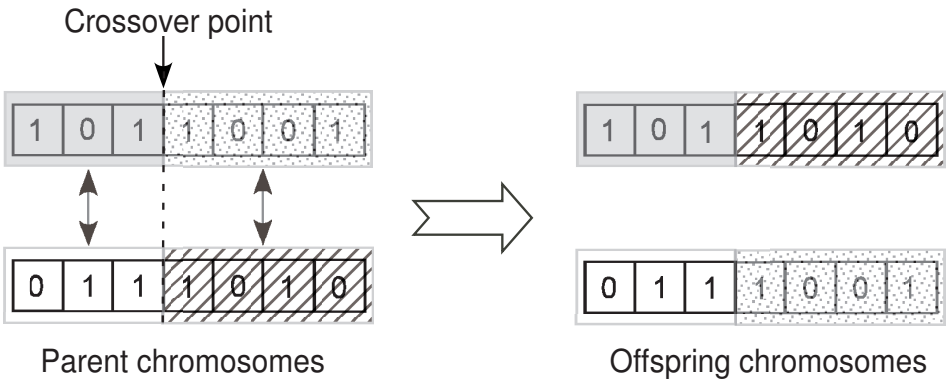


Figure 1.3: A typical one-point crossover in binary representation.

In two-point crossover, the chromosomes to be mated are split at two points and the central sets of genes are exchanged. Two-point crossover allows the alleles at the ends of chromosome strings to stay together. In some instances this is beneficial compared to the case of one-point crossover in order to keep the crossover operation less disruptive when larger chromosome stings are involved. On the other hand, uniform crossover is more disruptive to the population but this makes it better suited for exploring a specified domain of the solution space. In this type of crossover each parent's genes are exchanged with a given probability of occurrence such that each gene in the offspring has an equal probability of originating from either of the parents.

The uniform crossover technique will be discussed further in Section 2.3.2 of Chapter 2.

In a real-coded GA, the direct representation in terms of real values allows the crossover operators to be based on arithmetic operations and stochastic distributions. In the arithmetic crossover, an offspring string is generated using a weighted mean of the genes of the parent strings $\mathbf{x}(1)$ and $\mathbf{x}(2)$ as

$$\mathbf{x}^c(1) \;=\; w\mathbf{x}(1) \;+\; (1-w)\mathbf{x}(2) \tag{1.11}$$

where w is a weight often generated from a uniform distribution $U(0,1)$. In some cases, a weight vector $\mathbf{w} = [w_1 \; w_2 \; \cdots \; w_M]^T$ is used with each element w_k for each element x'_k in the chromosome vector $\mathbf{x}^c = [x'_1 \; x'_2 \; \cdots \; x'_M]^T$.

In the perturbation-based crossover technique, a new chromosome $\mathbf{x}'$ is created by adding a randomly generated vector $\mathbf{r} = [r_1 \; r_2 \; \cdots \; r_M]^T$ to the parent chromosome $\mathbf{x}$, i.e.,

$$\mathbf{x}' \;=\; \mathbf{x} \;+\; \mathbf{r} \tag{1.12}$$

where $\mathbf{r}$ is generated by using a Gaussian or unform distribution.

Simulated binary crossover is another crossover technique for real-encoded GAs which is designed to imitate one-point binary crossover. This technique will be discussed in more detail in Chapter 5.

Mutation

Mutation randomly changes an offspring after crossover. Mutation is treated as supporting operator for the purpose of restoring lost genetic material. Bit-flip mutation is the most common mutation operator for binary-encoded GAs. This is realized by simply inverting one or more bits in the chromosome string based on the probability of mutation, P_m. The mutation operator creates a mutated (new)

chromosome $\mathbf{x}^m$ from $\mathbf{x}$ as follows

$$\mathbf{x}^m = \left[g'_1\ g'_2\ \cdots\ g'_{N_x}\right]^T \tag{1.13}$$

where

$$g'_j = \begin{cases} \mu\left[g_j\right] & \text{if } P_m < u \in U(0,1) \\ g_j & \text{otherwise} \end{cases} \qquad \text{with } j = 1, 2, ..., N_x \tag{1.14}$$

The quantity $\mu[\cdot]$ in above Eqn. 1.14 is a bit-inversion operator that represents the bit flipped from '0' to '1' and vice versa. The binary mutation operation is illustrated in Fig. 1.4. In real-encoded algorithms, mutation is generally performed using a perturbation technique similar to that described for crossover except that the perturbation amount is rather small.

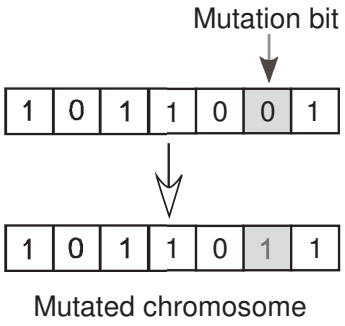


Figure 1.4: Mutation operation in binary representation

1.2.7 Selection Methods

The process of natural section is simulated to achieve a selection mechanism in the GAs. It essentially defines how the algorithms update the population from one generation to the next. In general, chromosomes $\mathbf{x}$ are selected from the population based on the requirements imposed on the solutions in terms of objective functions $\mathbf{F}^t$ (Eqn. 1.9) in order to create a new population on the principle of the "survival of the fittest". This can be achieved by using many different selection methods but the

most commonly used methods are *roulette-wheel*, *tournament*, *ranking*, and *steady-state* selection.

In *roulette-wheel selection* each individual's probability of being selected in the next population is proportional to its fitness value. The probability of survival $P_s(k)$ of a chromosome $\mathbf{x}^t(k)$ is calculated by using the normalized fitness value

$$P_s(k) \;=\; \frac{f^t(k)}{\sum_{k=1}^{N_p} f^t(k)} \tag{1.15}$$

with

$$\sum_{k=1}^{N_p} P_s(k) \;=\; 1$$

Since the probability of selection is based on the fitness proportion in the population, this method is also referred to as *proportionate selection* method.

Rank selection involves ranking the individuals from 'best' to 'worst' on the basis of their measured fitness values. The fitness rank is used to determine the probability of survival P_s. New fitness values that are inversely related to their ranking are then assigned to the individuals.

In *tournament selection*, a group of individuals are chosen iteratively from $\mathbf{P}^t$ by holding a tournament and the one with the best fitness value is chosen for $\mathbf{P}^{t+1}$ until it is filled with a predetermined number of individuals. The tournament size is typically set to a pair of individuals but it can be up to five.

Although the three selection methods described above exert certain *selection pressure* to drive an algorithm to convergence, there is always a risk that the best individual does not get selected and is subsequently lost. This can be avoided by ensuring that a number of individuals deemed to be the best are always passed on to the next generation unchanged. This method is called *elitism* and it often increases the convergence speed at the expense of a risk of getting stuck around the so-called *elite* solutions. However, a mechanism can be implemented to isolate the elite solutions from influencing the selection procedure.

The *steady-state selection* method employs a deterministic section procedure. In this method, most of the individuals survive and only a fraction of the population is updated in every generation. A fixed number, say, N_s, of new individuals are created and added to the population of N_p. Then the $N_p + N_s$ chromosomes are sorted according to their fitness values, the least fit N_s chromosomes are discarded and the rest survive to a new generation.

The configurations of GAs are very problem specific. The success of any genetic algorithm largely depends on how well it has been customized for a given application. The customization can be done by choosing proper objective function(s), chromosome encoding scheme, genetic operators, and section methods. Beside these parameters, there are other parameters and conditions that also affect the performance of a GA. Population size N_p, crossover and mutation probabilities P_x and P_m, respectively, and termination criteria play significant role in a GA's convergence. In spite of many attempts to find the optimal parameter values, systematic trials for specific problems remain the most accepted norm in configuring a GA.

The basic concept and configuration of GAs in general have been introduced in this section. However, GAs do not necessarily follow any strict configuration or specific guidelines. As a result, the number of GA variants with different configurations that exit today is overwhelming. This also brings enormous possibilities in the optimization domain whereby problems which were not within the scope of any existing method can now be solved. With the increasing computing power offered by advancements in integrated circuit technology, the simulation of evolutionary systems is becoming more and more tractable and GAs are being applied to many real world problems including the design of digital filters. The earliest use of GAs to any kind of filter design dates back to the eighties when it was applied for the design of adaptive IIR filters [25]. In later years, researchers have applied this powerful algorithm for the design of various types of FIR and IIR filters including cascade, parallel, and

fixed-point filters [26]- [27].

1.3 Contributions and Review of Related Work

Motivated by the inherent flexibility offered and the recent advancements in GA-based optimization methods, we propose in this dissertation new GA-based methods for the design of several types of digital filters as described in this section. We explore various GA configurations such as binary and real-encoded GAs, single and multiobjective GAs, as well as a hybrid GA approach. In each case, the coefficients of the filter are treated as chromosomes which are optimized by the GA to obtain a filter that would satisfy prescribed specifications.

1.3.1 Fractional-Delay Filters

Fractional-delay digital filters with a tunable delay are often needed to compensate for fractional delays introduced in many applications such as speech coding and synthesis, sampling-rate conversion, time-delay estimation, and analog-to-digital conversion [28]. In general, it is desirable that the fractional delay (FD) be tunable on line without redesigning the filter and the structure used should be suitable for real-time applications. An FD FIR filter of this type can be designed by using a parallel structure first proposed by Farrow in [29].

Fractional-delay filters based on the Farrow structure (FS), referred to hereafter as FDFS filters, are commonly designed by using least-squares (LS) [28] or weighted LS techniques [30]. Linear-programming methods have also been used to obtain optimal minimax solutions for such filters [31]. However, like the design problems associated with many types of digital filters, that of FDFS filters is a nonlinear optimization problem. Furthermore, the difficulty of the optimization task is compounded by the multimodal nature of the optimization problem.

Motivated by the fact that GAs can discard inferior suboptimal solutions in favour of better subsequent solutions, we have developed a GA-based optimization approach for the design of FDFS filters. An FS comprising a number of parallel subfilters of the same length is optimized to approximate a fractional delay that is tunable over a desired frequency range. The usual symmetry condition imposed on the filter coefficients for strict phase linearity [1] is removed and the values of the coefficients are optimized with a GA so as to achieve an approximately linear phase response with respect to a prescribed passband. The algorithm developed entails a concurrent optimization approach for all subfilter coefficients instead of a sequential approach, which leads to improved efficiency.

An attractive alternative to the FIR FS is the allpass IIR FS (AIFS) [28]. There are three advantages in using an allpass IIR instead of an FIR FS as follows: (1) the amplitude response is unity in the entire baseband, (2) the overall delay for the same maximum delay error is considerably smaller [1], and (3) the number of multipliers, adders, and unit delays required to implement the FS is significantly smaller.

In the past several years a number of methods have been proposed for the design of allpass IIR FD filters. An allpass IIR FD filter with a specific fractional delay can be designed by using a closed-form formula introduced by Thiran [28]. Several methods have been proposed for the design of allpass IIR-based FD filters with tunable delays which include an analytic method reported in [32], optimization-based methods such as least-squares (LS) [28], weighted LS [33], and minimax methods [34]. In these methods, the filter coefficients are expressed in terms of polynomials of the FD control parameter. A design method for AIFS similar to the LS design of FIR FSs was suggested in [28]. However, like the design problems of FDFS filters described in the previous paragraphs, problems for the design of AIFS filters are nonlinear multimodal optimization problems. Therefore, we have proposed a GA-based approach for the design of AIFS filters, which is similar to that for FDFS filters.

In the design of both FDFS and AIFS, the filter coefficients are encoded as binary strings and, as a consequence, a quantization-error-free hardware implementation is assured. The algorithm developed entails a concurrent optimization approach for all subfilter coefficients instead of a sequential approach, which leads to improved efficiency. Experimental results show that the GA-based approach leads to reduced maximum amplitude-response and/or phase-delay errors for a specified range of fractional delays relative to those achieved by using a least-squares approach.

1.3.2 IIR Group Delay Equalizers

Linear-phase filters are usually designed as nonrecursive (FIR) filters which can have constant group delay over the entire baseband. However, when highly selective filters are required, a very high filter order is needed which makes these filters uneconomical or impractical. To eliminate this problem, attempts have been made to develop methods to design recursive (IIR) filters whose delay characteristics approximate a constant value in the passband. These include IIR filter design approach that can satisfy both magnitude and phase characteristics simultaneously [35–39]. The design of IIR filters with constant group delay in the passband is also carried out by using allpass structures through evaluation of phase response instead of approximating the group delay directly [40–43]. Some other methods used an indirect approach based on model reduction techniques where a linear-phase FIR filter that meets the required specifications is first designed and then a lower order IIR filter is obtained that meets the original amplitude specifications while maintaining a linear-phase response in the passband [44–46].

In the past few years, a great deal of attention has been paid to a two-step approach whereby a recursive filter is first designed to meet the amplitude response specifications and a delay equalizer is then constructed to equalize the group delay of

the recursive filter [1], [47], [48]. A delay equalizer is an allpass filter which is designed by selecting its coefficients such that the overall group delay of the filter in cascade with the equalizer is flat to within a prescribed tolerance over the passband.

Usually, the equalizer is designed through the use of classical gradient-based optimization methods [1], [49–51]. Quasi-Newton methods are generally fast and efficient, and have been found to work reasonably well for the design of equalizers [1]. However, the stability of the equalizers obtained is not guaranteed and the quality of the solutions obtained depends heavily on the initial points used. Consequently, several designs using different starting points might be required to obtain a stable design [1]. The problem is compounded by the highly nonlinear and multimodal nature of the objective function. Motivated by the fact that GAs can overcome the constraints as described above, we have proposed a genetic algorithm for the design of recursive delay equalizers.

In the proposed approach, the equalizer coefficients are optimized using an objective function based on the passband filter-equalizer group delay. The required equalizer is built by adding new second-order sections until the desired accuracy in terms of the flatness of the group delay with respect to the passband is achieved. Experimental results show that the GA-based approach can achieve stable delay equalizers that would satisfy arbitrary prescribed specifications pertaining to the flatness of the group delay.

1.3.3 Multiplierless FIR Filters in Cascade Form

When digital filters are implemented on a computer or in terms of special purpose hardware, each filter coefficient is stored in a register of finite length and arithmetic operations are usually carried out by using adders and multipliers. If the coefficients can be expressed in terms of *sums of powers of two* (SOPOT), multiplications can

be carried out by simply using adders and data shifters and in this way a so-called *multiplierless* hardware implementation can be achieved. Multiplierless systems are very effective in terms of chip area, propagation delay, and power dissipation compared to systems that use general multipliers [52].

The design of discrete-coefficient filters has been a topic of special interest for the past three decades [53]- [54]. The simplest and most widely used solution to the problem is to round or truncate the coefficients to a fixed-bit representation. Simple designs using signed powers of two (POT) and canonical signed-digit number representations have been reported in [53], [55]. However, the designs so obtained are not optimal. Consequently, several methods have been developed for optimizing the frequency response of digital filters subject to discrete constraints imposed on the coefficient values. These include the use of mixed-integer linear programming (MILP) [56], [57], weighted least-squares methods [58], and local-search techniques [53], [57]. MILP has the advantage that it yields an optimum design but, unfortunately, the amount of computation increases exponentially with the filter length and, consequently, the method can be used only for small filter lengths less than 40 [58]. Some of the approaches start with a given optimal filter solution and find finite word-length solutions in the neighborhood of an optimal solution that reduce the implementation cost. Such schemes, although very simple, cannot be guaranteed to satisfy the desired frequency-response specifications because the frequency response of the filter is affected by the coefficient quantization.

GAs have been suggested for the design of discrete-coefficient FIR filters and some results have been reported in [54], [59]. In order to explore further the potential of GAs, we have developed a method based on a recently introduced robust form of GA known as the *orthogonal genetic algorithm* (OGA) [60], [61]. In this method, the crossover operation generates a few but representative samples of potential offsprings scattered uniformly over the feasible solution space. This enables the algorithm to

scan that space once to locate good offsprings for further exploration in subsequent generations.

The cascade realization of FIR filters offers several advantages when the goal is a fixed-point implementation. If an FIR filter is realized in the form of a single direct structure, the quantization of one coefficient affects all of the filter's zeros. In contrast, if a cascade structure is used, the quantization of coefficients in one of the cascade sections affects only the zeros of that section. Experimentation with discrete-coefficient FIR filters reported in [62] has shown that a smaller error can be achieved by cascading two FIR subfilters. Moreover, splitting a filter into two or more cascade sections simplifies the optimization task.

In this dissertation, an OGA approach is applied to a cascade FIR realization, where each coefficient is represented by an SOPOT. The values of the coefficients are optimized with the OGA so as to achieve prescribed amplitude response specifications. The algorithm developed entails a sequential optimization approach for the two direct-form cascaded subfilters, which leads to an improved amplitude response. Experimental results show that the OGA approach leads to improved amplitude response relative to that of an equivalent direct-form cascade filter obtained using the Remez exchange algorithm.

1.3.4 Asymmetric FIR Filters

FIR filters are usually designed with symmetric coefficients to achieve linear phase response with respect to the baseband. However, symmetric coefficients also result in a large group delay. The group delay can be reduced by removing the coefficient-symmetry condition and by using optimization, approximately linear-phase response with respect to the passband(s) as well as a specified amplitude response with respect to the baseband can be achieved [63]. Filters so designed would have nonlinear

phase in the stopband(s) which would lead to phase distortion but phase distortion in stopbands is of no concern in practice.

In the past several years, a number of optimization methods for the design of FIR filters with predefined amplitude and phase responses have been proposed [63]-[64]. Most of these methods are based on a single error criterion for all frequency bands, which may involve the L_∞ or L_2 norm. However, the exclusive use of one of these error criteria may not produce a truly optimum design for the application at hand [65].

Since minimization of the maximum amplitude distortion is important for signals to be passed, the L_∞ norm is appropriate for the passband. Furthermore, the use the L_∞ norm tends to yield a minimax solution whereby the optimization error tends to be uniformly distributed with respect to the frequency range of interest [1]. In many applications, especially where narrow-band filters are required, minimization of both the gain and total energy in the stopband is important. In such applications, an error measure based on the L_2 norm with a constraint imposed on the maximum amplitude-response error is more suitable for the stopbands [65]. In certain applications, a group-delay error measure should also be included in order to achieve a flat group delay characteristic with respect to the passband(s). In effect, a design problem of this type entails three criteria requiring simultaneous optimization of three objective functions with different individual optima. With such multiobjective formulation, there is generally no single best design that is optimum with respect to all the objective functions. A solution of such a problem can be achieved by using a multiobjective GA that would make all possible tradeoffs among competing objectives through evolution.

In this dissertation, an approach based on an *elitist nondominated sorting genetic algorithm* (ENSGA) is proposed to find so-called *Pareto-optimal solutions* for FIR filters designed to have a predefined amplitude response and a flat group-delay characteristic [66]. Three individual objective functions based on the passband and

stopband amplitude-response errors and a measure for the flatness of the group-delay characteristic with respect to passband are used, and a limit is imposed as a constraint on the maximum group delay. Experimental results show that the ENSGA leads to improved amplitude response as well as delay characteristic relative to those achieved by using a state-of-the-art weighted least-squares approach.

1.3.5 Hybrid Design Approach for IIR Filters

The many advancements in the area of numerical optimization in the past several decades in conjunction with the ever-increasing power of computers have made optimization-based IIR (recursive) filter design an increasingly important field of research [67]. This design problem has been tackled using a great variety of optimization methods such as least-pth [1], least-squares (LS) [68]- [38], weighted LS [69] and linear programming methods [70]. Genetic algorithm and genetic programming have also been used to obtain optimal solutions for such filters [71]- [72]. These methods offer a framework in which a variety of design criteria and specifications can be readily incorporated.

It is well-known that gradient-based optimization algorithms such as the steepest-descent and quasi-Newton (QN) algorithms can be used effectively for the design of IIR filters [1], [67]. However, the solutions obtained depend on the initialization used and many attempts may be required to obtain a satisfactory solution. GAs offer an advantage in this respect in that they can accumulate information about an unknown problem and then use this information to find promising regions of the parameter space. However, the performance of GAs is often compromised by their very slow convergence and lack of precision because they do not always utilize local information effectively [73]. By coupling gradient-based with search-based algorithms such as GAs, their individual advantages can be brought together and their individual

limitations can be avoided.

The prospects of combining the flexibility and reliability inherent in the GA with the fast convergence and precision of the QN algorithm have motivated us to propose a hybrid genetic algorithm formulated by using a GA along with a QN algorithm to simplify the design of IIR digital filters. The proposed algorithm involves a decimal encoding scheme. Starting with a randomly created initial population of chromosomes, the algorithm minimizes an L_2-norm objective function based on the amplitude-response error. Experimental results have shown that the proposed hybrid algorithm can consistently achieve IIR filters that would satisfy arbitrary prescribed specifications.

1.4 Organization of Dissertation

The remainder of this dissertation is organized as follows. In Chapter 2, two optimization approaches for the design of fractional delay filters based on a GA are presented. The first approach exploits the advantages of a global search technique to determine the coefficients of an FD FIR filter based on the Farrow structure. In the second approach, the FD filter was designed by using the allpass IIR based Farrow structure. Chapter 3 details a GA-based optimization approach for the design of delay equalizers. In Chapter 4, we propose an optimization approach for the design of multiplierless FIR filters to exploit a recently-introduced GA, called orthogonal GA, based on the so-called experimental design technique. In this approach, the filters are constructed as a cascade of two subfilters to reduce the quantization effect in the fixed-point implementation. Chapter 5 proposes a specially tailored ENSGA which involves a multiobjective error formulation based on the amplitude response and passband group delay for the design of asymmetric FIR filters. A hybrid approach for the design of IIR filters using a GA along with a quasi-Newton algorithm is presented

in Chapter 6. Finally, Chapter 7 summarizes the main results of this dissertation, and suggests directions for future research.

Chapter 2

Design of Tunable Fractional-Delay Filters

2.1 Introduction

In this chapter, GA-based methods for the design of FIR and allpass-IIR fractional-delay (FD) filters are described. Both FIR and allpass-IIR FD filters are based on the so-called Farrow structure (FS) and we refer to these structures as the FDFS and AIFS, respectively. In the first approach, an FS comprising a number of parallel subfilters of the same length is optimized to approximate a fractional delay that is tunable over a desired frequency range. The usual symmetry condition imposed on the filter coefficients for strict phase linearity [1] is removed and the values of the coefficients are optimized with a GA so as to achieve an approximately linear phase response with respect to a prescribed passband. The proposed approach involves a multiobjective error formulation based on the amplitude response and phase delay. In the second approach, a similar genetic algorithm is used for the design of AIFS filters exploiting the advantages offered by the allpass IIR filter structure over the FIR structure. As in the GA for FDFS, an FS comprising a number of parallel IIR allpass subfilters of the same order is optimized to achieve a fractional delay that is tunable over a desired frequency range. Chromosomes are constructed in matrix form

with each column representing the coefficients of each of the allpass subfilters in the FS and the optimization is carried out by minimizing an objective function based on the phase delay error. A stability constraint is also incorporated in the design to avoid an unstable solution.

In both of the proposed design techniques, the filter coefficients are encoded as binary strings and, as a consequence, a quantization-error-free hardware implementation is assured. The algorithm developed entails a concurrent optimization approach for all subfilter coefficients instead of a sequential approach, which leads to improved efficiency.

The chapter is organized as follows. Section 2.2 introduces the notion of ideal fractional delay. The design problem of FDFS filters, the details regarding the methodology of the proposed GA, and related design examples are presented in Section 2.3. The design of AIFS filters is considered in Section 2.4.

2.2 Ideal Fractional Delay

The delayed version of a discrete-time signal $x(nT)$ may be represented as

$$y[nT] \;=\; x[(n - D_0)T] \tag{2.1}$$

where T is the sampling period and D_0 is a positive integer that denotes the amount of time by which the signal is delayed. If the desired continuous-time delay is τ, in typical applications the value of D_0 can take only integer values and may be obtained by rounding the ratio τ/T to the nearest integer. A fractional delay may arise from such rounding, which would need to be corrected in certain applications by using an FD filter. In such a case, the delay can be expressed as

$$D_0 \;=\; D \;+\; \mu \quad \text{where } D = \text{int}(D_0), \quad -0.5 \leq \mu \leq\; 0.5$$

and μ is the fractional delay.

An ideal fractional-delay filter has a frequency response

$$H_{id}(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})} = e^{-j\omega D_0} \tag{2.2}$$

where $H_{id}(z)$ is the transfer function of the filter. The corresponding impulse response, $h_{id}(n)$, is a delayed *sinc function* that can be obtained by taking the inverse Fourier transform of the frequency response [74]. Assuming a sampling period T of 1 s, the impulse response is obtained as

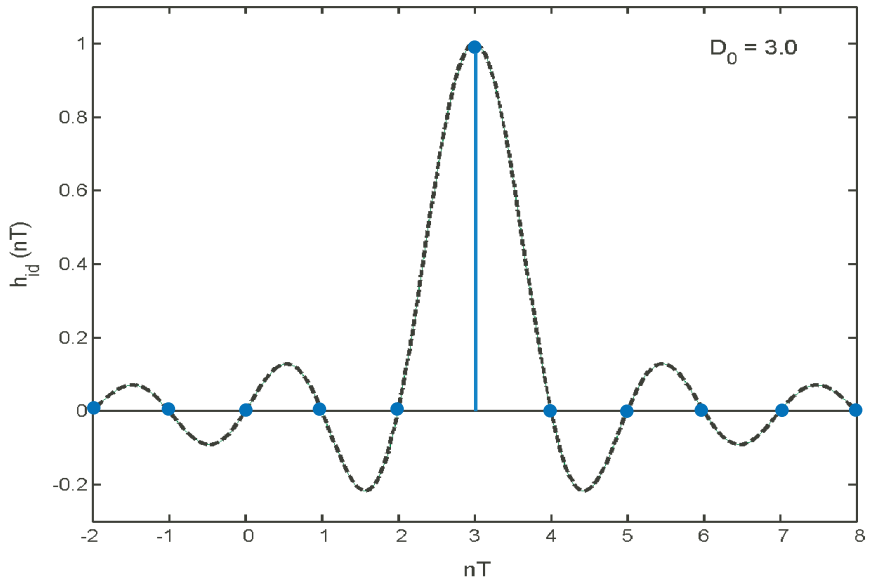
$$h_{id}(n) \;=\; \mathrm{sinc}(n - D_0) \;=\; \frac{\sin[\pi(n - D_0)]}{[\pi(n - D_0)]}, \quad -\infty < n < \infty \tag{2.3}$$

According to Shannon's sampling theorem, a sinc interpolator can be used to exactly evaluate a signal value at any point in time as long as it is sampled at a rate higher than twice the maximum signal frequency. The sample of a discrete-time signal $y(n)$ at any arbitrary continuous time D_0 can be obtained by convolving the signal with $\mathrm{sinc}(n - D_0)$ according to the equation

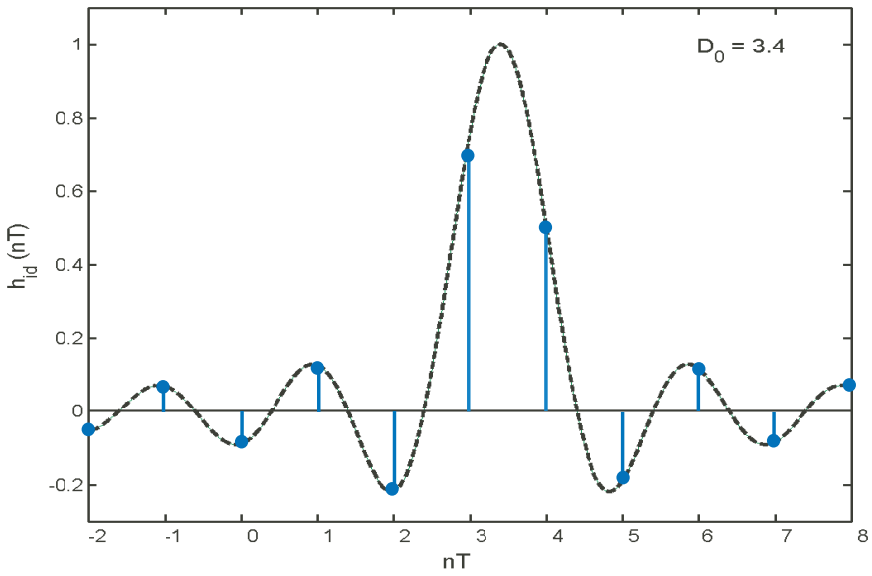
$$y(D_0) \;=\; \sum_{n=-\infty}^{\infty} y(n)\ \mathrm{sinc}(n - D_0) \tag{2.4}$$

Fig. 2.1 shows the ideal impulse response when $D_0 = 3.0$ and $D_0 = 3.4$. In the former case, $h_{id}(n)$ is zero at all n except $n = D_0 = 3.0$. In the latter case, the impulse response sequence has nonzero values for $-\infty < n < \infty$ although it diminishes quickly and approaches zero as n approaches $\pm\infty$. As a consequence, it represents a *noncausal* filter which cannot be made causal by applying a finite shift in time domain. Furthermore, the filter is unstable since the impulse response is not absolutely summable [75]. An ideal FD filter is thus *nonrealizable*.

The problem of delaying a signal by a fractional delay corresponds to the problem of interpolating a signal at arbitrary (noninteger) sampling points between the discrete-time input samples rather than simply delaying the signal. Besides, the infinitely-long ideal impulse response can only be approximated with a filter of



(a)



(b)

Figure 2.1: Impulse response of the ideal fractional delay filter with delay (a) $D_0 = 3.0$ (b) $D_0 = 3.4$ samples.

finite length where the approximation can be carried out for desired amplitude and phase responses with respect to the *bandwidth of interest*. A filter can be designed to approximate the ideal impulse response such that a specific delay is obtained. If a different delay is required the filter will have to be redesigned.

In some applications, the fractional delay is required to be tunable on-line without redesigning the filter. The rest of this chapter is concerned with the design of such filters.

2.3 Tunable Fractional Delay FIR Filter Design

2.3.1 The FDFS Filter

Ideally, an FD filter is required to have a constant amplitude response of unity and a phase response that is linear with respect to some prescribed passband say, $0 \leq \omega \leq \omega_p$, where ω_p is the passband edge. Furthermore, the fractional delay realized should be adjustable without changing the filter coefficients. An FDFS filter is based on a parallel connection of $P+1$ FIR subfilters, each of length N, as depicted in Fig. 2.2. Straightforward analysis gives the transfer function of the structure as

$$H(z,\mu) = \sum_{k=0}^{P} \mu^k B_k(z) \tag{2.5}$$

where

$$B_k(z) = \sum_{n=0}^{N-1} b_{kn}\ z^{-n} \tag{2.6}$$

Hence

$$\begin{aligned} H(z,\mu) &= \sum_{k=0}^{P} \mu^k \sum_{n=0}^{N-1} b_{kn}\ z^{-n} \\ &= \sum_{n=0}^{N-1} \left(\sum_{k=0}^{P} \mu^k\ b_{kn} \right) z^{-n} \end{aligned}$$

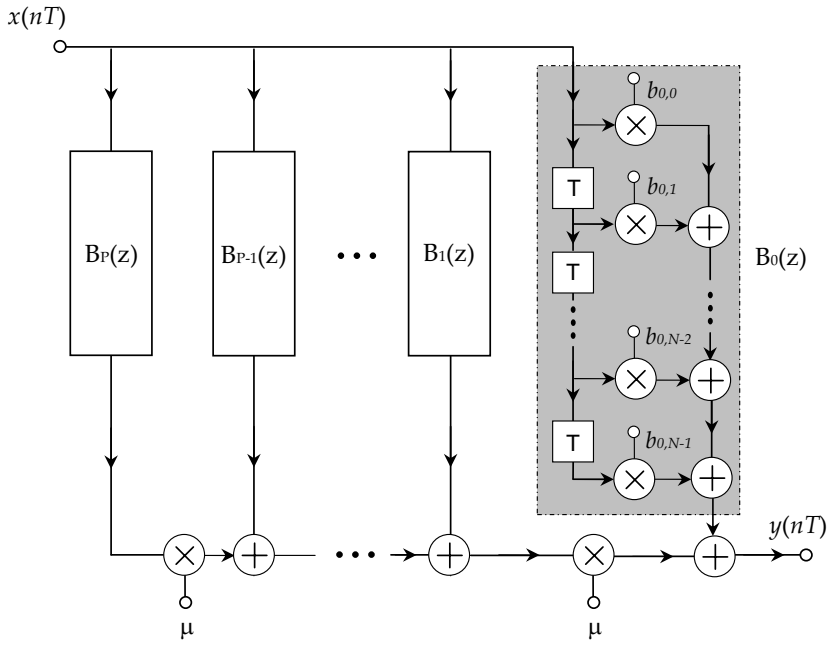


Figure 2.2: The Farrow structure implementation of FD FIR filters.

or

$$H(z,\mu) \;=\; \sum_{n=0}^{N-1} h_\mu(n) z^{-n} \tag{2.7}$$

where

$$h_\mu(n) \;=\; \sum_{k=0}^{P} b_{kn}\; \mu^k$$

An FDFS filter can be designed by optimizing coefficients b_{kn} such that the frequency response of the filter, $H(e^{j\omega},\mu)$, approaches the desired frequency response

$$H_d(e^{j\omega},\mu) \;=\; e^{-j\omega(D+\mu)} \quad \text{for } 0 \le \omega \le \omega_p \tag{2.8}$$

to within some degree of precision, where

$$D \;=\; \frac{N-1}{2} \quad \text{and} \quad \mu \;\in\; [-\hat{\mu},\hat{\mu}] \tag{2.9}$$

are a fixed delay and the required fractional delay, respectively, and $\hat{\mu}$ is a fraction in the range in the range 0 to 1. This problem has been solved in the past by minimizing an objective function based on the L_2 norm using an LS approach [28].

In the LS method, an FDFS filter is designed in two steps. In the first step, M prototype FIR filters of the same length N approximating the desired frequency response are designed assuming uniformly distributed FD values μ in a range $0 \leq \mu \leq 0.5$. In the second step, the coefficients for the FS are deduced from the coefficients of the prototype filters through LS optimization.

A prototype filter can be designed by solving a quadratic equation that yields the unique minimum-error solution [28]. The coefficient vector of the prototype filter obtained is given by

$$\mathbf{c}_\mu = \mathbf{P}^{-1}\mathbf{q} \tag{2.10}$$

where $\mathbf{P}$ is a Toeplitz matrix with elements

$$p_{kl} = \frac{1}{\pi}\int_0^{\omega_p} \cos[(k-l)\omega]\, d\omega \quad \text{for } k, l = 1, ..., N-1 \tag{2.11}$$

and $\mathbf{q}$ is a column vector with elements

$$q_k = \frac{1}{\pi}\int_0^{\omega_p} \cos[(k-(D+\mu)\omega]\, d\omega \quad \text{for } k = 1, ..., N-1 \tag{2.12}$$

Once the prototype filters are designed, the coefficients b_{kn} for the FS are obtained by solving the system

$$c_{\mu m}(n) = \sum_{k=0}^{P} b_{kn}\ \mu_m^k \quad \text{for } m = 0, ..., M-1 \ \text{ and } n = 0, ..., N-1$$

using the least-squares curve fitting technique.

In the present method, an objective function based on the L_∞ norm is formulated whose minimization yields a minimax solution. An important merit of minimax solutions is that the optimization error tends to become uniform with respect to the

frequency range of interest as the solution is approached [1]. From Eqns. 2.7 and 2.8, the peak errors in the amplitude response and phase delay can be expressed as

$$\delta_a = \max \left| 1 - |H(e^{j\omega}, \mu)| \right| \quad \text{for } 0 \leq \omega \leq \omega_p \text{ and } |\mu| \leq \hat{\mu}$$

and

$$\delta_d = \max \left| \frac{\omega(D+\mu) - \arg H(e^{j\omega}, \mu)}{\omega} \right| \quad \text{for } 0 \leq \omega \leq \omega_p \text{ and } |\mu| \leq \hat{\mu}$$

respectively. In a discretized version of the optimization problem at hand, uniformly distributed values of μ are chosen in the range $-\hat{\mu} \leq \mu \leq \hat{\mu}$. The approximation errors are computed for each value of μ and the errors are sampled at frequency points ω_k for $k = 1, 2, ..., K$ and $0 \leq \omega_k \leq \omega_p$. Based on these errors, an objective function can be constructed as

$$\Delta = W_a \delta_a + W_d \delta_d \tag{2.13}$$

where W_a and W_d are positive weighting factors used to control the approximation errors in the amplitude response and phase delay, respectively. The coefficients of the FS can be expressed in matrix form as

$$\mathbf{B} = \begin{bmatrix} b_{00} & b_{10} & \cdots & b_{P0} \\ b_{01} & b_{11} & \cdots & b_{P1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{0(N-1)} & b_{1(N-1)} & \cdots & b_{P(N-1)} \end{bmatrix} \tag{2.14}$$

where each column represents the impulse response values of a subfilter.

2.3.2 The GA Approach

In this section, we describe a GA that can be used to design FDFS filters by minimizing the weighted sum Δ in Eqn. 2.13.

Coefficient matrix **B** in Eqn. 2.14 defines the *phenotype* structure of a *chromosome*, i.e., the candidate solution for the optimization process. As in a typical GA, the proposed algorithm applies the genetic operators *crossover* and *mutation* on the *population* of chromosomes and implements an *elitism*-based selection scheme to create new *generations*.

The chromosomes are encoded in binary form to obtain the *genotype* representation which would allow the GA to use its full strength through the classical recombination process. Each of the decimal values in the phenotype is first linearly mapped to an integer defined in a specified range, and the integer is encoded using a fixed number of binary bits for the genotype representation. Limiting the variable mapping to a specified integer range enables the search of the GA to be confined within a feasible region.

The GA adopts a uniform crossover operation, where two complementary offspring chromosomes are generated from two randomly selected parent chromosomes by randomly selecting some parts from Parent X and some parts from Parent Y using a random crossover mask. The crossover mask is constructed by assigning a 1 or 0 to each location on a random basis. If there is a 1 (0) in a given location of the random mask, the chromosome parts from Parents X and Y are inherited by Offsprings A (2) and B (1), respectively. In a mutation operation, the value(s) of particular gene(s) are changed by applying random bit inversions. Each individual generated by the crossover undergoes a mutation. The crossover and mutation operations are illustrated in Fig. 2.3.

In the proposed GA, the frequencies of crossovers and mutations are controlled by probabilities P_x and P_m. In each case, a random number is generated in the range 0 to 1 and if it is less than P_x or P_m, as appropriate, crossover or mutation is applied; otherwise, crossover or mutation is not applied. As in a classical GA, mutation is treated as a supporting operator for the purpose of restoring lost genetic

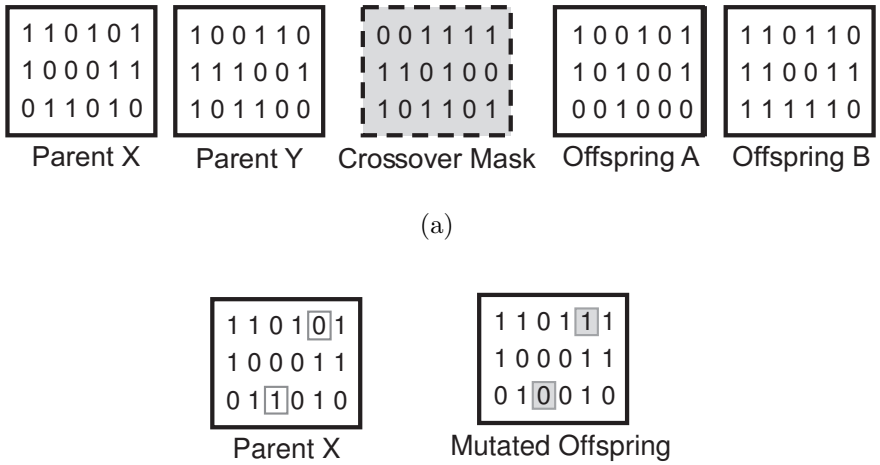


Figure 2.3: (a) Uniform crossover and (b) mutation applied to the chromosomes.

material, for example, if all chromosomes in a population have converged to 0 at a given position and the optimal solution has a 1 at that position. The crossover cannot regenerate a 1 at that position while a mutation can. On the other hand, a higher mutation frequency can cause the algorithm to become random to an undesirable extent; furthermore, mutations are less effective in reducing the objective function. To avoid these problems, the frequency of mutations is kept fairly low by using a low value for P_m. Initially, the values of P_x and P_m are set to 0.75 and 0.05, respectively, and if no improvement in the solution is achieved after a number of generations, say, K, the values of P_x and P_m are increased until specified upper limits, say, $\hat{P}_x$ and $\hat{P}_m$ are reached. The increased frequencies of crossovers and mutations used enable the GA to explore new areas of the parameter space in the event where progress towards a solution is slow.

In the selection process, chromosomes are ranked on the basis of their fitness. The total population that competes for survival comprises the parents as well as the

offsprings created during crossovers and mutations. The algorithm then records a small number of top-ranked solutions as *elite* solutions and selects a fixed number of best-fit chromosomes as potential parents for the next generation.

The best fitness value in a generation is defined as the minimum of all fitness values for a population and can be expressed as

$$\Psi \;=\; \min\left(\Delta_i\right) \quad \text{for } i = 1, ..., N_p$$

where N_p is the size of the population. If the reduction in Ψ in a given generation is less than ε, where ε is a small constant, the generation is deemed to be an 'unproductive generation'.

The algorithm can be initialized by assigning filter coefficients obtained through various standard techniques such as the Remez exchange algorithm or through LS techniques [1]. In the proposed approach, the LS solution discussed in the previous section is used for initialization. The coefficients obtained using the LS method are used as seed in the initial population of the GA approach. Half of the population is generated from neighborhood points of that seed where the neighborhood points are obtained by perturbing the decimal-valued seed by random but small amounts and encoding them in binary form. The remaining population are generated randomly to maintain diversity in the solutions under exploitation. In each successive generation, two-thirds of the solutions are taken from the best-fits of the previous generation, and one-third are generated randomly.

Typically, GAs are stopped when new crossovers and mutations cease to improve the solution, and this situation can best be checked by allowing the GA to run over a prespecified number of unproductive generations. In the proposed GA, we use a two-phase stopping criterion. During the early stages of the algorithm when the current solution point is far from the optimum point, the GA is allowed to continue for L unproductive generations but when a maximum number of prespecified generations,

say, G, is exceeded, the number of sequential unproductive generations allowed is reduced to $L/2$ since improvements of the solution are less likely during the later stages of the algorithm. When the number of unproductive generations or a predefined maximum number of generations, say, $\hat{G}$, is exceeded, the GA stops and the best chromosome is deemed to be the desired solution.

Based on our experience with the proposed GA, suitable values for parameters N_p, $\hat{P}_x$, $\hat{P}_m$, G, $\hat{G}$, K, L, and ε for the problem at hand are 48, 0.9, 0.1, 100, 60, 7, 10, and 10^{-2}, respectively. The algorithm is summarized in a step-by-step form in Table 2.1.

Table 2.1: GA for the Design of FD Filters.

Step 1: Initialize the GA parameters: population size N_p, crossover and mutation probabilities, P_x and P_m, maximum allowable limits for P_x and P_m, $\hat{P}_x$ and $\hat{P}_m$, respectively, and the number of bits for the binary representation of the filter coefficients.
Step 2: a) Set the number of unproductive generations for updating maximum crossover and mutation rates and stopping criterion to K and L.
b) Specify the maximum numbers of generations for first and second phases, G and $\hat{G}$, respectively.
Step 3: Compute the coefficients for the Farrow structure using the LS method in [28] to be used as seed.
Step 4: a) Initialize generation counter, $j = 1$.
b) Set two counters to record the number of successive generations that fail to improve the best-fit solution, $k = 0$ and $l = 0$.
c) Initialize the termination tolerance ε.
Continued on Next Page ...

Step 5: Generate half of the chromosomes in the neighborhood of the seed and the remaining chromosomes randomly.

Step 6: Perform crossover according to the following rules: Randomly select a pair of chromosomes and generate a random number, $R \in [0, 1]$, assuming uniform distribution. If $R \leq P_x$, apply crossover on this pair as in Fig. 2.3.

Step 7: Perform mutation within a chromosome according to the following rules: Generate a random number S as in *Step 6* for each of the bits of a chromosome. Apply mutation on this chromosome by inverting the bits for which the condition $S \leq P_m$ is satisfied.

Step 8: Evaluate the fitness of all chromosomes by using Δ in Eqn. 2.13.

Step 9: Select the $2N/3$ best-fit chromosomes, generate $N/3$ random chromosomes for the next generation, and record the two top-ranked solutions as the elite chromosomes.

Step 10: If $\Psi_{j-1} - \Psi_j < \varepsilon$, set $k = k + 1$, and $l = l + 1$; otherwise set $k, l = 0$.

Step 11: If $k = K$, set $P_x = 1.05Px \ \leq \ \hat{P}_x$, $P_m = 1.1Pm \ \leq \ \hat{P}_m$, and $k = 0$.

Step 12: If $[(j \leq G$ and $l < L)$ or $(j > G$ and $l < L/2)]$ and $j < \hat{G}$, set $j = j + 1$ and go to *Step 6*.

Step 13: Select the best chromosome from the record of elites as the optimized solution, and stop.

2.3.3 Design Examples and Results

The proposed GA was used to design several FDFS filters. The desired specifications of two of these filters are listed in rows 1 to 3 of Table 2.2. The filter coefficients were encoded as 16-bit binary strings with initial crossover and mutation probabilities set

Table 2.2: Results of Design Examples (FDFS Filters).

	Ex. 1	Ex. 2
Normalized bandwidth of interest	0.6	0.8
Number of FIR subfilters	3	3
Length of each subfilter	9	13
Largest p-p amplitude-response error, LS method	3.079×10^{-3}	23.608×10^{-3}
Largest p-p amplitude-response error, GA method	2.824×10^{-3}	21.159×10^{-3}
Largest p-p delay error, LS method	5.372×10^{-3}	23.571×10^{-3}
Largest p-p delay error, GA method	3.977×10^{-3}	16.015×10^{-3}
Total number of generations	100	100

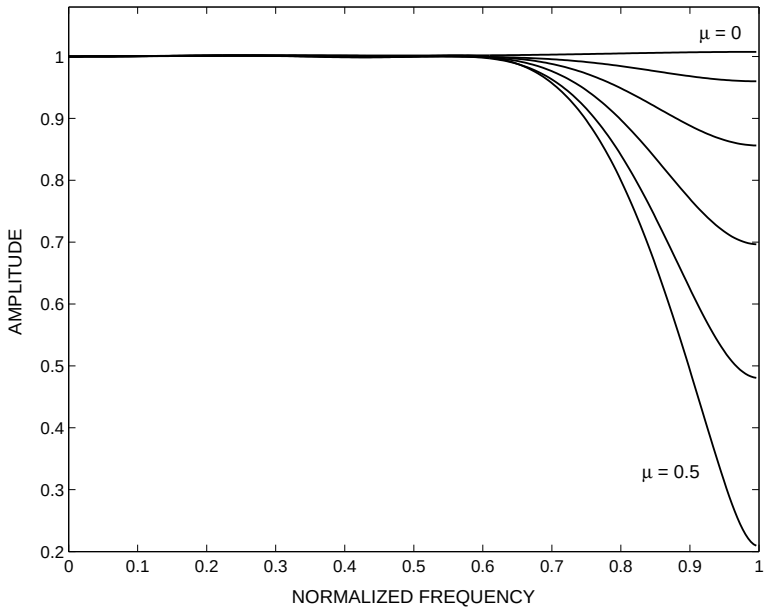
to 0.75 and 0.05, respectively, and the population size was 48. The initial solution used for the GA was obtained by using the LS method in [28] and the resulting Farrow coefficients were encoded into binary values.

The coefficients obtained using the LS method were quantized to the same precision as the binary GA coefficients in order to provide a fair comparison. The amplitude responses and phase delays of the optimized FDFS filters obtained in Examples 1 and 2 are illustrated in Figs. 2.4 and 2.5, respectively. The maximum values of the amplitude response and delay errors for various values of μ for examples 1 and 2 are plotted in Figs. 2.6 and 2.7, respectively. The peak-to-peak values of the amplitude response and delay errors for various values of μ are also presented in Table 2.3. The values of δ_a and δ_d for the GA and LS methods are given in rows 4 to 7 of Table 2.2. The results obtained show that the GA approach leads to improvements over the LS designs by providing lower maximum amplitude and delay errors. Reductions in the maximum amplitude response and delay errors in the ranges of 27.0 - 29.47% and 25.01 - 60.59%, respectively, were achieved. Table 2.2

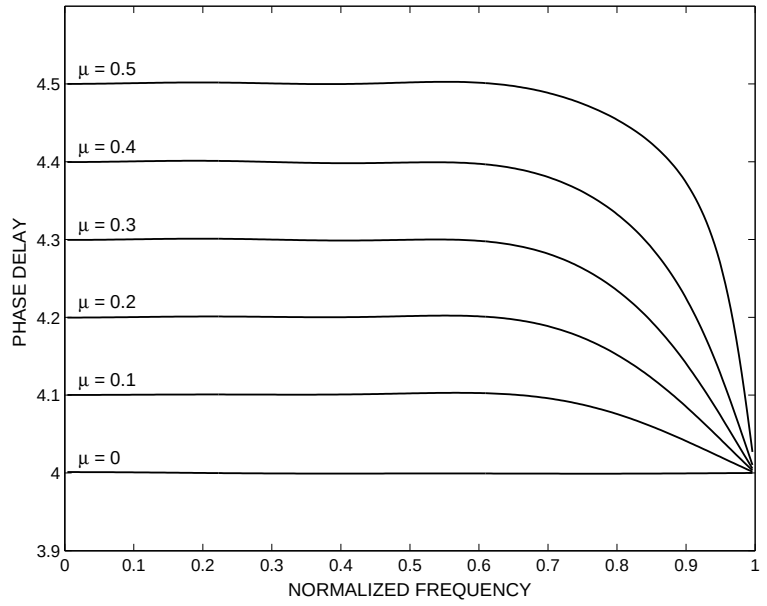
Table 2.3: Peak-to-Peak Errors for Varying Values of μ (FDFS Filters)

	Example 1		Example 2	
μ	LS method	GA method	LS method	GA method
Amplitude-response error				
0	1.556×10^{-3}	1.904×10^{-3}	4.6247×10^{-3}	6.297×10^{-3}
0.1	1.231×10^{-3}	0.753×10^{-3}	4.534×10^{-3}	3.121×10^{-3}
0.2	1.166×10^{-3}	0.736×10^{-3}	7.448×10^{-3}	6.350×10^{-3}
0.3	1.822×10^{-3}	1.314×10^{-3}	13.845×10^{-3}	12.180×10^{-3}
0.4	3.079×10^{-3}	2.667×10^{-3}	20.601×10^{-3}	18.695×10^{-3}
0.5	2.779×10^{-3}	2.824×10^{-3}	23.608×10^{-3}	21.159×10^{-3}
Delay error				
0	1.827×10^{-3}	1.899×10^{-3}	2.680×10^{-3}	3.670×10^{-3}
0.1	3.227×10^{-3}	2.752×10^{-3}	10.811×10^{-3}	7.521×10^{-3}
0.2	3.531×10^{-3}	2.537×10^{-3}	15.184×10^{-3}	8.768×10^{-3}
0.3	3.307×10^{-3}	2.852×10^{-3}	18.418×10^{-3}	13.144×10^{-3}
0.4	4.235×10^{-3}	3.977×10^{-3}	21.471×10^{-3}	16.015×10^{-3}
0.5	5.372×10^{-3}	2.941×10^{-3}	23.571×10^{-3}	10.197×10^{-3}

also gives the CPU time required to obtain the designs of Examples 1 and 2 using the GA and LS approaches. As can be seen, the GA approach requires much more computation than the LS approach as is usually the case with GAs in general. Fig. 2.8 shows the evolution of the objective function through successive generations for the design examples.

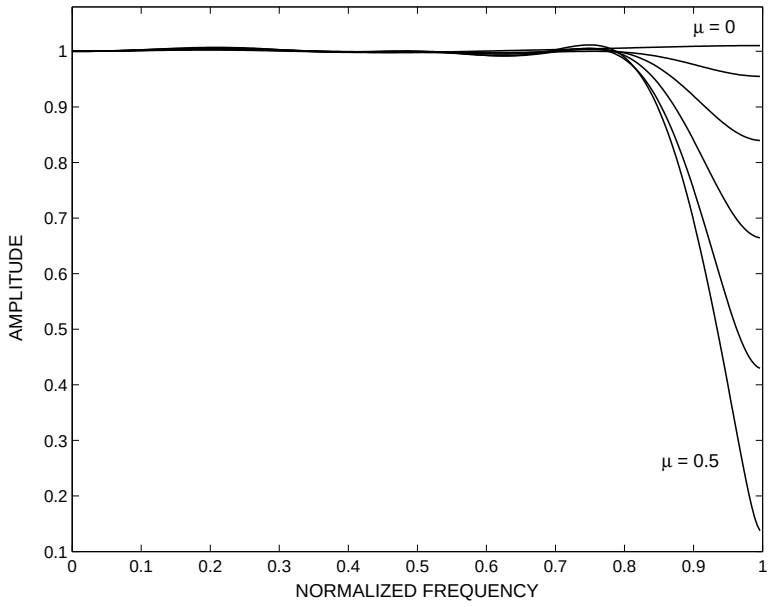


(a)

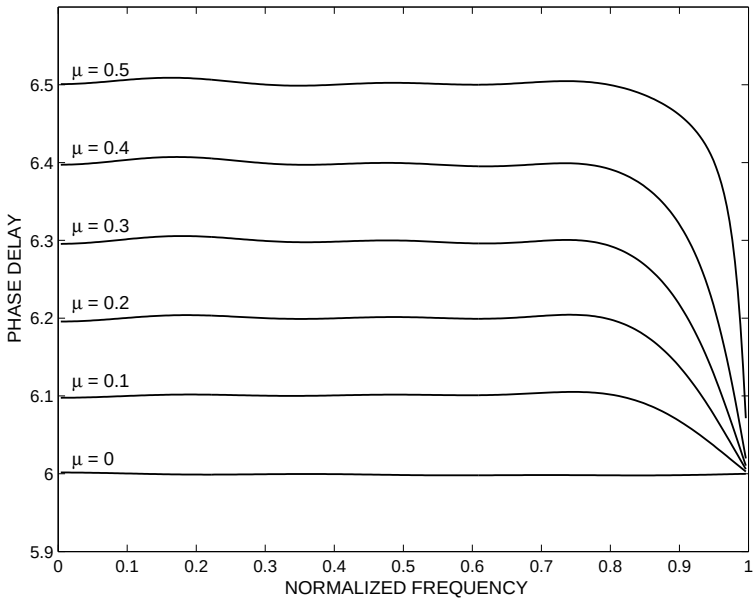


(b)

Figure 2.4: (a) Amplitude response and (b) phase delay of optimized FDFS filter (Example 1).

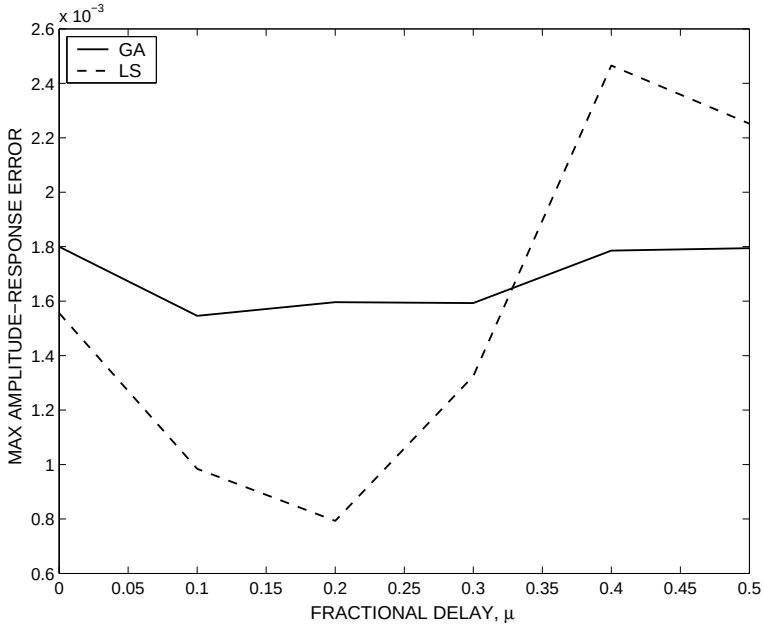


(a)

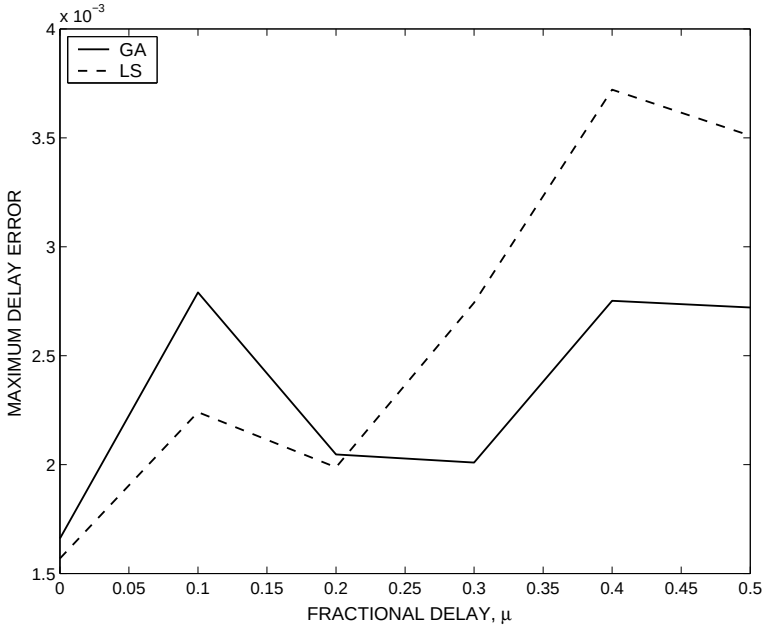


(b)

Figure 2.5: (a) Amplitude response and (b) phase delay of optimized FDFS filter (Example 2).

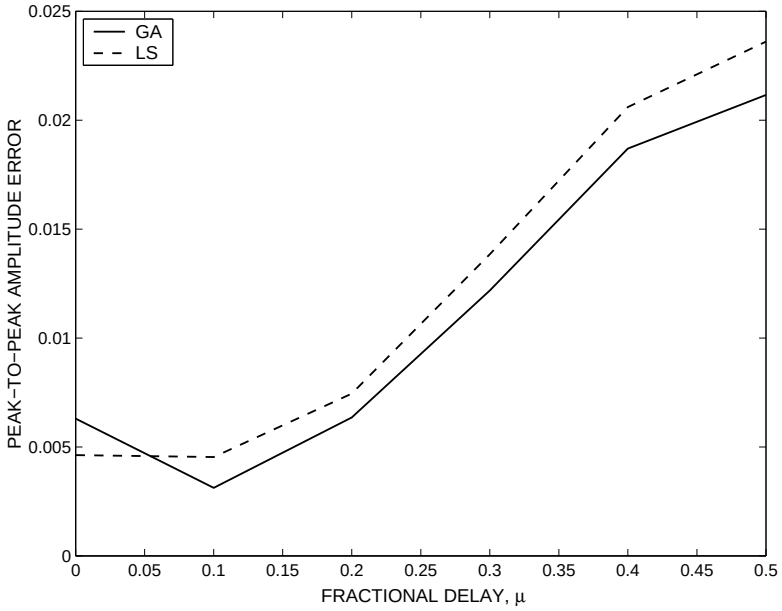


(a)

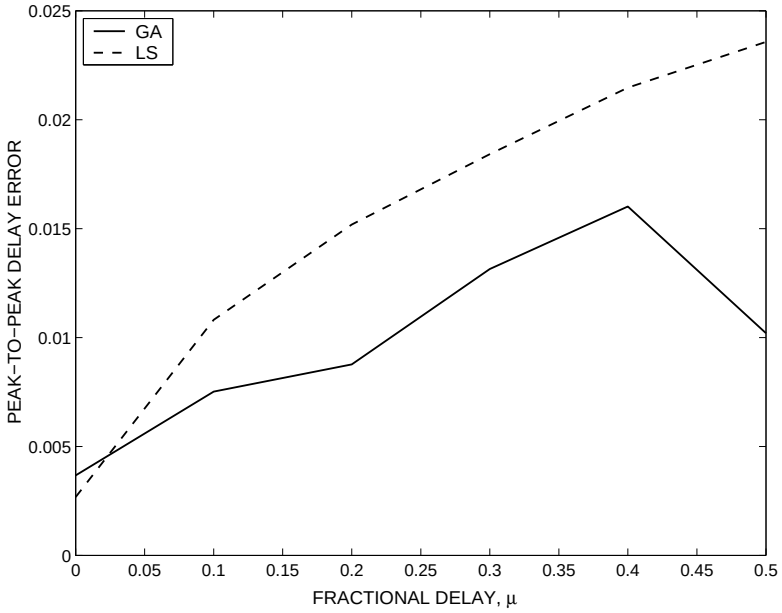


(b)

Figure 2.6: Maximum (a) amplitude-response and (b) delay errors in optimized FDFS filter (Example 1).

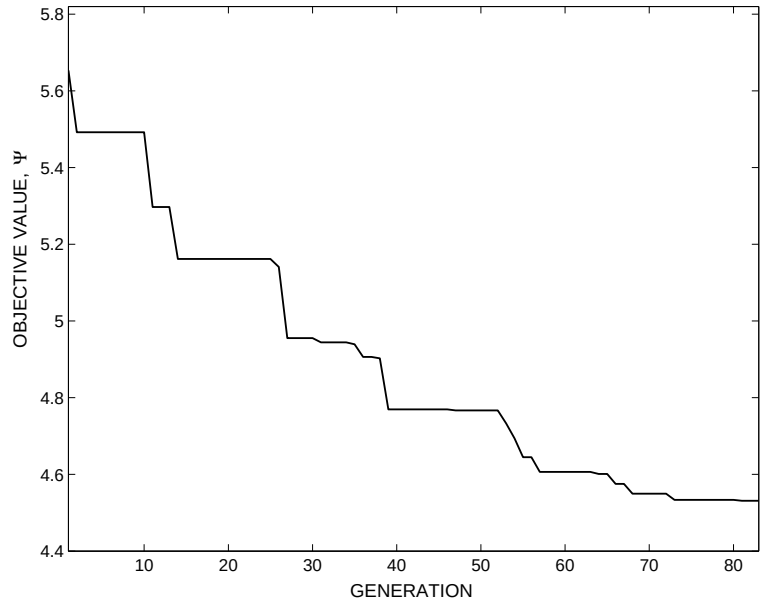


(a)

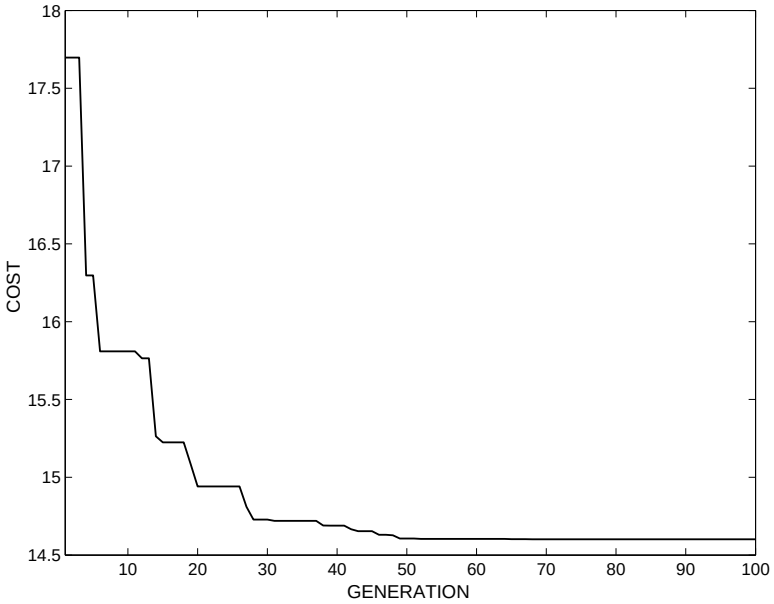


(b)

Figure 2.7: Maximum (a) amplitude-response and (b) delay errors in optimized FDFS filter (Example 2).



(a)



(b)

Figure 2.8: Evolution of objective function through the generations in optimizing FDFS filter (a) Example 1 and (b) Example 2.

2.4 Tunable Fractional Delay IIR Filter Design

2.4.1 The AIFS Filter

An AIFS is based on a parallel connection of $P+1$ allpass IIR subfilters, each of order N. Its transfer function is given by

$$H(z,\mu) \;=\; z^{-N}\frac{A(z^{-1},\mu)}{A(z,\mu)} \tag{2.15}$$

where denominator $A(z,\mu)$ assumes the form

$$A(z,\mu) = 1 \;+\; \sum_{k=0}^{P} \mu^k C_k(z) \tag{2.16}$$

with

$$C_k(z) = \sum_{n=1}^{N} c_{kn}\; z^{-n} \tag{2.17}$$

Hence

$$\begin{aligned} A(z,\mu) \;&=\; 1 \;+ \sum_{k=0}^{P} \mu^k \sum_{n=1}^{N} c_{kn}\; z^{-n} \\ &=\; 1 \;+ \sum_{n=1}^{N} \left(\sum_{k=0}^{P} \mu^k \; c_{kn} \right) z^{-n} \end{aligned}$$

or

$$A(z,\mu) \;=\; 1 \;+\; \sum_{n=1}^{N} a_\mu(n) z^{-n} \tag{2.18}$$

where

$$a_\mu(n) \;= \sum_{k=0}^{P} c_{kn}\; \mu^k \tag{2.19}$$

Due to the mirror symmetric property between numerator and denominator in Eqn. 2.15, the phase delay of the allpass FS can be deduced as

$$\tau(\omega,\mu) \;=\; -N - \frac{2\theta_A(\omega,\mu)}{\omega} \tag{2.20}$$

where $\theta_A(\omega, \mu)$ is the phase angle of the denominator $A(z, \mu)$, which is given by

$$\theta_A(\omega, \mu) \;=\; \arctan\left[\frac{\sum_{n=1}^{N} a_\mu(n) \sin n\omega}{1 + \sum_{n=1}^{N} a_\mu(n) \cos n\omega}\right] \tag{2.21}$$

Fig. 2.9 gives a straightforward realization of the of an AIFS filters. The AIFS filter can be designed by optimizing coefficients c_{kn} such that the phase delay of the filter, $\tau(\omega, \mu)$, approaches the desired phase delay $\tau_d\ (\mu)$ in the frequency range $0 \leq \omega \leq \omega_c$. Assuming an idealized frequency response

$$H_d(e^{j\omega}, \mu) \;=\; e^{-j\omega(D+\mu)} \quad \text{for } 0 \leq \omega \leq \omega_c \tag{2.22}$$

for the filter, where

$$D \;=\; N \quad \text{and} \quad \mu \;\in\; [-\hat{\mu}, 0] \tag{2.23}$$

are a fixed delay and the required fractional delay, respectively, and $\hat{\mu}$ is a fraction in the range in the range 0 to 1, the desired phase delay can be expressed as

$$\tau_d\ (\mu) \;=\; D + \mu \tag{2.24}$$

This problem has been solved in the past by minimizing an objective function based on the L_2 norm using an LS approach [28]. In the proposed method, an objective function based on the L_∞ norm is formulated whose minimization yields a minimax solution. From Eqns. 2.20 and 2.24, the peak error in the phase delay can be expressed as

$$\delta_d \;=\; \max_{0 \leq \omega \leq \omega_c,\ -\hat{\mu} \leq \mu \leq 0} |\tau(\omega, \mu) - \tau_d(\mu)| \tag{2.25}$$

In a discretized version of the optimization problem at hand, uniformly distributed values of μ are chosen in the range $-\hat{\mu} \leq \mu \leq 0$. The approximation error is computed for each value of μ and it is then sampled at frequency points ω_k for $k = 1, ..., K$ and $0 \leq \omega_k \leq \omega_c$. By minimizing δ_d, an allpass IIR FS can be obtained.

To obtain a stable AIFS filter design, a stability constraint must be imposed. A linear constraint on the coefficients $a_\mu(n)$ that would assume stability is based on

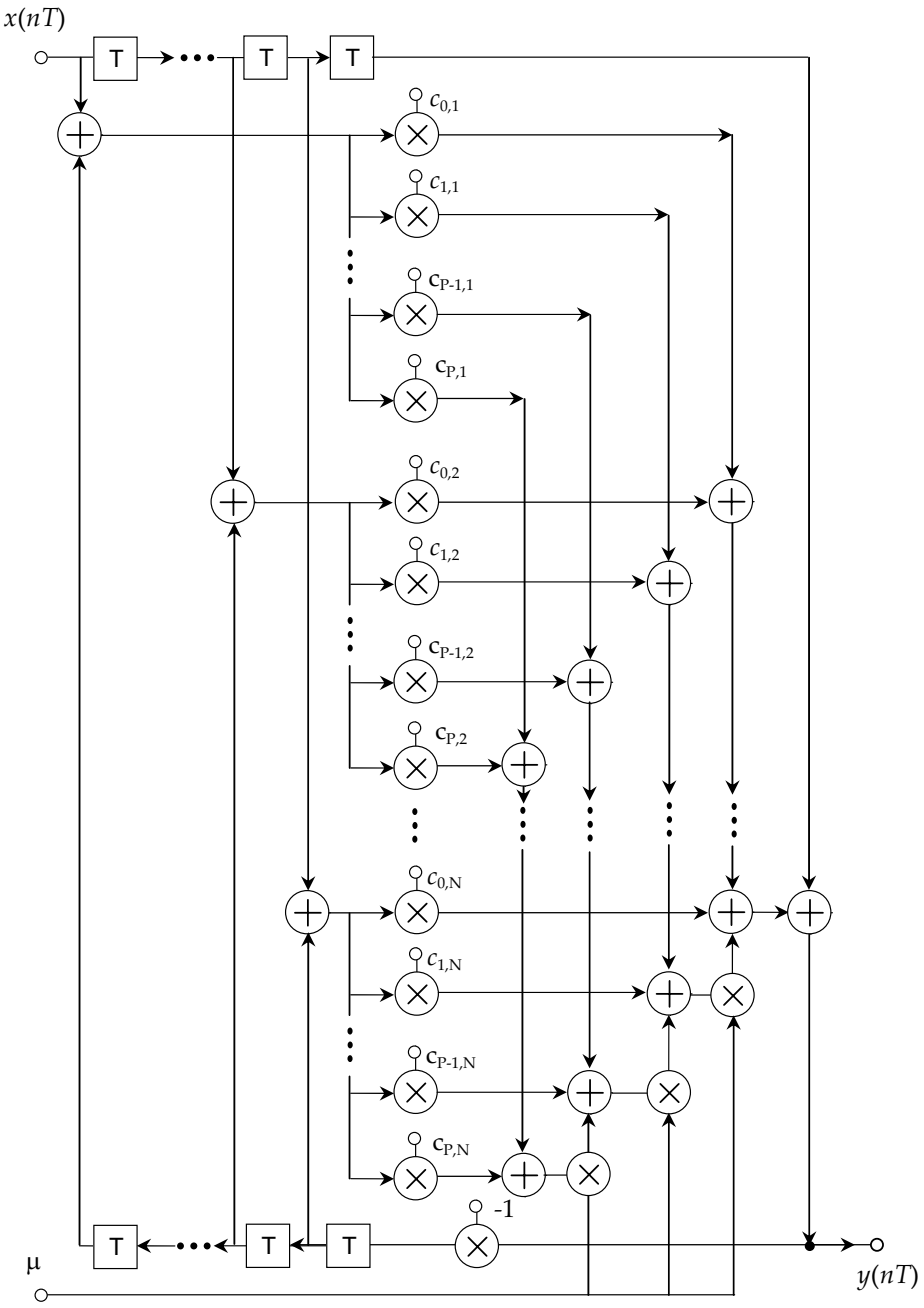


Figure 2.9: A straightforward implementation of AIFS filters.

the requirement that the real part of the denominator polynomial $A(z,\mu)$ must be greater than zero for all z =l [76], i.e.,

$$\Re[A(z,\mu)] \;=\; 1 \;+\; \sum_{n=1}^{N} a_\mu(n) z^{-n} \;\;>\;\; 0 \quad \text{for } |z| = 1 \tag{2.26}$$

or

$$1 \;+\; \sum_{n=1}^{N} a_\mu(n) \cos(n\omega) \;\;>\;\; 0 \quad \text{for } \omega \in [0,\pi] \tag{2.27}$$

2.4.2 The GA Approach for AIFS Design

The design of AIFS filters can be achieved by minimizing the delay error δ_d in Eqn. 2.25 by using a slightly modified version of the algorithm for FDFS filters described in Section 2.3.2.

Coefficient matrix

$$\mathbf{C} \;=\; \begin{bmatrix} c_{01} & c_{11} & \cdots & c_{P1} \\ c_{02} & c_{12} & \cdots & c_{P2} \\ \vdots & \vdots & \ddots & \vdots \\ c_{0N} & c_{1N} & \cdots & c_{PN} \end{bmatrix} \tag{2.28}$$

where each column represents the coefficient vector of a subfilter defines the structure of a *chromosome* for the optimization process. The issue of stability is handled by incorporating a stability criterion in the GA of Section 2.3.2 to screen out *unstable* solutions in the AIFS design. In *Step* 8 of Table 2.1, every chromosome is first tested for stability using the stability criterion given in Eqn. 2.27 and each chromosome that passes the test is then assigned a fitness value on the basis of the associated value of peak error in the phase delay given in Eqn. 2.25.

For initialization purposes, an AIFS filter is designed by using the LS technique [28]. Half of the initial GA population is created from neighborhood points of the

Table 2.4: Results of Design Examples (AIFS filters)

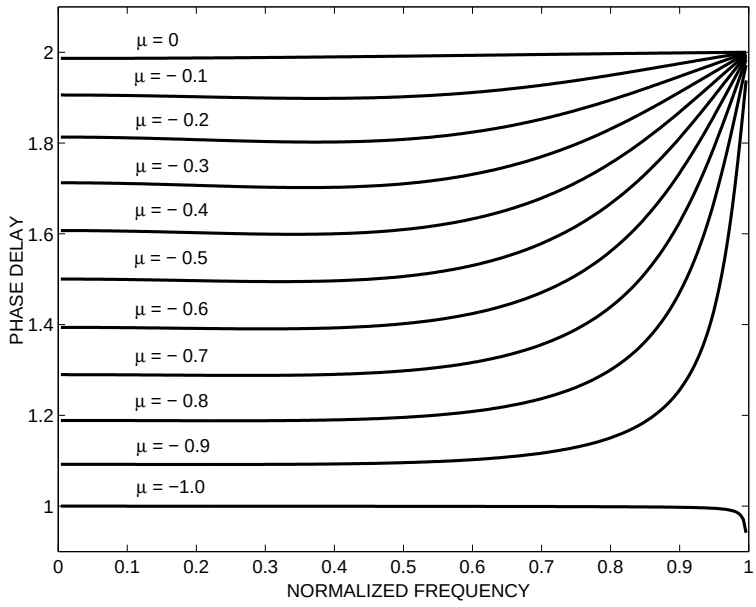
	Example 1	Example 2
ω_c	0.5π	0.8π
Number of subfilters	3	3
subfilter order	2	5
δ_d, LS method	1.8605×10^{-2}	1.458×10^{-2}
δ_d, GA method	1.370×10^{-2}	0.980×10^{-2}
Number of generations, GA	77	61
Computation time, LS	0.25 sec	1.95 sec
Computation time, GA	137.64 sec	128.55 sec

obtained design solution and the remaining is generated randomly as in the case of FDFS design.

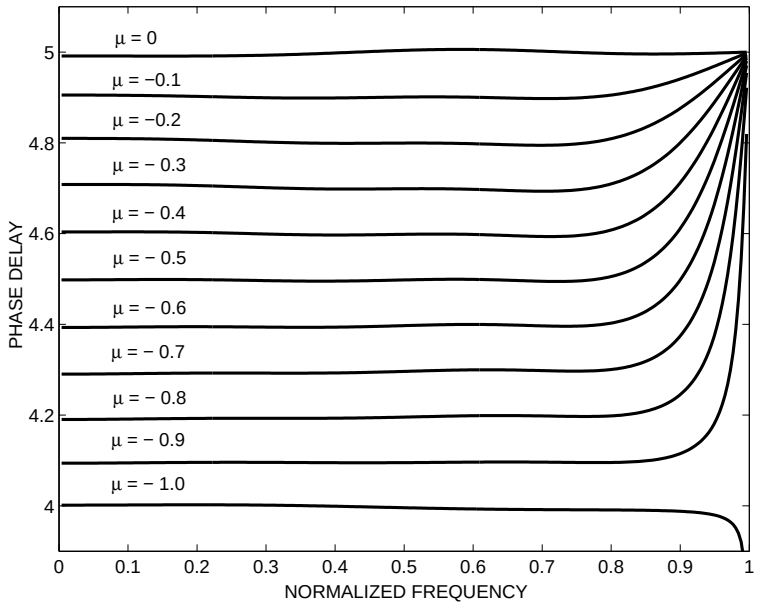
2.4.3 Design Examples and Results

Two AIFS filters were designed by using the GA. The desired specifications of these filters are listed in rows 1 to 3 of Table 2.4. The filter coefficients were encoded as 16-bit binary strings with initial crossover and mutation probabilities set to 0.75 and 0.05, respectively, and the population size was 48. The designs obtained with the GA approach are compared with the LS designs in rows 4 to 8.

The coefficients obtained using the LS method were quantized to the same precision as the binary GA coefficients for the sake of a fair comparison. Fig. 2.10 shows the phase delay of the designed AIFS filters and the maximum values of the delay errors for various values of μ are plotted in Fig. 2.11. Table 2.4 shows that the GA approach leads to improvements over the LS designs by providing lower delay errors. Reductions in the maximum delay errors of 26.4% and 32.8% were achieved



(a)



(b)

Figure 2.10: Phase delay achieved using the GA in optimized AIFS filters (a) Example 1 and (b) Example 2.

in Examples 1 and 2, respectively. Table 2.4 also gives the CPU time required by the two approaches to obtain the designs. As for the FDFS filter designs, this GA approach also required much more computation than the LS approach. Fig. 2.12 shows the evolution of the objective function through successive generations for the examples.

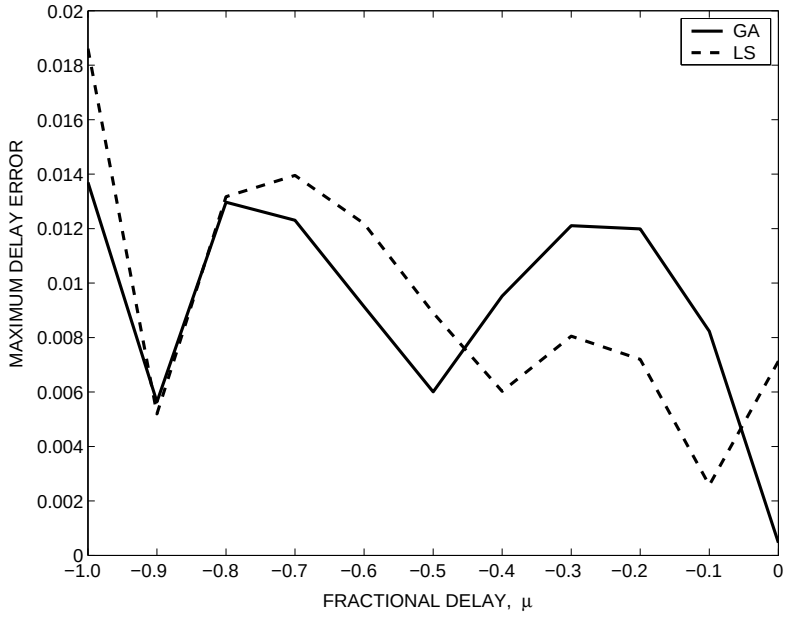
2.5 Conclusions

Two methods for the design of FDFS and AIFS filters based on GAs were introduced. The proposed methods lead to quantization-error-free designs.

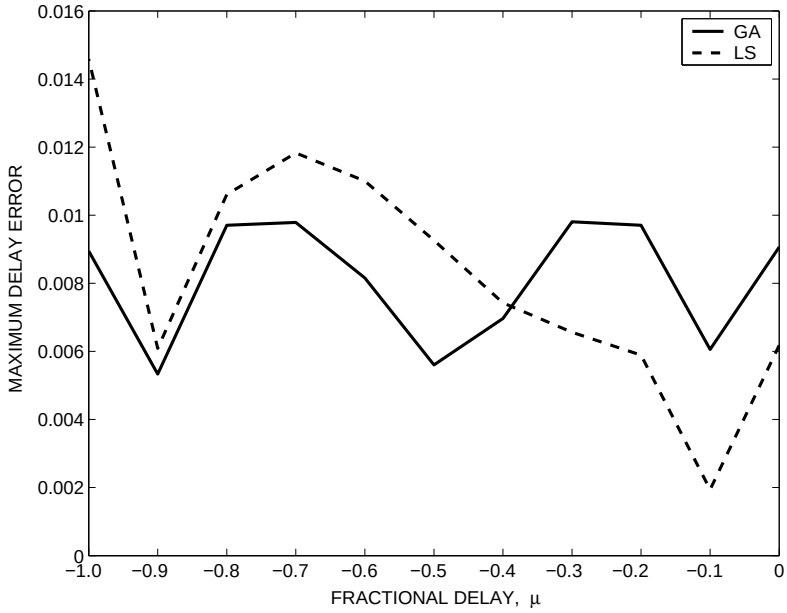
In the first method, the coefficient-symmetry constraints are removed in order to introduce flexibility in the design of FDFS filters. Simulations have shown that the proposed algorithm leads to reductions in the largest maximum amplitude-response and delay errors relative to those achieved with the design based on the LS method. For example, reductions in the amplitude-response and delay errors in the ranges 27.0-29.47% and 25.01-60.59%, respectively, were achieved in the FDFS filters designed as the examples.

The second method is based on an allpass structure and as a result it introduces flexibility in the optimization process since it removes the additional burden of optimizing the amplitude response. Furthermore, since the algorithm is programmed to discard unstable designs, the stability of the obtained IIR-based FD filters is guaranteed. Experimental results have shown that the proposed algorithm leads to reductions in the largest maximum delay errors relative to those achieved with the design based on the LS method. For example, reductions in the delay errors in the range 26.4-32.8% were achieved in the AIFS filters included in this chapter.

The proposed GAs require a much larger amount of computation than gradient-based algorithms as is usually the case in GAs in general.

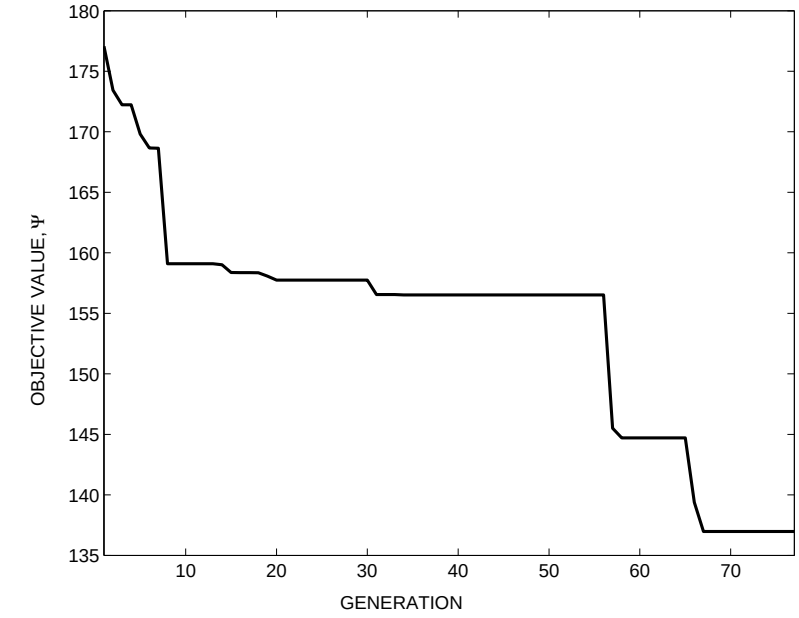


(a)

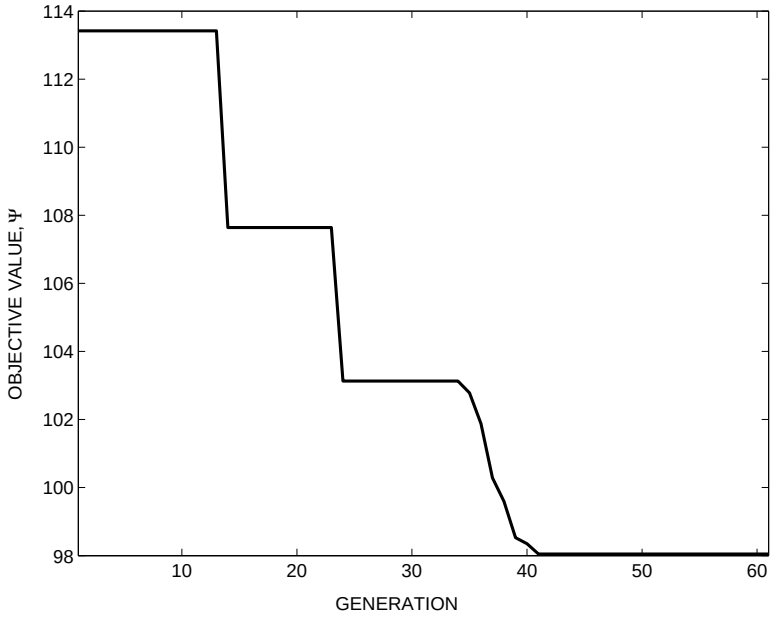


(b)

Figure 2.11: Maximum delay error as a function of the fractional delay (comparison between GA and LS methods) in optimized AIFS filters (a) Example 1 and (b) Example 2.



(a)



(b)

Figure 2.12: Evolution of objective function through the generations in optimizing AIFS filters (a) Example 1 and (b) Example 2.

Chapter 3

Design of Digital IIR Delay Equalizers

3.1 Introduction

In this chapter, a GA-based approach for the design of delay equalizers is presented. In the proposed approach, a GA is used to minimize the combined filter-equalizer group-delay deviation with respect to a specified passband. The GA employs a decimal-valued encoding scheme and an adaptive perturbation technique for crossovers and mutations where the elements of a parent chromosome are perturbed by small amounts to obtain an offspring. The algorithm developed entails a sequential optimization approach whereby new second-order equalizer sections are added until the desired amount of equalization is achieved [1]. The proposed GA leads to stable equalizer designs that would satisfy arbitrary prescribed specifications.

The chapter is organized as follows. Section 3.2 introduces the design problem of IIR equalizers. The constraints essential to assure equalizer stability is discussed in Section 3.3. Section 3.4 describes the methodology used for the design of equalizer. Design examples are presented in Section 3.5 and conclusions are drawn in Section 3.6.

3.2 Design of IIR Equalizers

The group delay of an IIR filter can be equalized by connecting an equalizer in cascade with the filter as shown in Fig. 3.1a.

An Mth-order recursive filter can be represented by a transfer function of the form

$$H_F(z) \;=\; H_0 \prod_{j=1}^{M/2} \frac{a_{0j} \;+\; a_{1j} z \;+\; z^2}{b_{0j} \;+\; b_{1j} z \;+\; z^2} \tag{3.1}$$

where a_{0j}, a_{1j} and b_{0j}, b_{1j} are real coefficients, $M/2$ is the number of second-order filter sections, and H_0 is a positive multiplier constant. By substituting $z = e^{j\omega T}$ where ω is the frequency in rad/s and T is the sampling period in seconds, the frequency response of the filter can be written as

$$H_F(e^{j\omega T}) \;=\; M(\omega) e^{-j\omega\tau_F} \tag{3.2}$$

where $M(\omega)$ and τ_F represent the amplitude response and group delay of the filter. The group delay of the filter, τ_F, is defined as

$$\tau_F(\omega) \;=\; -\frac{d\theta_F(e^{j\omega T})}{d\omega} \tag{3.3}$$

where

$$\theta_F(\omega) \;=\; \arg\, H_F(e^{j\omega T}) \tag{3.4}$$

From Eqns. (3.1) and (3.4), the group delay can be obtained as [1]

$$\tau_F(\omega) \;=\; -T \sum_{j=1}^{M/2} \frac{\tilde{N}_j(\omega)}{N_j(\omega)} \;+\; T \sum_{j=1}^{M/2} \frac{\tilde{D}_j(\omega)}{D_j(\omega)} \tag{3.5}$$

where

$$\begin{aligned}
\tilde{N}_j(\omega) &= 1 - a_{0j}^2 + a_{1j}(1 - a_{0j}) \;\cos\omega T \\
N_j(\omega) &= (1 - a_{0j})^2 + a_{1j}^2 + 2a_{1j}(1 + a_{0j}) \;\cos\omega T + 4a_{0j}\cos^2\omega T \\
\tilde{D}_j(\omega) &= 1 - b_{0j}^2 + b_{1j}(1 - b_{0j}) \;\cos\omega T \\
D_j(\omega) &= (1 - b_{0j})^2 + b_{1j}^2 + 2b_{1j}(1 + b_{0j}) \;\cos\omega T + 4b_{0j}\cos^2\omega T
\end{aligned}$$

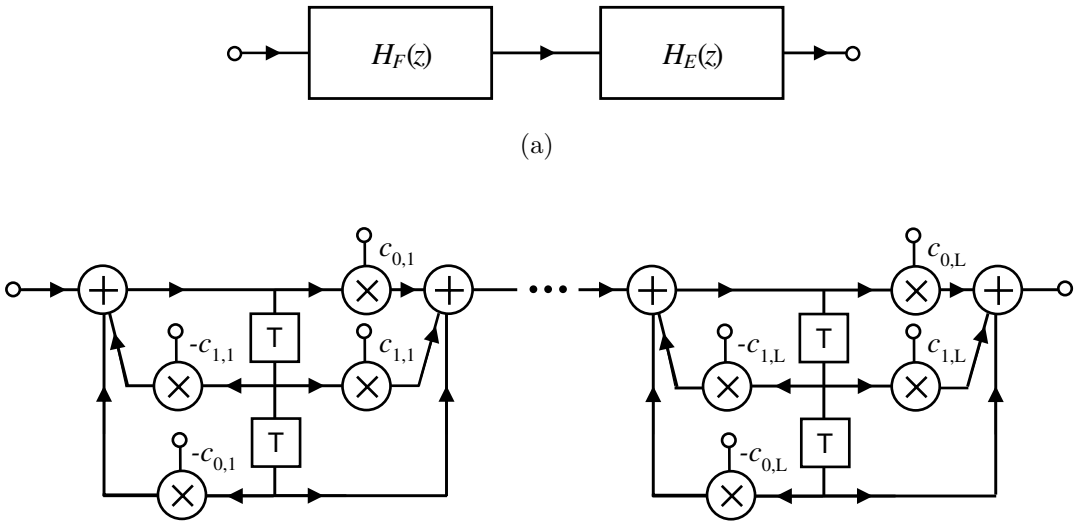


Figure 3.1: (a) Equalized IIR filter and (b) canonic realization of L-section equalizer.

The transfer function of the equalizer can be expressed as

$$H_E(z) \;=\; \prod_{j=1}^{N/2} \frac{1 \;+\; c_{1j} z \;+\; c_{0j} z^2}{c_{0j} \;+\; c_{1j} z \;+\; z^2} \tag{3.6}$$

where $N/2$ is the number of equalizer sections. The group delay of the equalizer can be obtained as

$$\tau_E(\mathbf{x}, \omega) \;=\; -\,\frac{d\theta_E(e^{j\omega T})}{d\omega}$$

where

$$\theta_E(\mathbf{x}, \omega) \;=\; \arg\, H_E(e^{j\omega T})$$

Hence

$$\tau_E(\mathbf{x}, \omega) \;=\; 2T \sum_{j=1}^{L} \frac{\tilde{C}_j(\omega)}{C_j(\omega)} \tag{3.7}$$

where

$$\begin{aligned}\tilde{C}_j(\omega) &= 1 - c_{0j}^2 + c_{1j}(1 - c_{0j})\ \cos\omega T \\ C_j(\omega) &= (1 - c_{0j})^2 + c_{1j}^2 + 2c_{1j}(1 + c_{0j})\ \cos\omega T + 4c_{0j}\cos^2\omega T\end{aligned}$$

and

$$\mathbf{x} \ = \ [c_{01} \ \ c_{11} \ \ c_{02} \ \ c_{12} \ \ \cdots \ \ c_{0L} \ \ c_{1L}] \tag{3.8}$$

is a vector representing the equalizer coefficients.

The group delay of the filter-equalizer combination can be obtained from Fig. 3.1a as

$$\tau_{FE}(\omega) \ = \ \tau_F(\omega) \ + \ \tau_E(\omega) \tag{3.9}$$

The group delay of the filter can be equalized with respect to a specified passband $\omega_{p1} \leq \omega \leq \omega_{p2}$ by minimizing the variation in $\tau_{FE}(\omega)$ in order to obtain a flat group delay with respect to the passband. The flatness of the group delay can be measured in terms of the percentage difference between the maximum and minimum group delay relative to the mean group delay [50], namely,

$$Q \ = \ \frac{100\ (\hat{\tau}_{FE} \ - \ \check{\tau}_{FE})}{\hat{\tau}_{FE} \ + \ \check{\tau}_{FE}} \tag{3.10}$$

where $\hat{\tau}_{FE}$ and $\check{\tau}_{FE}$ represent the maximum and minimum values of the passband group delay of the filter-equalizer combination. Hence, minimization of Q will yield a minimax solution. Typically a value of Q in the range between 1 and 10 percent is acceptable depending upon the application.

3.3 Stability of IIR Equalizer

Stability constraints must be imposed to ensure that the designed IIR equalizers are stable. An IIR filter is considered stable if and only if its poles are located within the

unit circle of the z plane [1]. Since the transfer function in Eqn. 3.6 is represented as a product of second-order transfer functions, the two poles of each second-order transfer function must be located inside the unit circle to guarantee stability.

An IIR filter is stable if and only if the denominator of its transfer function is a Schur polynomial [77]. A monic polynomial $p(z)$ with real coefficients is said to be a Schur polynomial if the roots of $p(z) = 0$ are located strictly inside the unit circle.

Now a second-order monic polynomial

$$p_j(z) \;=\; c_{0j} \;+\; c_{1j}z \;+\; z^2 \tag{3.11}$$

is a Schur polynomial if and only if [1], [77]

$$c_{0j} \;<\; 1 \tag{3.12a}$$

$$c_{1j} \;-\; c_{0j} \;<\; 1 \tag{3.12b}$$

$$c_{1j} \;+\; c_{0j} \;>\; -1 \tag{3.12c}$$

As a result, the stability region for $p_j(z)$ is an open triangle in the parameter space of c_{0j}, c_{1j} as illustrated in Fig. 3.2. Hence, for stability, an equalizer with a transfer function of the form given by Eqn. 3.6 must satisfy the conditions in Eqn. 3.12 for $j \;=\; 1, 2, \;\ldots\; , L$.

3.4 The GA Approach

In this section, we describe a GA that can be used to design delay equalizers for IIR digital filters by minimizing parameter Q in Eqn. 3.10 with respect to the equalizer coefficients $\mathbf{x}$ in Eqn. 3.8.

Coefficient vector $\mathbf{x}$ is expressed in matrix form such that each column in the matrix represents the coefficients of an equalizer section. The matrix defines the

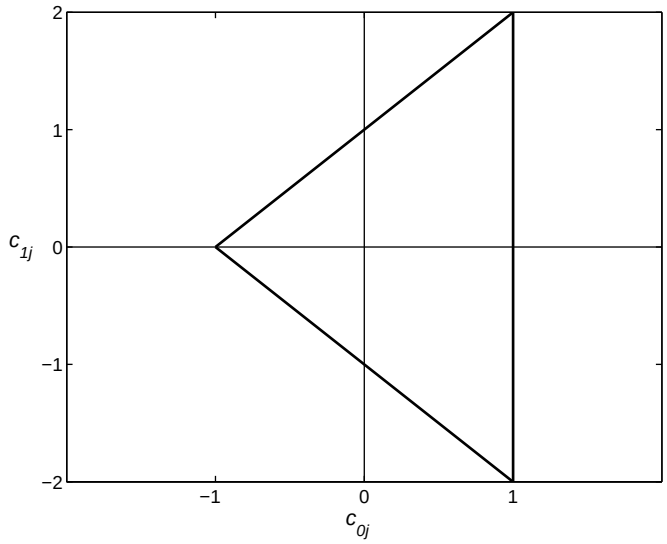


Figure 3.2: Stability triangle for polynomial $p_j(z)$.

structure of a chromosome for the optimization process. The GA creates new generations of chromosome-population by applying specially designed crossover and mutation techniques and a steady-state type selection method. A *random insertion* technique is also introduced to enable the GA to explore new areas of the parameter space away from the locales occupied by the set of the current best-fit solutions.

Typically the chromosomes in GAs are encoded in binary form but for the problem under consideration the obtained delay characteristic is greatly affected by the precision of equalizer coefficients. Consequently, encoding the chromosomes would result in very long binary strings. To avoid this difficulty and achieve the required precision, a floating-point representation is used. Different approaches for crossovers and mutations are explored to address the sensitivity of the equalizer coefficients.

The standard crossover operation (see Sec. 1.2.6 in Chap. 1) is replaced by an adaptive perturbation technique whereby the elements of a parent chromosome are perturbed by small amounts to obtain an offspring. Adaptability is achieved by

introducing a factor to control the level of perturbation. Initially, relatively large perturbations are applied. As time advances, the level of perturbation is reduced exponentially. The basis behind this approach is that at the beginning when a large region needs to be explored, bigger steps are used to enable the algorithm to ignore small bumps and avoid shallow minima. When a good solution is approached, smaller steps are used to ensure that an accurate solution is achieved. The crossover process is detailed in Table 3.1.

Table 3.1: Crossover Operation

a: Define a crossover probability of occurrence P_x (usually taken in the range 80 to 100%)

b: Randomly select a chromosome and generate a random random number $R \in [0, 1]$ assuming uniform distribution.

c: If $R \leq P_x$, apply crossover on this chromosome according to the rules:

i) Generate random numbers, d_{ij}, assuming a normal distribution with zero mean and variance of one to create a perturbation matrix. Keep the dimension of the matrix identical to that of the chromosome.

ii) Compute the coefficients for an offspring as $c_{ij(new)} = c_{ij(old)} + \alpha d_{ij}$ where α is a control factor as in Eqn. 3.13, which limits the perturbations to the parent chromosome coefficients.

The control factor α used in the crossover operation is given by

$$\alpha \;=\; 0.4\; e^{-K/15} \tag{3.13}$$

where K is an integer with the initial value of one. The value of K is increased by one if no improvement in the solution is achieved after a number of generations, say, F.

Mutation is applied in a manner similar to that of crossover. A fixed occasional perturbation is applied to randomly chosen coefficients of a chromosome to obtain a mutated offspring. In the proposed GA, the frequencies of crossovers and mutations are controlled by probabilities of occurrence P_x and P_m. In each case, a random number is generated in the range 0 to 1 and if it is less than P_x or P_m, as appropriate, crossover or mutation is applied; otherwise, neither is applied. For the problem at hand, mutation is essentially treated as a supporting operator to prevent premature convergence. The mutation rate is usually kept low in the range of 1 to 5%.

In the selection process, every chromosome is first tested for stability using the stability criterion given in Eqn. 3.12 and each chromosome that passes the test is then assigned a fitness value on the basis of the associated value of Q. A chromosome that yields lower Q receives higher fitness. The chromosomes are then ranked on the basis of their fitness. The total population competing for survival comprises parents and the offsprings created during crossover and mutation. The algorithm then selects a fixed number of best-fit chromosomes as potential parents for the next generation. To maintain diversity in the solutions under exploitation, a random insertion technique is implemented whereby a number of new randomly created chromosomes are added to the population in each generation. In the proposed GA, two-thirds of the population are taken from the set of best fits of the previous generation whereas one-third originates from the random insertion technique.

The best fitness value in a generation is defined as the minimum of all fitness values for a population and can be expressed as

$$\Psi \;=\; \min Q_i \quad \text{for } i = 1, ..., N_p \tag{3.14}$$

where N_p is the size of the population. If the reduction in Ψ in a given generation is less than ε, where ε is a small constant, the generation is deemed to be an 'unproductive generation'.

As in the GA presented in Section 2.3.2, a scheme of varying crossover and mutation probabilities of occurrence is adopted. This scheme enables the GA to terminate a cycle of unproductive generations by creating offsprings in a new area of the parameter space. When the number of unproductive generations exceeds a pre-defined number, say, K_1, the values of P_x and P_m are updated as

$$\begin{aligned} P_x &= 1.05Px \ \leq \ \hat{P}_x \\ P_m &= 1.1Pm \ \leq \ \hat{P}_m \end{aligned}$$

where $\hat{P}_x$ and $\hat{P}_m$ are the specified upper limits for P_x and P_m, respectively.

Typically, a GA is stopped when new crossovers and mutations cease to improve the solution, and this situation can best be checked by allowing the GA to run over a prespecified number of unproductive generations. The proposed GA is allowed to continue for G unproductive generations. When the number of unproductive generations or a predefined maximum number of generations, say, $\hat{G}$, is exceeded, the GA stops and the best chromosome is selected as the desired solution.

In order to equalize a recursive filter, the equalizer order that would satisfy the desired specifications is required but, unfortunately, no methods are available that can be used to predict the equalizer order (see Sec. 16.8 in [1]). To overcome this problem, a sequential design approach is adopted whereby new equalizer sections are added until the desired specifications are satisfied.

The GA is initialized with a randomly generated population but as the algorithm progresses the best solution of the previous design is added to the initial population of the subsequent design. In this way, the GA tends to advance towards more promising domains of the parameter space.

When the number of independent parameters in the optimization problem at hand is small, GAs tend to converge to a solution relatively quickly but if the number of parameters is large, a large number of iterations is usually required. In the sequential design approach adopted in this dissertation, the number of independent parameters increases as more and more sections are added and, therefore, a relatively small maximum number of generations, $\hat{G}$, should be used initially in order to achieve fast convergence. As the number of sections, is increased the value $\hat{G}$ should be increased to ensure that convergence is achieved before termination. For a j-section equalizer, selecting $\hat{G}$ according to the rule

$$\hat{G} \;=\; \hat{G}_i \;+\; 100(j-1) \quad \text{for } j = 1, ..., L \tag{3.15}$$

where $\hat{G}_i$ is the initial value of $\hat{G}$ for a 1-section equalizer, was found to work well.

Based on our experience with the proposed GA, suitable values for parameters N_p, $\hat{P}_x$, $\hat{P}_m$, K, K_1, $\hat{G}_i$, and ε for the problem at hand are 48, 1.0, 0.25, 15, 20, 50, and 10^{-2}, respectively.

3.5 Design Examples and Results

The proposed GA was used to design equalizers for a diverse range of *elliptic* and *Chebyshev* recursive digital filters of various types (lowpass, highpass and bandpass). Two of these equalizers were designed for elliptic highpass and bandpass filters satisfying the prescribed specifications in [1]. For the designs presented, the initial probabilities of crossover and mutation were assumed to be 0.75 and 0.075, respectively, and the population size was taken to be 48.

Example 1 – The GA was used to design a delay equalizer for a highpass elliptic IIR filter satisfying the specifications listed in Table 3.2. The results obtained are listed in Table 3.3 and the group delay characteristics are plotted in Fig. 3.3.

Table 3.2: Highpass Filter Specifications

Maximum passband ripple, δ_p, dB	0.50
Minimum stopband attenuation, δ_a, dB	50
Stopband edge, ω_a, rad/s	0.64
Passband edge, ω_p, rad/s	0.75
Sampling frequency, ω_s, rad/s	2.0

Table 3.3: Results of Design Example 1

j	Filter + j-Section Equalizer Q_j	Coefficients of 5-Section Equalizer	
		c_{0j}	c_{1j}
0	67.522	-	-
1	34.919	0.67910546446626	1.60599136090483
2	19.677	0.69201940470724	1.55283561907181
3	9.055	0.79539643727247	1.37618813719757
4	3.713	0.66622722070854	1.62771709258158
5	1.694	0.72744825101826	1.47871671250912

Initially, a one-section equalizer was designed and the number of equalizer sections was increased successively from one to five in order to reduce parameter Q to a value less than 2%. The numbers of equalizer sections and the corresponding values of Q are listed in columns 1 and 2 of Table 3.3, respectively, and the coefficients of the 5-section equalizer achieved are listed in columns 3 and 4. The evolution of the

objective function for the 2-, 3-, 4-, and 5-section equalizers is plotted in Fig. 3.4a to d. Fig. 3.5 shows the variations in crossover and mutation probabilities in the optimization of final 5-section equalizer.

As can be seen in Fig. 3.3, the achieved group delay characteristic is almost flat with respect to the passband with a value of Q of 1.69%. The results obtained are compared with corresponding results described in Example 16.5 of [1] (see p. 759) in Table 3.4, which were obtained using a gradient-based algorithm.

Table 3.4: Comparison of GA and Design Method in [1] (Example 1)

	GA	Method in [1]
Number of equalizer sections	5	5
$\tilde{\tau}_{FE}/T$ [a]	38.55	39.08
Minimum $\tau_{FE}/\tilde{\tau}_{FE}$	0.98	0.99
Maximum $\tau_{FE}/\tilde{\tau}_{FE}$	1.03	1.03
Q_{FE}	1.69	0.83

[a] $\tilde{\tau}_{FE}$ = Mean value of τ_{FE}

Example 2 – The GA was also used to design a delay equalizer for a bandpass elliptic IIR filter of desired specifications listed in Table 3.5. The results obtained are listed in Table 3.6 and the group delay characteristics are plotted in Fig. 3.6. Initially, a one-section equalizer was designed and the number of equalizer sections was increased successively from one to five in order to reduce parameter Q to a value less than 2%. The numbers of equalizer sections and the corresponding values of Q are listed in columns 1 and 2 of Table 3.6, respectively, and the coefficients of the 5-section equalizer achieved are listed in columns 3 and 4. The evolution of the objective function for the 2-, 3-, 4-, and 5-section equalizers is plotted in Fig. 3.7a to d. Fig. 3.8 shows the variations in crossover and mutation probabilities in the

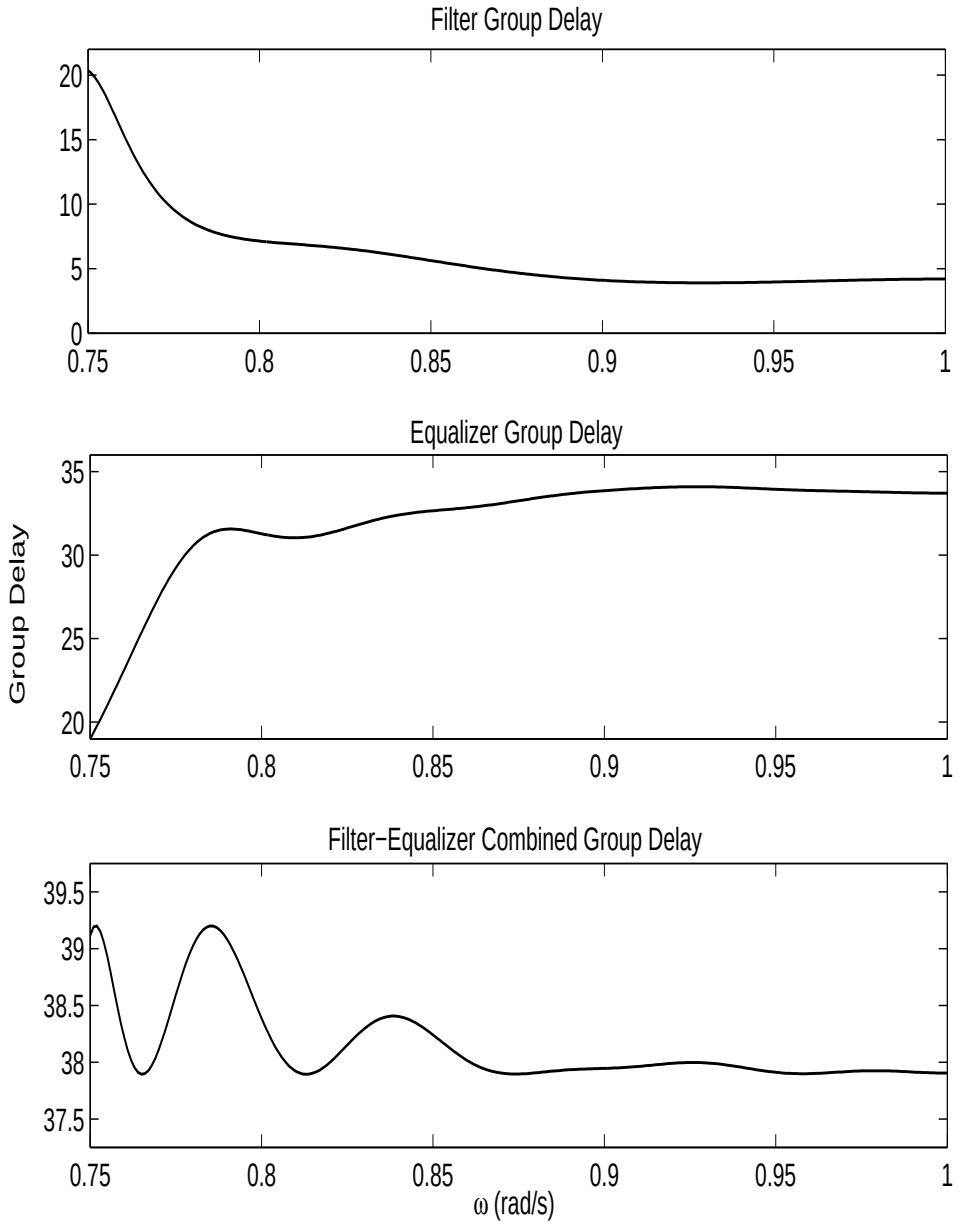


Figure 3.3: Group delay equalization of an elliptic highpass filter (Example 1).

optimization of final 5-section equalizer.

As can be seen from Fig. 3.6, the group-delay characteristic tends to become

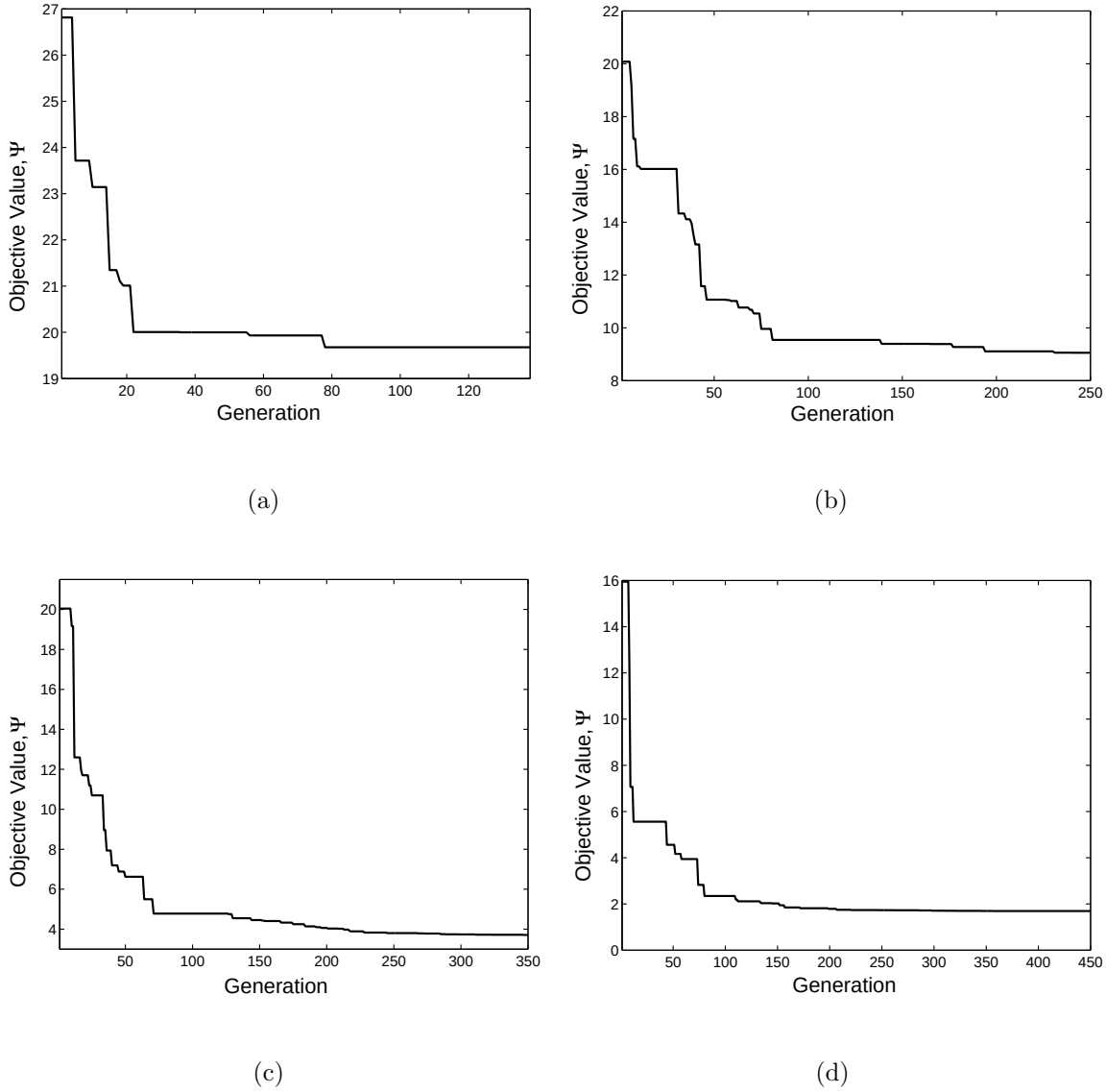
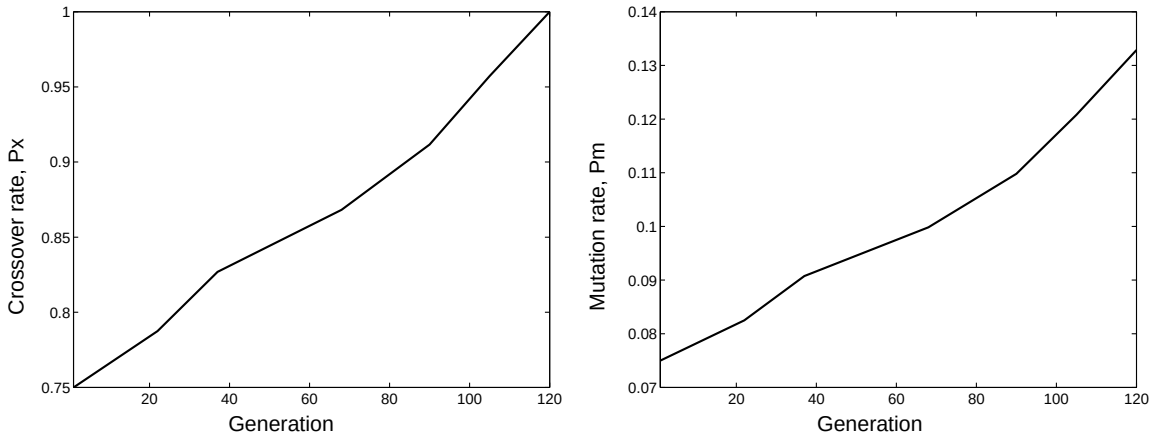


Figure 3.4: Variation in objective function through generations for a) 2-, b) 3-, c) 4-, and d) 5-section equalizers (Example 1).

equiripple with respect to the passband as in the case where quasi-Newton minimax algorithms are used [1], [3]. The results obtained are compared with corresponding results described in Example 16.6 of [1] (see p. 761) in Table 3.7, which were obtained



(a) (b)

Figure 3.5: Variation in a) crossover rate, P_x and b) mutation rate in optimizing the final 5-section equalizer (Example 1).

Table 3.5: Bandpass Filter Specifications

Maximum passband ripple, δ_p, dB	1
Minimum stopband attenuation, δ_a, dB	40
Lower stopband edge, ω_{a1}, rad/s	0.2
Lower passband edge, ω_{p1}, rad/s	0.3
Upper passband edge, ω_{p2}, rad/s	0.5
Upper stopband edge, ω_{a2}, rad/s	0.7
Sampling frequency, ω_s, rad/s	2.0

using a gradient-based algorithm.

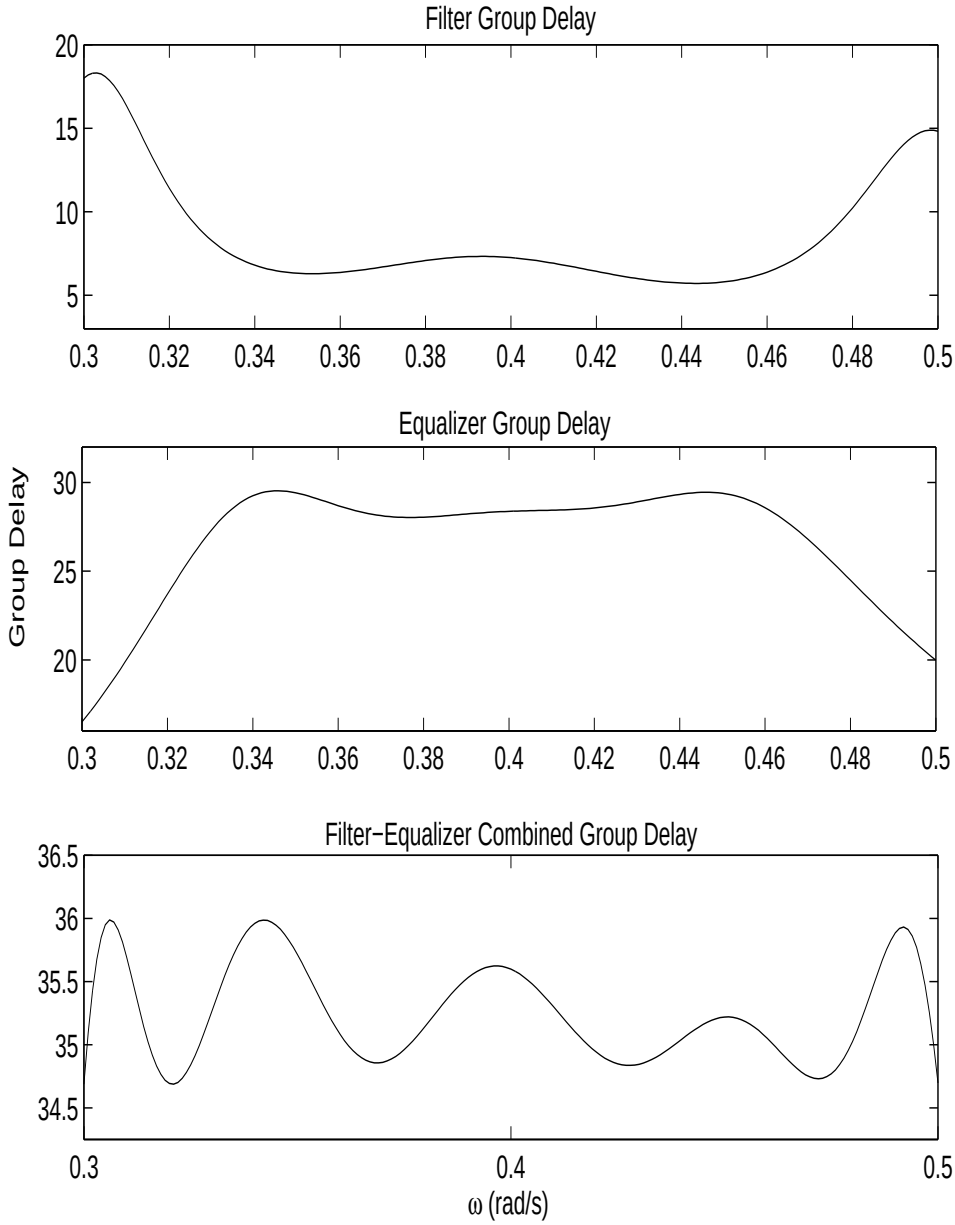


Figure 3.6: Group delay equalization of an elliptic bandpass filter (Example 2).

Table 3.6: Results of Design Example 2

j	Filter + j-Section Equalizer Q_j	Coefficients of 5-Section Equalizer	
		c_{0j}	c_{1j}
0	52.464	-	-
1	27.613	0.74529812615567	−0.23916280501736
2	9.436	0.78771465962887	−0.86738472409954
3	3.423	0.70879892778405	−0.54614620896512
4	2.507	0.39119806953664	0.08398848979643
5	1.839	0.40217710443492	−0.04719550044842

Table 3.7: Comparison of GA and Design Method in [1] (Example 2)

	GA	Method in [1]
Number of equalizer sections	5	4
$\tilde{\tau}_{FE}/T$	35.33	32.44
Minimum $\tau_{FE}/\tilde{\tau}_{FE}$	0.98	0.98
Maximum $\tau_{FE}/\tilde{\tau}_{FE}$	1.03	1.05
Q_{FE}	1.84	1.96

3.6 Conclusions

A GA-based sequential optimization approach for the design of delay equalizers whereby new second-order equalizer sections are added until the desired amount of equalization is achieved has been proposed. The GA minimizes an objective

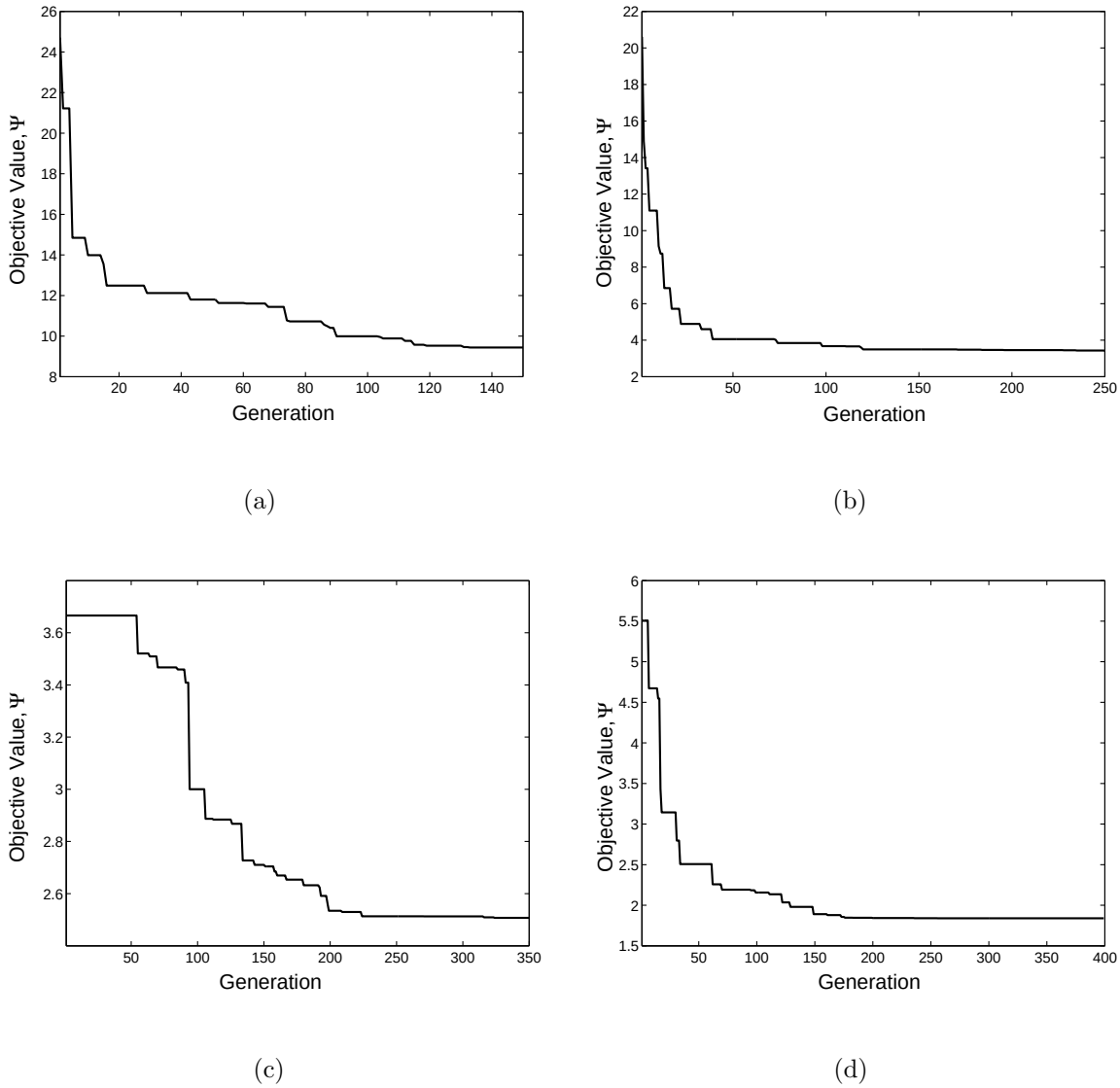
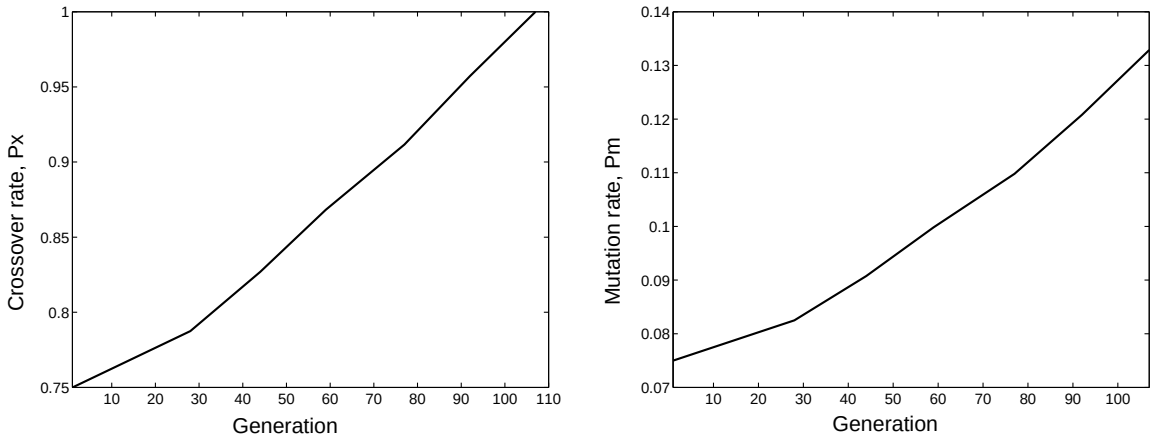


Figure 3.7: Variation in objective function through generations for a) 2-, b) 3-, c) 4-, and d) 5-section equalizers (Example 2).

function based on the passband filter-equalizer group-delay deviation and it can be programmed to discard designs with undesirable features, for example, unstable designs. Experimental results have shown that *stable* delay equalizers satisfying



(a) (b)

Figure 3.8: Variation in a) crossover rate, P_x and b) mutation rate in optimizing the final 5-section equalizer (Example 2).

arbitrary prescribed specifications with the desired degree of group-delay flatness can be easily achieved. Comparisons of the results obtained with the GA with corresponding results described in [1], which were obtained with a state-of-the-art gradient-based algorithm, are comparable. Although the gradient-based approach performed marginally better in terms of the quality of equalization, the merit of the GA is that it always gives a stable design whereas the gradient-based algorithm reported in [1] may sometimes fail to give a stable design. Improved results could be achieved with the GA by adopting the initialization strategy used in Chap. 16 of [1] and also tuning the termination criteria and/or crossover and mutation operators of the GA to this particular problem. However, like the GAs of Chap. 2, the GA described in this chapter requires much more computation than gradient-based algorithms.

Chapter 4

Design of Multiplierless FIR Filters

4.1 Introduction

In this chapter, an optimization approach based on a genetic algorithm for the design of multiplierless FIR filters is presented. The approach exploits a recently-introduced GA, called orthogonal GA (OGA), based on the so-called experimental design technique. The OGA approach is applied to a cascade FIR realization, where the coefficients are expressed in terms of sums of powers of two (SOPOT) to obtain a multiplierless or fixed-point design. The OGA employs an integer encoding scheme for chromosome construction whereby the filter coefficients are optimized so as to achieve prescribed amplitude response specifications.

Chromosomes are constructed in matrix form with each row representing the powers of two (POT) terms of the coefficients of a cascade section. The algorithm developed entails a sequential optimization approach for two direct-form cascaded subfilters, which leads to an improved amplitude response.

The chapter is organized as follows. In Section 4.2, the design problem of cascade-form multiplierless FIR filters is presented. To explain the basis of the proposed method, a brief overview of the orthogonal design method is given in Section 4.3.

Details regarding the methodology of the OGA approach are supplied in Section 4.4. Design examples and results are presented in Section 4.5 and conclusions are drawn in Section 4.6.

4.2 Design of Cascade-Form Multiplierless FIR Filters

4.2.1 Cascade-Form FIR Filters

An FIR causal filter can be characterized by the transfer function

$$H(z) \;=\; \sum_{n=0}^{N-1} h(n) z^{-n} \tag{4.1}$$

To achieve linear phase response the impulse response must satisfy the symmetry property

$$h(n) \;=\; h(N-1-n) \quad \text{for } 0 \le n \le N-1 \tag{4.2}$$

The transfer function of a linear-phase FIR filter can be represented by a *mirror-image polynomial* [1]. A typical zero-pole plot for such polynomial is shown in Fig. 4.1. An even-order transfer function with distinct roots can be decomposed into a product of second- and fourth-order subfilters as

$$H(z) \;=\; \prod_{k=1}^{K} H_k(z) \tag{4.3}$$

where

$$H_k(z) \;=\; 1 - (r_k + r_k^{-1}) z^{-1} + z^{-2} \tag{4.4}$$

for mirror image real zeros,

$$H_k(z) \;=\; 1 - 2(\cos\omega_k) z^{-1} + z^{-2} \tag{4.5}$$

for complex conjugate zeros on the unit circle, and

$$H_k(z) \;=\; 1 - [(r_k + r_k^{-1})\cos\omega_k] z^{-1} + (r_k^2 + 4\cos^2\omega_k + r_k^{-2}) - [2(r_k + r_k^{-1})\cos\omega_k] z^{-2} + z^{-4} \tag{4.6}$$

for mirror image complex conjugate zeros off the unit circle.

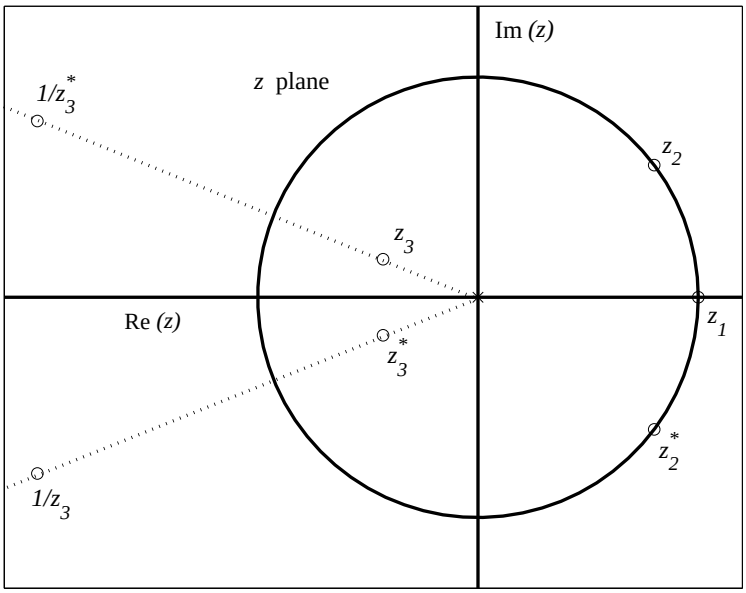


Figure 4.1: Typical zero-pole plot for a mirror-image polynomial.

In Eqn. 4.3, K denotes the number of subfilters having real-valued coefficients and $z_k = r_k e^{j\omega_k}$ is the kth zero of the transfer function. The coefficients of each subfilter satisfy the same symmetry as in the original filter. These subfilters are considered as the 'basic building blocks' and can be combined in cascade to obtain one or more longer subfilters. Two subfilters with transfer functions $H_{01}(z)$ and $H_{02}(z)$ can be formed by selecting second- and fourth-order blocks in a way such that their zeros are interleaved as in [78]. If θ_k is the phase angle of zero z_k, the corresponding basic blocks can be arranged such that

$$|\theta_k| \;\leq\; |\theta_{k+1}|$$

Under these circumstances, $H_{01}(z)$ and $H_{02}(z)$ can be expressed as

$$H_{01}(z) \quad = \quad \cdots H_{k-2} H_k H_{k+2} \cdots \tag{4.7a}$$

$$H_{02}(z) \quad = \quad \cdots H_{k-1} H_{k+1} H_{k+3} \cdots \tag{4.7b}$$

To allow simple multiplierless implementation, the coefficients of $H_{01}(z)$ and $H_{02}(z)$ can be expressed in terms of SOPOTs.

Due to the symmetry condition, the coefficient vector and the frequency response of an even-order subfilter of order $N-1$ can be represented by

$$h_k(n) \;=\; \left[h(0)\; h(1)\; \cdots\; h\left(\frac{N-1}{2}\right)\right]^{\mathrm{T}} \tag{4.8}$$

and

$$H_k(e^{j\omega}) \;=\; e^{-j\omega M}\sum_{n=0}^{M} h_k(n)\mathrm{cos}n\omega \quad \text{with } M=\frac{N-1}{2} \tag{4.9}$$

respectively, where a sampling period T of 1 s is assumed.

4.2.2 SOPOT Representation of Filter Coefficients

In multiplierless realization, a constant multiplicand, i.e., a filter coefficient $h(n)$ can be expressed in terms of POT digits $c_i(n)$ for $i = 1, 2, \ldots, I$. The SOPOT representation of the filter coefficient can be written as

$$h(n) = \sum_{i=1}^{I} d_i(n)2^{-p_i(n)} \tag{4.10}$$

where I is the wordlength,

$$d_i(n) = \mathrm{sgn}[c_i(n)] \in [-1, 0, 1] \tag{4.11}$$

$$p_i(n) = |c_i(n)| \in [0, 1, ..., S] \tag{4.12}$$

and S is a positive integer. The POT terms in the filter coefficients can take discrete values in increments of 2^{-1} in the range

$$-2^{-S} \le 2^{-|c_i(n)|} \le +2^{-S} \tag{4.13}$$

in which there are $2 \times 2^{S+1}$ distinct values. The coefficients can be represented with same or different numbers of total POT terms depending on the hardware platform used to implement the filter. In the proposed method, a fixed number of POT terms was assumed.

4.2.3 Optimization Problem

The design problem of multiplierless FIR filters can be formulated as an optimization problem as follows: Given the number of POT digits in the representation of the filter coefficients, I, and the dynamic range for the POT digits, S, find the set of POT digits for

$$c_i(n) \in [-S, ..., -1, 0, 1, ..., S]$$

which would yield an amplitude response that would satisfy the prescribed specifications.

Restricting the maximum number of POT digits in the SOPOT representation to two, the amplitude-response error can be minimized to obtain the optimized coefficients. The optimization can be carried out by minimizing an objective function based on one, two, or more criteria. One could, for example, use either the L_2- or L_∞-norm for all frequency bands or the L_∞-norm for passbands and the L_2-norm for stopbands. In this paper, both of these options are explored.

Minimization of an objective function based on the L_∞-norm yields a minimax solution. An important merit of minimax solutions is that the optimization error tends to become uniform with respect to the frequency range of interest as the solution is approached [1]. The maximum errors in the amplitude response in the passband and stopband can be expressed as

$$F_p \;=\; \max \left| 1 - |H(e^{j\omega})| \right| \quad \text{for } 0 \le \omega \le \omega_p \tag{4.14}$$

and

$$F_a \;=\; \max \left| H(e^{j\omega}) \right| \quad \text{for } \omega_a \le \omega \le \omega_s/2 \tag{4.15}$$

respectively, where ω_p and ω_a are the passband and stopband edges, respectively, and ω_s is the sampling frequency. The approximation errors are sampled uniformly at

frequency points ω_k for

$$k = 1, 2, \ldots, K_1 \quad \text{and} \quad 0 \leq \omega_k \leq \omega_p \quad \text{in the passband}$$

and

$$k = 1, 2, \ldots, K_2 \quad \text{and} \quad \omega_a \leq \omega_k \leq \omega_s/2 \quad \text{in the stopband}$$

respectively. If δ_p and δ_a represent the passband ripple and minimum stopband attenuation in dB, respectively, the L_∞-norm objective function can be expressed as

$$F_1 \;=\; \max\left(F_p,\; \frac{\delta_p}{\delta_a}F_a\right) \tag{4.16}$$

Since minimization of amplitude distortion is important for signals to be passed, the L_∞-norm is more appropriate for the passband. In many applications, especially where narrow-band filters are required, minimization of both the gain and total energy in the stopband are important. Consequently, an error measure based on the L_2-norm with a constraint on the maximum amplitude-response error would be more suitable for the stopbands in such a case [65]. The constrained L_2-norm error in the stopband can be expressed as

$$F_a^{'} \;=\; \sum_{k=1}^{K_2} \left|H(e^{j\omega})\right|^2 \quad \text{for } \omega_a \leq \omega \leq \omega_s/2 \tag{4.17}$$

subject to the constraint

$$\max \left|H(e^{j\omega})\right|_{dB} \leq \delta_a \quad \text{for } \omega_a \leq \omega \leq \omega_s/2 \tag{4.18}$$

Now on combining the passband and stopband error measures in Eqns. 4.14) and 4.17, the objective function

$$F_2 \;=\; W_p F_p + W_a F_a^{'} + \Delta \tag{4.19}$$

can be formulated, where W_p and W_a are positive weighting factors for the approximation errors in the passband and stopband, respectively, and Δ is a *penalty*

factor used to limit the peak amplitude error in the stopband. The penalty factor can be calculated by using the equation

$$\Delta = \begin{cases} |\delta_a| + \max|H(e^{j\omega})|_{dB} & \text{if } \max|H(e^{j\omega})|_{dB} > \delta_a \\ 0 & \text{otherwise} \end{cases} \quad \text{for } \omega_a \leq \omega \leq \omega_s/2 \qquad (4.20)$$

4.3 Orthogonal Experimental Design

The so-called *experimental design* is a plan for performing experiments to study the effects of *factor levels* on dependent variables [79]. The factors of an experimental design are variables that have two or more fixed values or levels. Many design experiments use special techniques for determining suitable combinations of factor levels for implementing the experiments and analyzing the results. The so-called *orthogonal design* is an efficient way of studying the effects of several factors simultaneously in a design experiment.

Let us assume that an experimental design consists of N factors with each factor having Q levels. The number of all possible combinations of the factor levels would be Q^N. However, for most practical situations, it is cost prohibitive and tedious to test all these possible combinations. *Orthogonal arrays* (OAs) are, therefore, developed to provide a means for optimal sampling of experiments. In orthogonal design, all estimable effects are uncorrelated [60]. For example, with three factors, each at two levels, there are $2^3 = 8$ possible combinations whereas in an orthogonal design the four best representative combinations would be chosen. The concept of orthogonality for this example is illustrated in Fig. 4.2.

An OA is a rectangular matrix designated as $L_M(Q^N)$

where

N = the number of columns representing factors

Q = the number of factor levels, and

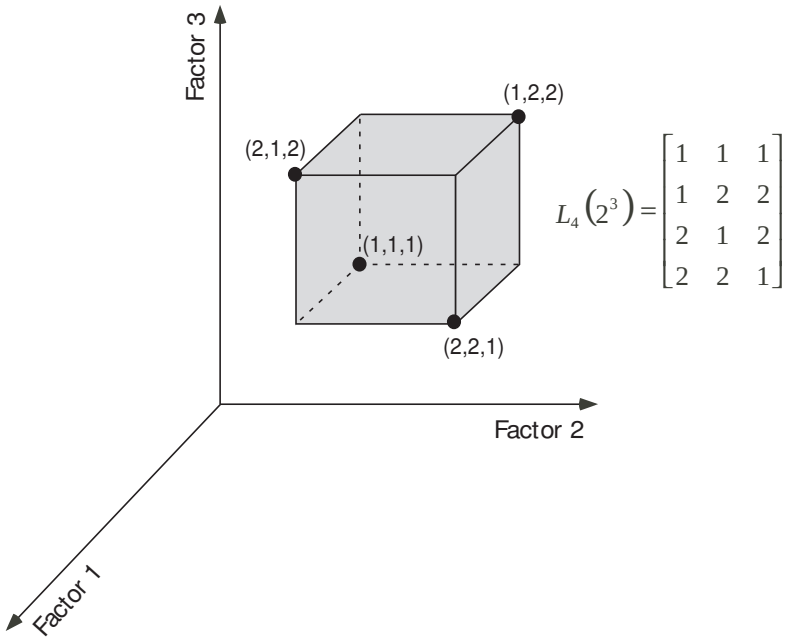


Figure 4.2: Concept of orthogonal experimental design with the edges representing the combination of levels and marked edges are the selected combinations. $L_4(2^3)$ is the corresponding orthogonal array.

M = the number of rows representing the combinations of levels, i.e., the number of experiments.

If the jth factor in the ith combination of an OA has level

$$a_{ij} \in (1, 2, \ldots, Q)$$

the OA can be represented as

$$L_M(Q^N) = [a_{ij}]_{M \times N} \tag{4.21}$$

In orthogonal experimental designs, OAs are used as tools to arrange experiments and find out the optimal design with fewer experiments. When an orthogonal array $L_M(Q^N)$ is tabulated, the following relation holds for Q odd and $M = Q^J$ [60],

$$N = \frac{Q^J - 1}{Q - 1} \tag{4.22}$$

An example of an OA, $L_4(2^3)$, is shown in Fig. 4.2.

4.4 OGA Approach

In this section, we describe an OGA that can be used to design multiplierless FIR filters by minimizing either the single criterion objective function F_1 in Eqn. 4.16 or the multi-criterion objective function F_2 in Eqn. 4.19.

The set of POT digits $c_i(n)$ that represents the filter coefficients $h(n)$ in terms of SOPOTs of the format in Eqns. 4.10-4.12 defines the phenotype structure of a chromosome for the optimization process. The OGA creates new generations of populations by applying an orthogonal crossovers and 'sign-inverting' mutations to the individuals of a population.

In the genotype representation, the chromosomes are encoded in terms of integer values which allow the OGA to employ the orthogonal design-based recombination process. The phenotype representation is mapped to the genotype representation with integer values limited in a specified range, which enables the search of the OGA to be confined to within a feasible region. A typical chromosome mapping is shown in Fig. 4.3.

$$[h(0)\ h(1)\ \cdots\ h(M)]^T \longrightarrow [c_1(0)\ c_2(0)\ c_1(1)\ c_2(1)\ \cdots\ c_1(M)\ c_2(M)\ ]^T$$

Impulse response — Chromosome

Figure 4.3: Chromosome mapping.

By analogy with the orthogonal experimental design in [79], if we define each filter coefficient as the 'factor' and the dynamic range of the SOPOT coefficients as the 'factor level', the design will have $N = 2M$ factors with each factor having S levels. An OA $L_K(S^N)$ would be necessary for the crossover operation for such a case.

For typical values of N and S for a filter with SOPOT coefficients, the size of an OA could be prohibitively large. To keep the size of the OA tractable, a fixed OA, such as, $L_9(3^4)$, can be used as a crossover tool. In the proposed method, the number of columns in the array of the OA is adjusted according to the number of elements in the chromosome. The OA as a whole or part of it is re-used after randomly re-arranging its elements to form an extended OA (EOA).

The solution evolves from generation to generation through the creation of offsprings by applying genetic operators such as crossover and mutation. To perform crossovers, a set of randomly selected chromosomes are used as rows in a matrix. An EOA is then applied as a 'mask' to re-distribute the chromosome elements in that group. Repeating the process of grouping and masking, a population-wide crossover can be achieved. An example of the orthogonal crossover operation is illustrated in Fig. 4.4. In the 'sign-inverting' mutation operation, the values of particular genes are changed by applying random 'sign' inversions. A table is constructed with all the individuals generated by the crossover and a population-wide mutation is performed. The frequency of mutations is controlled by probability of occurrence P_m. A random number is generated in the range 0 to 1 and if it is less than P_m, mutation is applied; otherwise, mutation is not applied. For the problem at hand, mutation is treated as a supporting operator for the purpose of restoring lost genetic material.

In the selection process, chromosomes are ranked on the basis of their fitness. The total population that competes for survival comprises the parents as well as the offsprings created during crossovers and mutations. The algorithm then records a small number of top-ranked solutions as *elite* solutions and selects a fixed number of best-fit chromosomes as potential parents for the next generation.

The best fitness value in a generation is defined as the minimum of all fitness

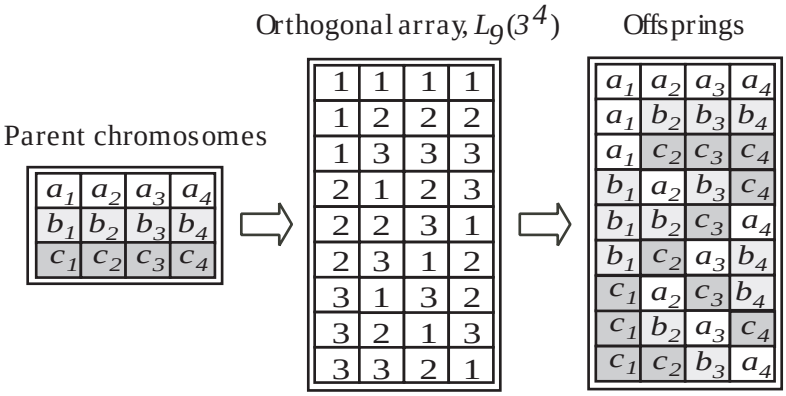


Figure 4.4: Orthogonal crossover applied to a set of chromosomes.

values for a population and can be expressed as

$$\Psi \;=\; \min\left(\Delta_i\right) \quad \text{for } i = 1, 2, ..., N_p \tag{4.23}$$

where N_p is the size of the population. If the reduction in Ψ in a given generation is less than ε, where ε is a small constant, the generation is deemed to be an 'unproductive generation'.

For the initialization of the algorithm, a direct-form FIR filter designed by using the Remez algorithm [1] is decomposed into two subfilters and the coefficients of the two subfilters are rounded to SOPOT values. The resulting coefficients are used to construct a chromosome string which is inserted in the initial population of the OGA. The remaining chromosomes are created randomly. In each successive generation, half of the chromosomes are taken from the best fits of the previous generation and the rest are generated randomly. The OGA is terminated if it fails to improve the best fitness value in a specified number of successive unproductive generations or if a pre-specified maximum number of generations is performed. Eventually, the best chromosome is selected as the desired solution.

The algorithm developed entails a sequential optimization approach for the SOPOT coefficients of the two cascaded subfilters. It is implemented by optimizing

the coefficients of one of the two subfilters using the OGA while keeping the coefficients of the other subfilter constant. Then the coefficients of the second subfilter are optimized while keeping the coefficients of the first subfilter constant, and so on, until the objective function cannot be improved any further. The steps involved are summarized in Table 4.1. The notation $\tilde{H}_{ij}(z)$ denotes the optimized version of $H_{ij}(z)$.

Table 4.1: Sequential Optimization using The OGA

Step 1: Design an FIR filter of length $N = 2M$ with transfer function $H(z)$ using the *Remez algorithm.*
Step 2: Factorize $H(z)$ into a product of transfer functions as $H(z) = H_{01}(z)H_{02}(z)$ according to the procedure described in Section 4.2.1.
Step 3: Optimize $H_{01}(z)$ with $H_{02}(z)$ fixed by minimizing an objective function given in Section 4.2.3 to obtain $H'(z) = H_{11}(z)H_{02}(z)$ where $H_{11}(z) = \tilde{H}_{01}(z)$.
Step 4: Optimize $H_{02}(z)$ with $H_{11}(z)$ fixed to obtain $H'(z) = H_{11}(z)H_{12}(z)$ where $H_{12}(z) = \tilde{H}_{02}(z)$.
Step 5: Optimize $H_{11}(z)$ with $H_{12}(z)$ fixed to obtain $H'(z) = H_{21}(z)H_{12}(z)$ where $H_{21}(z) = \tilde{H}_{11}(z)$.
Step 6: Optimize $H_{12}(z)$ with $H_{21}(z)$ fixed to obtain $H'(z) = H_{21}(z)H_{22}(z)$ where $H_{22}(z) = \tilde{H}_{12}(z)$.
Repeat Steps 3-6 until no further progress can be made.

4.5 Design Examples and Results

The proposed OGA was used to design several cascade FIR filters with SOPOT coefficients. The filters were designed separately by minimizing the objective functions based on single-criterion (Eqn. 4.16) and multi-criterion (Eqn. 4.19). The desired passband and stopband edges, ω_p and ω_a, in rad/s, and the filter lengths of two of these filters are given in rows 1 to 3 of Table 4.2. The chromosomes were defined as integer strings in the range of $[-10, 10]$. An EOA-based on the OA $L_9(3^4)$ was used as the crossover operator with the mutation probability of occurrence set to 5%, and the population size was 40. The initial solution used for the OGA was obtained by factorizing an even-length FIR filter designed by the Remez algorithm with the resulting coefficients rounded to SOPOT values.

The results achieved are summarized in rows 4 to 11 of Table 4.2 where δ_p and δ_a represent the passband ripple and minimum stopband attenuation, respectively. Rows 10 and 11 give the results obtained using the Remez method but with the coefficients rounded to the same number of bits. The results show that the use of the OGA approach leads to significant improvements over the initial cascade and the equivalent direct-form Remez designs with the coefficients rounded to SOPOT values in each example. Fig. 4.5 illustrates the amplitude response of the subfilters and the resulting cascaded filter of length 28 obtained by minimizing Eqn. 4.16 in Example 1. The SOPOT coefficients of the subfilters are listed in Table 4.3. The amplitude responses achieved by the initial cascade, OGA optimized, and Remez direct-form filters for Examples 1 and 2 are plotted in Fig. 4.6 and 4.7, respectively.

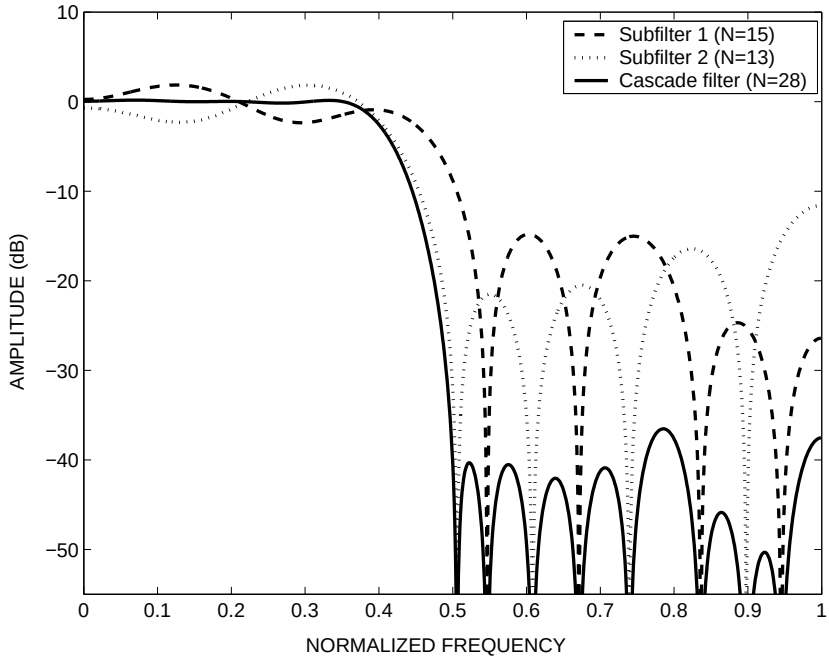
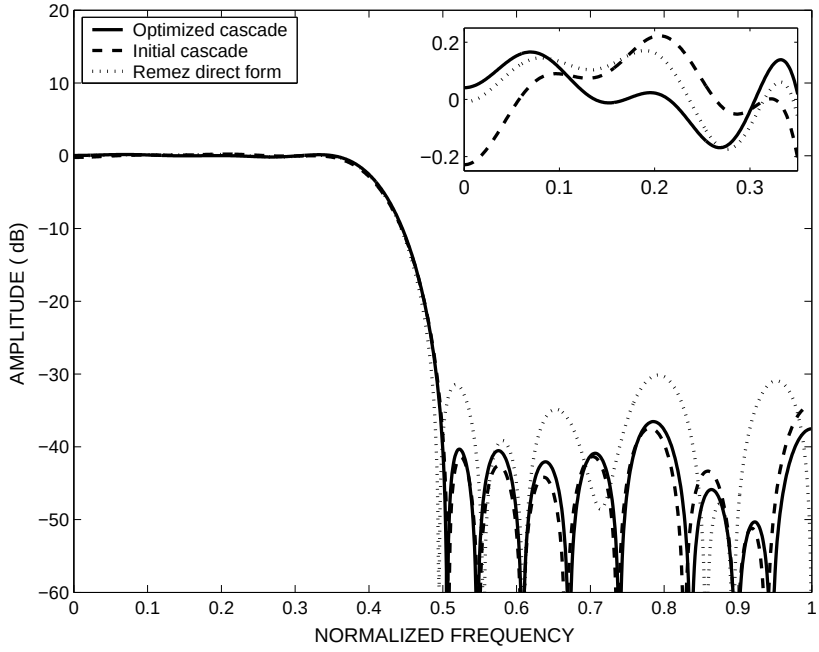
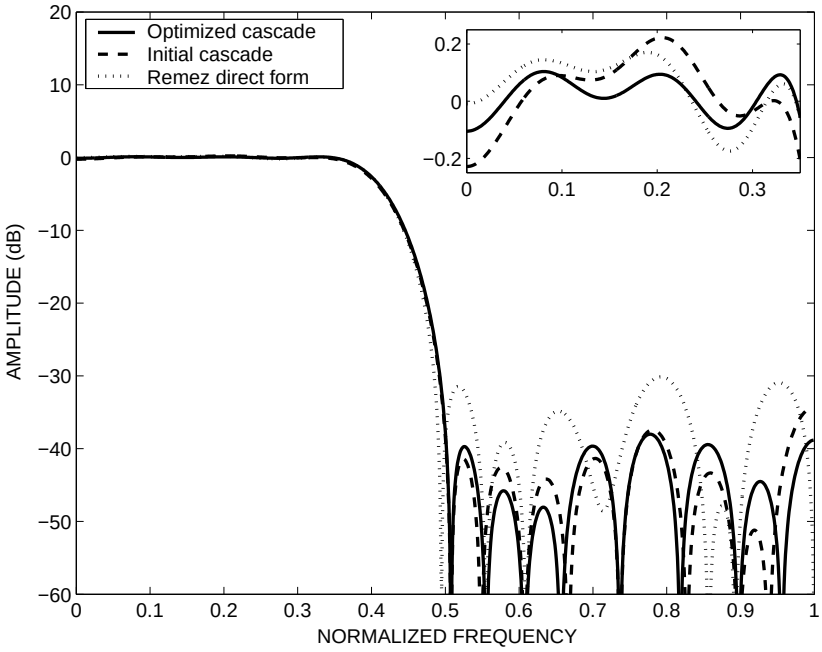


Figure 4.5: Amplitude response of the subfilters and the resulting cascade filter of length 28 obtained by minimizing Eqn. 4.16 (Example 1).

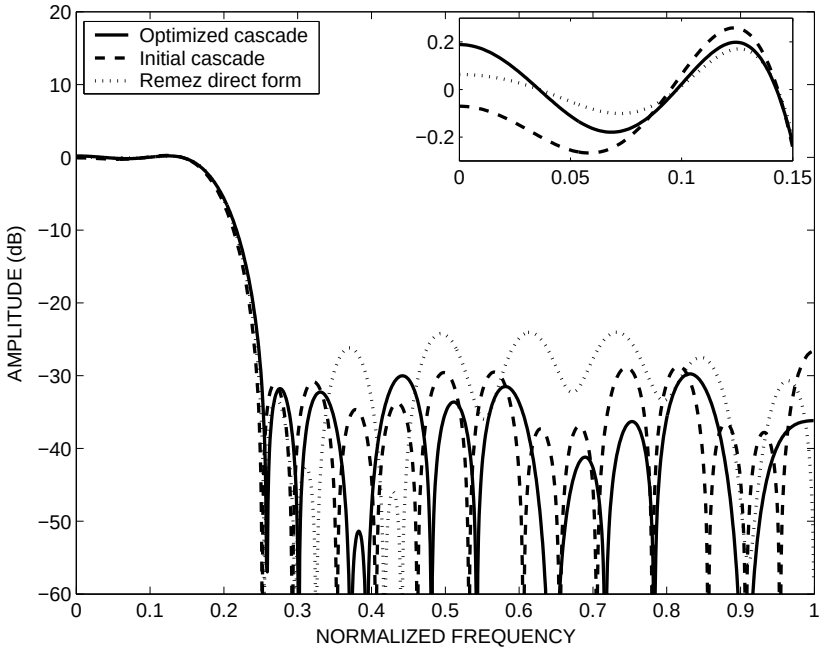


(a)

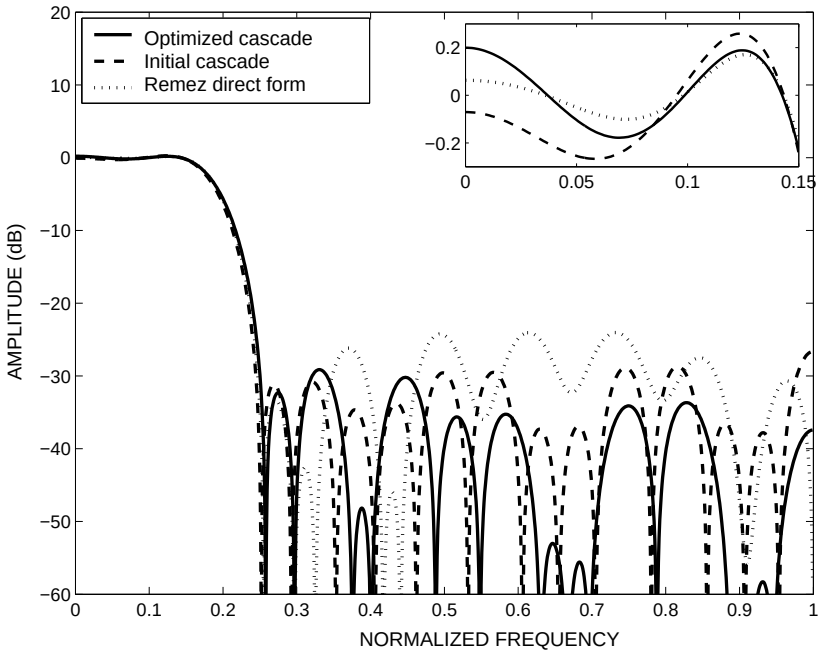


(b)

Figure 4.6: Amplitude response of a cascade filter of length 28 optimized with (a) single and (b) multi-criterion (Example 1).



(a)



(b)

Figure 4.7: Amplitude response of a cascade filter of length 34 optimized with (a) single and (b) multi-criterion (Example 2).

Table 4.2: Results of Design Examples

		Ex. 1	Ex.2
Normalized	ω_p, rad/s	0.35	0.15
band edges	ω_a, rad/s	0.5	0.25
Filter length	N	15+13=28	17+17=34
Initial	δ_p, dB	0.229	0.267
cascade design	δ_a, dB	34.30	26.63
Design using	δ_p, dB	0.169	0.203
single-criterion OGA	δ_a, dB	36.53	29.41
Design using	δ_p, dB	0.105	0.205
multi-criteria OGA	δ_a, dB	37.70	29.13
Remez	δ_p, dB	0.174	0.174
direct-form filter	δ_a, dB	30.15	24.03

4.6 Conclusions

An orthogonal GA approach for the design of cascade-form multiplierless FIR filters has been proposed. The GA approach incorporates the so-called experimental design technique and it works in conjunction with an initial design obtained by using the Remez exchange algorithm. The effects of finite wordlength are minimized by realizing the filter as a cascade of two sections. The algorithm entails a sequential optimization approach whereby each of the two subfilters is optimized in turn until the desired amplitude response is achieved. Two objective functions based on a single and a multiple amplitude-response error criterion have been explored. The simulation results show that the proposed OGA led to improvements in the stopband attenuation

Table 4.3: Coefficients in SOPOT Form Obtained by Minimizing Eqn. 4.16 (Ex. 1)

	Subfilter 1	Subfilter 2
h(0)	$-2^{-5}-2^{-7}$	$2^{-4}-2^{-7}$
h(1)	$-2^{-7}-2^{-10}$	$2^{-8}+2^{-10}$
h(2)	$2^{-5}-2^{-10}$	$-2^{-5}-2^{-10}$
h(3)	$-2^{-7}+2^{-10}$	$-2^{-4}-2^{-7}$
h(4)	$-2^{-5}+2^{-8}$	$2^{-5}-2^{-7}$
h(5)	$2^{-5}-2^{-10}$	$2^{-3}+2^{-5}$
h(6)	$2^{-3}+2^{-5}$	$2^{-2}-2^{-6}$
h(7)	$2^{-5}-2^{-6}$	$2^{-3}+2^{-5}$
h(8)	$2^{-3}+2^{-5}$	$2^{-5}-2^{-7}$
h(9)	$2^{-5}-2^{-10}$	$-2^{-4}-2^{-7}$
h(10)	$-2^{-5}+2^{-8}$	$-2^{-5}-2^{-10}$
h(11)	$-2^{-7}+2^{-10}$	$2^{-8}+2^{-10}$
h(12)	$2^{-5}-2^{-10}$	$2^{-4}-2^{-7}$
h(13)	$-2^{-7}-2^{-10}$	
h(14)	$-2^{-5}-2^{-7}$	

in both Examples 1 and 2. The passband amplitude ripple was also reduced relative to that of the initial cascade filters for both Examples 1 and 2 and the direct-form realization obtained by using the Remez method for Example 1 in the cascade-form FIR filters considered. However, for Example 2, the passband ripple was slightly increased relative to that obtained with the Remez exchange algorithm although the increase is well within the acceptable range. Therefore, the new algorithm is a useful tool for the design of multiplierless FIR filters.

Chapter 5

Design of Asymmetric FIR Filters

5.1 Introduction

In this chapter, a GA for the design of asymmetric FIR filters that would satisfy multiple requirements imposed on the amplitude response and group-delay characteristic is proposed. The GA implements a multiobjective optimization approach for obtaining so-called *Pareto-optimal* solutions for FIR filters. Flexibility is introduced in the design by imposing phase linearity only in the passband instead of the entire baseband as in conventional designs. The proposed GA is a specially tailored *elitist nondominated sorting genetic algorithm* (ENSGA) [80] and it involves a decimal encoding scheme and a multiobjective error formulation based on the amplitude response and passband group delay.

The capability of GAs and other evolutionary algorithms in handling complex problems involving features such as discontinuities and multimodality, their flexibility, as well as their population-based nature make them well suited for achieving multiple objectives [9]. Multiobjective GAs are similar to single-objective GAs in terms of structure but differ in terms of fitness assignment and selection strategies. In the proposed ENSGA approach, three individual objective functions based on the

passband and stopband amplitude-response errors, a measure for the flatness of the group-delay characteristic with respect to passband, and an upper limit on the maximum group delay.

The chapter is organized as follows. In Section 5.2, the optimization problem is formulated. Section 5.3 provides a brief overview of multiobjective optimization. The details regarding the methodology of the ENSGA approach are described in Section 5.4. Design examples and results are presented in Section 5.5 and conclusions are drawn in section 5.6.

5.2 Problem Formulation

The transfer function of an FIR filter of length N is given by

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n} \tag{5.1}$$

Assuming a sampling period T of 1 s, the frequency response can be expressed as

$$\begin{aligned} H(e^{j\omega}) &= \sum_{n=0}^{N-1} h(n) e^{-j\omega n} \\ &= \sum_{n=0}^{N-1} h(n) \cos n\omega + \sum_{n=0}^{N-1} h(n) \sin n\omega \\ &= \mathbf{h}^{\mathrm{T}} \mathbf{c}(\omega) + j \mathbf{h}^{\mathrm{T}} \mathbf{s}(\omega) \end{aligned}$$

where

$$\begin{aligned} \mathbf{h} &= [h(0)\ h(1)\ \cdots\ h(N-1)]^{\mathrm{T}} \\ \mathbf{c} &= [1\ \cos\omega\ \cdots\ \cos(N-1)\omega]^{\mathrm{T}} \\ \mathbf{s} &= [0\ \sin\omega\ \cdots\ \sin(N-1)\omega]^{\mathrm{T}} \end{aligned}$$

Let us consider a lowpass filter with passband and stopband edges ω_p and ω_a, respectively, and let $\omega_s = 2\pi$ be the sampling frequency. The approximation errors

can be sampled uniformly at frequency points ω_k where

$$0 \leq \omega_k \leq \omega_p \quad \text{for} \quad k = 1, \ldots, K_p \quad \text{and}$$

$$\omega_a \leq \omega_k \leq \omega_s/2 \quad \text{for} \quad k = 1, \ldots, K_a$$

The peak pasband error in the amplitude response can be expressed as

$$F_p \;=\; \max \left| 1 - |H(e^{j\omega})| \right| \quad \text{for } 0 \leq \omega \leq \omega_p \tag{5.2}$$

Similarly, a peak-constrained L_2-norm stopband error can be deduced as

$$F_a \;=\; \sum_{k=1}^{K_a} \left| H(e^{j\omega}) \right|^2 \quad \text{for } \omega_a \leq \omega \leq \omega_s/2 \tag{5.3}$$

subject to the constraint

$$\max\, A(\omega) \leq \delta_a \quad \text{for } \omega_a \leq \omega \leq \omega_s/2$$

where

$$A(\omega) = 20 \log \left| H(e^{j\omega}) \right| \tag{5.4}$$

and δ_a represent the amplitude response at frequency ω and the prescribed minimum stopband attenuation in dB, respectively.

An objective function based on the stopband error measure in Eqn. 5.3 can be formulated as

$$F_a^{CLS} \;=\; F_a + \Delta \tag{5.5}$$

where Δ is a *penalty constant* used to limit the peak amplitude error in the stopband, which can be calculated as

$$\Delta = \begin{cases} \delta_a + \max\, A(\omega) & \text{if} \quad \max\, A(\omega) > |\delta_a| \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } \omega_a \leq \omega \leq \omega_s/2$$

To obtain approximately linear phase response with respect to the passband, the group delay, $\tau(\omega)$, should be constant to within some specified tolerance. The

flatness of the group-delay characteristic can be measured in terms of the parameter Q defined in Eqn. 3.10. The minimization of Q with a limit on the maximum group delay imposed as a constraint tends to result in an equiripple delay characteristic.

In the above problem formulation, the design of lowpass FIR filters has been assumed. However, the method can also be used to design highpass, bandpass, bandstop, and other types of filters. Highpass filters can be designed by simply switching the passband and stopband from $0 \leq \omega \leq \omega_p$ and $\omega_a \leq \omega \leq \omega_s/2$ to $\omega_p \leq \omega \leq \omega_s/2$ and $0 \leq \omega \leq \omega_a$, respectively.

Evidently, the design problem just described entails multiple requirements and through the use of a multiobjective strategy, a set of possible alternative solutions can be obtained from which a specific design can be chosen. The next two sections describe the principles of multiobjective optimization and its implementation in terms of a GA for the design of asymmetric FIR filters that would satisfy multiple design requirements.

5.3 Multiobjective Optimization

When there are several, possibly conflicting, objective functions to be optimized simultaneously, usually there is no unique optimal solution. What one can do instead is to find a set of solutions of equivalent quality. These solutions are optimal in the sense that no other solutions in the parameter space are superior to them when all the objective functions are considered. They are known as *Pareto-optimal* solutions [81].

Without loss of generality, a multiobjective optimization can be stated mathe-

matically as the minimization problem

$$\begin{aligned} \text{minimize } \mathbf{y} &= \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x})\ f_2(\mathbf{x})\ \cdots\ f_M(\mathbf{x})]^{\mathrm{T}} \\ \text{where } \mathbf{x} &= [x_1\ x_2\ \cdots\ x_N]^{\mathrm{T}} \in \mathbf{X} \\ \mathbf{y} &= [y_1\ y_2\ \cdots\ y_M]^{\mathrm{T}} \in \mathbf{Y} \end{aligned} \tag{5.6}$$

Vectors $\mathbf{x}$ and $\mathbf{y}$ represent the independent variables and the corresponding values of the individual objective functions, respectively; on the other hand, $\mathbf{X}$ and $\mathbf{Y}$ are called the *solution space* and *objective space*, respectively.

The notion of Pareto-optimality is an important concept for multiobjective optimization and may be explained in terms of a 'dominance relation'. A solution $\mathbf{x}_i$ is said to dominate another solution $\mathbf{x}_j$ if the following conditions hold:

$$\begin{aligned} f_k(\mathbf{x}_i) &\leq f_k(\mathbf{x}_j) \quad \text{for all } k = 1, 2, \ldots, M \\ f_k(\mathbf{x}_i) &< f_k(\mathbf{x}_j) \quad \text{for some } k = 1, 2, \ldots, M \end{aligned} \tag{5.7}$$

If a solution is not dominated by any other solutions when all the objective functions are considered, that solution is said to be a *nondominated solution* [9]. The solutions that are nondominated regarding the entire parameter space are called Pareto-optimal solutions [81]. However, a multiobjective optimization problem may have multiple Pareto-optimal solutions, and these solutions can be regarded as the best compromise solutions. The set of all possible Pareto-optimal solutions constitutes a *Pareto front* in the objective space [9]. A typical Pareto front observed in a mutliobjective optimization problem is illustrated in Fig. 5.1.

5.4 ENSGA for Asymmetric FIR Filters

In this section, we describe an ENSGA that can be used to design asymmetric FIR filters by minimizing the three individual objective functions F_p, F_a^{CLS}, and Q given in Eqns. 5.2, 5.5 and 3.10, respectively, simultaneously.

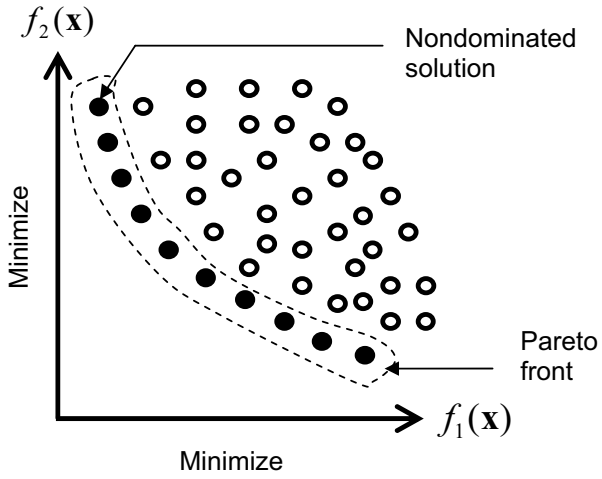


Figure 5.1: Pareto front in a multiobjective optimization problem.

Coefficient vector **h**, which represents the impulse response of the FIR filter, defines the structure of a chromosome for the optimization process. The ENSGA creates new generations by applying ENSGA-specific genetic operators to the individuals of a population. In effect, a scheme that resembles natural selection by means of a multiobjective formulation is adopted through the use of a GA to optimize the FIR filter.

The central idea behind an ENSGA is that a ranking selection method is used to emphasize 'good points' and a mechanism is used to maintain population diversity in the Pareto-optimal front. It is different from a simple GA only in the way the selection operator works. The crossover and mutation operations are based on standard techniques [9]. Values of the objective functions are first calculated for each member of the population. Then the individuals are evaluated in terms of assigned ranking instead of their actual objective fitness values assure nondominance in the evolving solutions. A parameter called *crowding distance* is used to maintain diversity in the poluation.

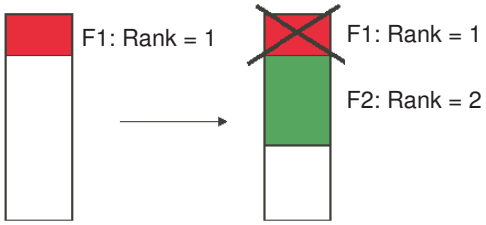


Figure 5.2: Nondominated sorting procedure

In the proposed ENSGA, a ranking technique that orders individuals based on nondomination is used. The nondominated solutions present in the population are first identified from the current population by using the two conditions given in Eqn. 5.7. Then, all these solutions are assumed to constitute the first nondominated level (NDL) in the population and are assigned a rank of 1, and ignored temporarily to process the rest of the population in the same way in order to identify individuals for the second nondominated front as illustrated in Fig. 5.2. The members of the second NDL are sissigned a rank of 2. This process is continued until the entire population is classified into several fronts and all individuals are assigned a rank.

Once the nondominated sorting is complete, a crowding distance is assigned to each individual. The crowding distance is a diversity metric defined as the front density in a specific locale. Solutions located in less crowded space are preferred over those located in high-density space as illustrated in Fig. 5.3. This mechanism prevents quick convergence toward a single solution. The crowding distance is obtained by using the average distance between individuals, which is measured using the value of the objective functions. In Fig. 5.3, the crowding distance of the ith solution is the average side length of he cuboid shown in the dashed box. The normalized *Euclidean distance* is used to measure the side length (distance) of the ith solution from another

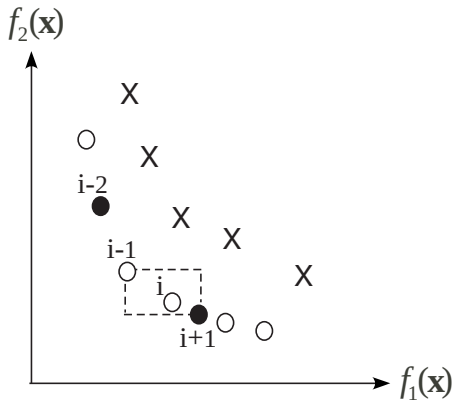


Figure 5.3: Crowding distance measurement; $(i-2)$th solution is preferred over $(i+1)$th solution in the same NDL.

solution $j \neq i$, i.e.,

$$d_{ij} = \sqrt{\sum_{k=1}^{K} \left(\frac{x_k^{(i)} - x_k^{(j)}}{\hat{x}_k - \check{x}_k} \right)}$$

where $\hat{x}_k$ and $\check{x}_k$ represent the upper and lower bounds of variable x_k, respectively, and K is the number of variables.

During the reproduction process, the solutions undergo *binary tournament selection*, crossover, and mutation to produce an offspring population of size M and they evolve from generation to generation until prescribed termination criteria are satisfied. The ENSGA procedure is illustrated in Fig. 5.4.

As in the GA of Chap. 3, the proposed method uses a floating-point representation for chromosome encoding to avoid long binary strings that would be necessary for the problem under consideration. In the ENSGA binary tournament, randomly selected members are compared with dominant members on the basis of nondomination and crowding distance in advancing to the next generation. In this process, the better ranking individuals are selected and the crowding distance is used as a tie breaker. After the offsprings are generated, elitism is introduced by including

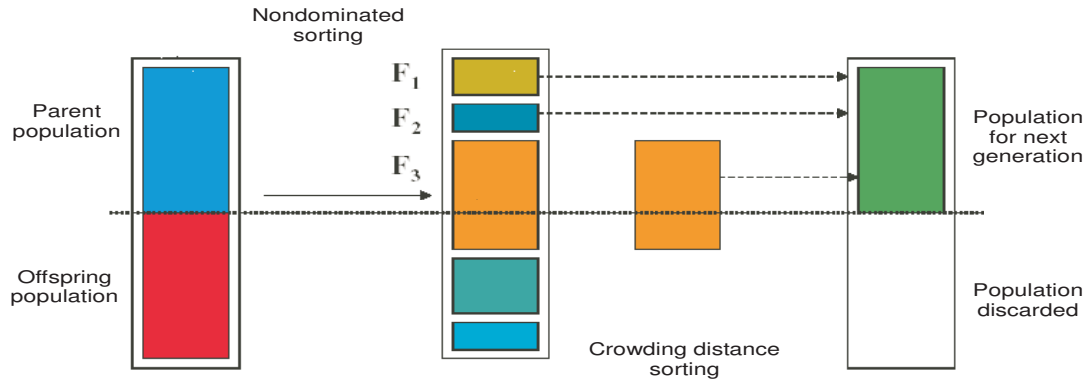


Figure 5.4: ENSGA procedure.

them with their parents to form a population of size $2M$. The new population is again sorted according to nondomination and M members are selected from the population of $2M$ to form a new parent population of M members for the next generation.

In the proposed ENSGA, *simulated binary crossover* (SBX) and a *polynomial mutation* (PM) are applied. The genetic operators, SBX and PM, which simulate the working principles of the traditional single-point crossover and mutation used for binary encoding, are constructed in terms of floating-point representation [66]. The steps incorporated in an ENSGA that can be used to find the best tradeoffs among the objective functions in the design of FIR filters are illustrated in terms of the flowchart in Fig. 5.5.

Simulated Binary Crossover – In SBX, two offspring chromosomes $\mathbf{x}^c(1)$ and $\mathbf{x}^c(2)$ are created from two randomly selected individuals $\mathbf{x}(1) = [x_1\ x_2\ \cdots\ x_N]^T$ and $\mathbf{x}(2) = [x'_1\ x'_2\ \cdots\ x'_N]^T$ using the following procedure:

1) Generate a random number u from a uniform range of numbers $U \in [0, 1]$.

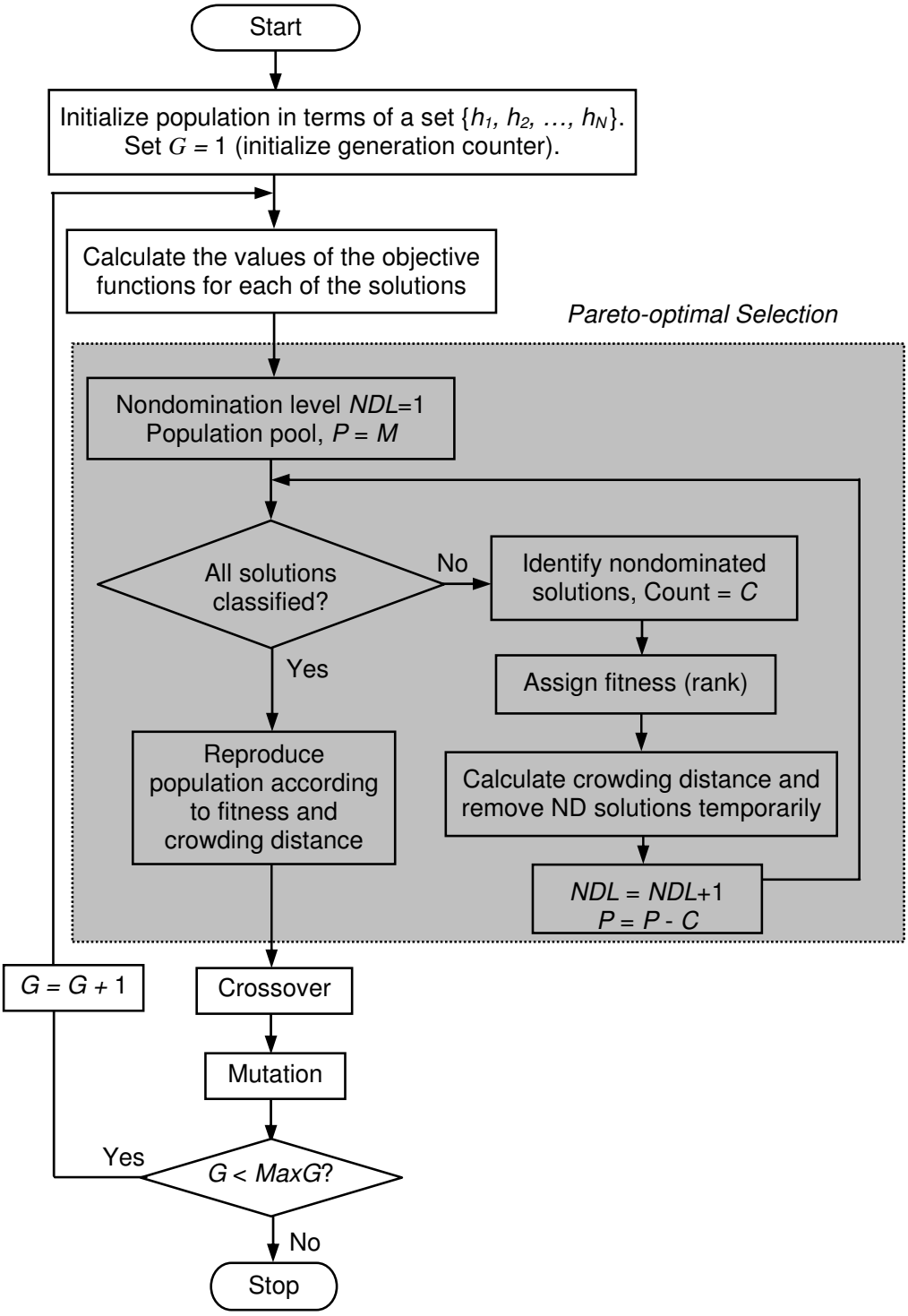


Figure 5.5: Flowchart of multiobjective design of FIR filters using the ENSGA.

2) Calculate a distribution parameter β as

$$\beta = \begin{cases} (2u)^{\frac{1}{\eta+1}} & \text{if } u \leq 0.5 \\ \left[\frac{1}{2(1-u)}\right]^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases}$$

where η is a positive real number. A larger or smaller value of η gives a higher or lower probability of creating near parent offsprings. A value of η in the range of 2 to 5 closely matches the results of SBX with that for the single-point crossover in binary-encoded GAs [82].

3) Create two offsprings:

$$\begin{aligned} x_k^c(1) &= \frac{1}{2}\left[(1-\beta_k)x_k(1) + (1+\beta_k)x_k(2)\right] \\ x_k^c(2) &= \frac{1}{2}\left[(1+\beta_k)x_k(1) + (1-\beta_k)x_k(2)\right] \end{aligned}$$

for $k = 1, 2, \ldots, N$.

Polynomial Mutation –The mutated offspring is calculated as

$$x_k^m = x_k + \delta_k(\hat{x}_k - \check{x}_k) \quad \text{for } k = 1, 2, \ldots, N$$

where $\hat{x}_k$ and $\check{x}_k$ are the upper and lower bounds of x_k, respectively, and δ_k is parameter calculated from a polynomial distribution as

$$\delta_k = \begin{cases} (2r_k)^{\frac{1}{\eta_m+1}} & \text{if } r_k < 0.5 \\ 1 - [2(1-r_k)]^{\frac{1}{\eta_m+1}} & \text{if } r_k \geq 0.5 \end{cases}$$

where r_k is a random number generated from a uniform range of numbers $U \in [0, 1]$ and η_m is a distribution parameter.

The algorithm can be initialized by designing an FIR filter of the required length using one of the standard techniques, such as least-squares (LS) or weighted LS (WLS) techniques [83], [84], whose coefficients can be used as seed. The initial population can be generated by applying random variations to the seed.

5.5 Design Examples and Results

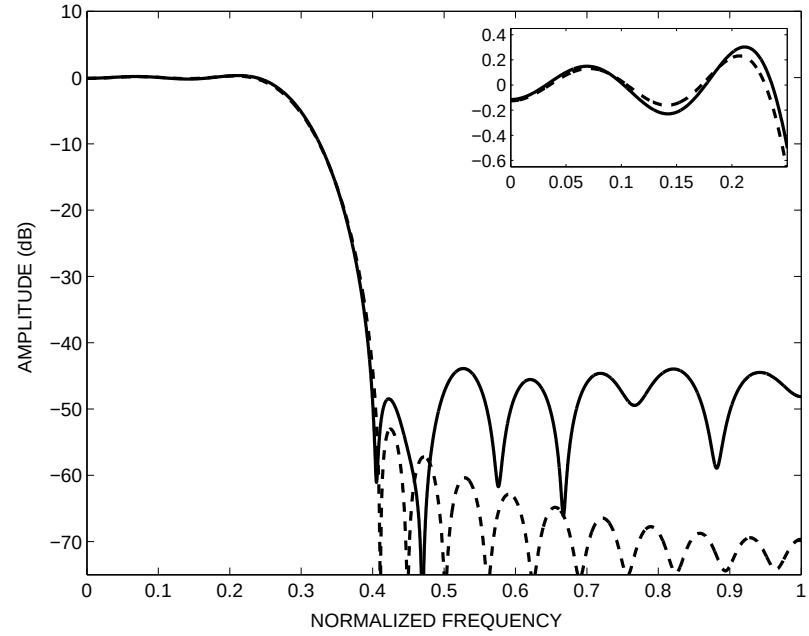
In order to evaluate the performance of the proposed ENSGA and to compare it with standard techniques, it was used to design a lowpass and a highpass filter. For each design, two apparently *good* solutions, i.e., *Solutions a* and *b*, were chosen from the Pareto-optimal set obtained to demonstrate the interesting fact about multiobjective optimization that the user can choose different acceptable solutions to satisfy different combinations of design requirements.

For the designs presented, the WLS solution described in [84] was used as a seed for the initial population. The frequencies of occurrence of crossover and mutation were assumed to be 0.9 and $1/N$, respectively, the population size was taken to be 70, and each design was optimized for 250 ENSGA *generations*.

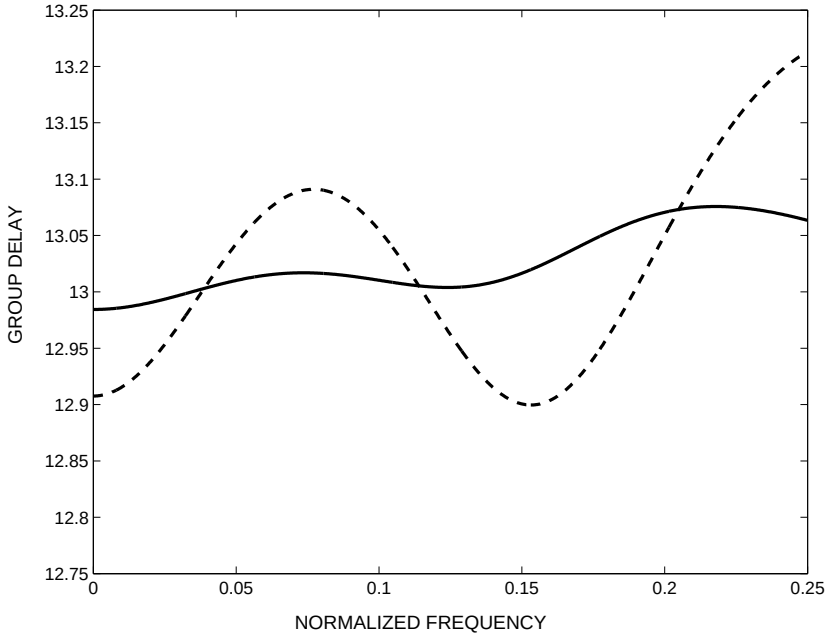
Example 1 – The ENSGA was used to design an asymmetric lowpass FIR filter of length 28. The desired specifications and the results obtained are listed in Table 5.1. Comparisons of *Solutions a* and *b* with that obtained with the WLS method with respect to the amplitude-response and group-delay characteristic are provided in terms of Figs. 5.6 and 5.7, respectively.

From the results we note that *Solution b* is superior to *Solution a* in terms of maximum passband ripple and group-delay linearity whereas the minimum stopband attenuation offered by *Solution b* is slightly lower than that of *Solution a*. Both solutions outperform the WLS design in terms of maximum passband ripple by 55.9% and 59.3%, and group-delay linearity measured in terms of parameter Q by 70.8% and 89.2%, respectively. However, the minimum stopband attenuation obtained with ENSGA is marginally lower by 0.2% and 0.4%, respectively. A 3-D scatter plot of the set of Pareto-optimal solutions obtained is shown in Fig. 5.8.

Example 2 – The ENSGA was used to design an asymmetric highpass FIR filter of length 35. The desired specifications and the results obtained are listed

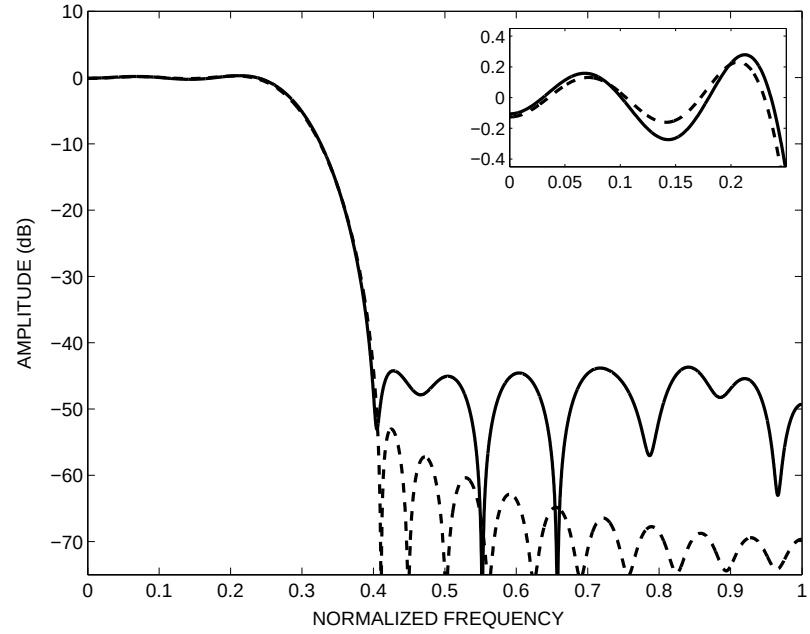


(a)

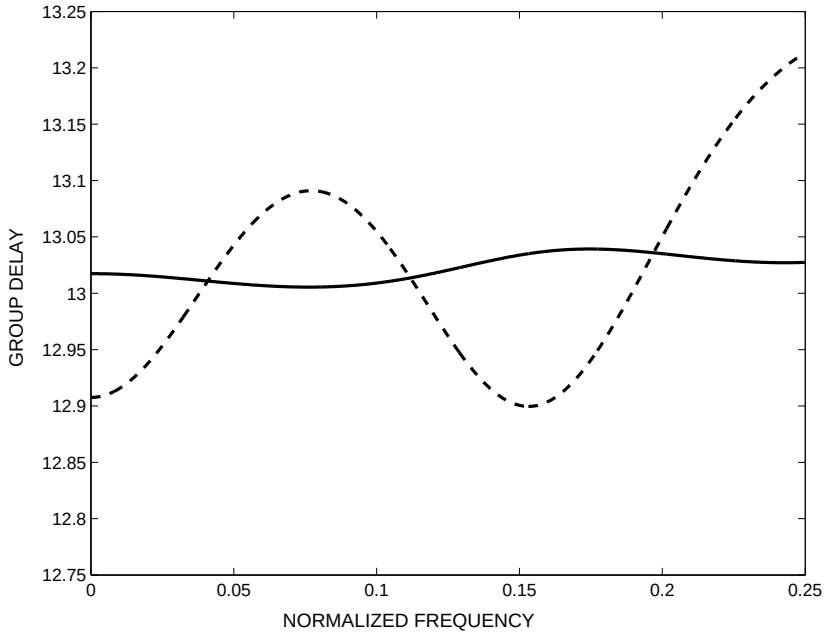


(b)

Figure 5.6: (a) Amplitude response and (b) group-delay characteristic for the lowpass FIR filter designed by using the WLS (dashed curves) and ENSGA (solid curves) methods of *Solution a* (Example 1).



(a)



(b)

Figure 5.7: (a) Amplitude response and (b) group-delay characteristic for the lowpass FIR filter designed by using the WLS (dashed curves) and ENSGA (solid curves) methods of *Solution b* (Example 1).

Table 5.1: Specifications and Results for Design Example 1

	Method	*Solution a*	*Solution b*
Passband edge (rad/s)	0.25		
Stopband edge (rad/s)	0.4		
Filter length, N	28		
Passband ripple (dB)	WLS	0.686	
	ENSGA	0.3025	0.2789
Stopband attenuation (dB)	WLS	44.001	
	ENSGA	43.90	43.823
Passband group-delay (Q)	WLS	1.2002	
	ENSGA	0.3506	0.1298

in Table 5.2. Comparisons of *Solutions a* and *b* with that obtained with the WLS method with respect to the amplitude-response and group-delay characteristic are provided in terms of Figs. 5.9 and 5.10, respectively.

The results show that *Solution b* is superior to *Solution a* when group-delay linearity is the most important design requirement whereas the latter is slightly superior in terms of maximum passband ripple and minimum stopband attenuation. Both solutions outperform the WLS design in terms of maximum passband ripple by 28.4% and 23.6%, stopband attenuation by 2.5% and 1.2%, and the parameter Q (for group delay) by 36.9% and 46.7%, respectively. Fig. 5.11 shows a 3-D scatter plot of the set of Pareto-optimal solutions obtained.

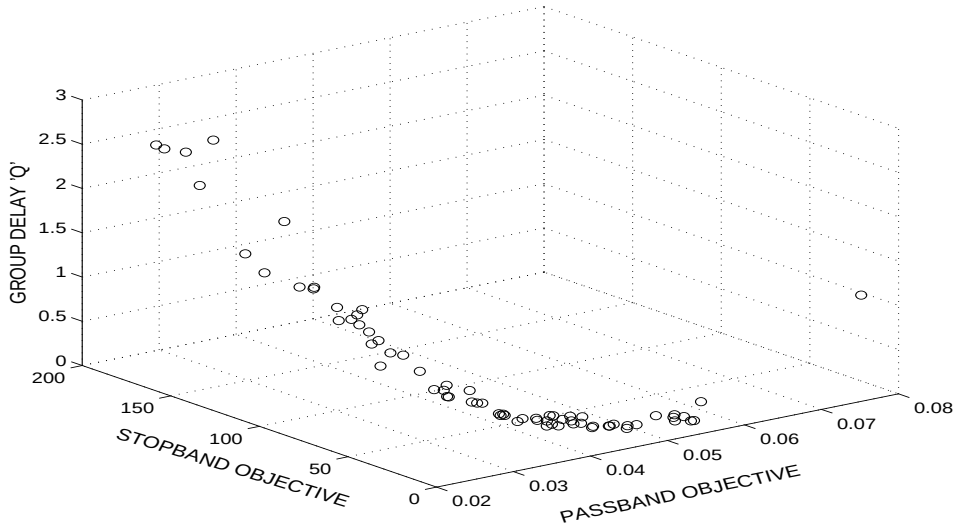
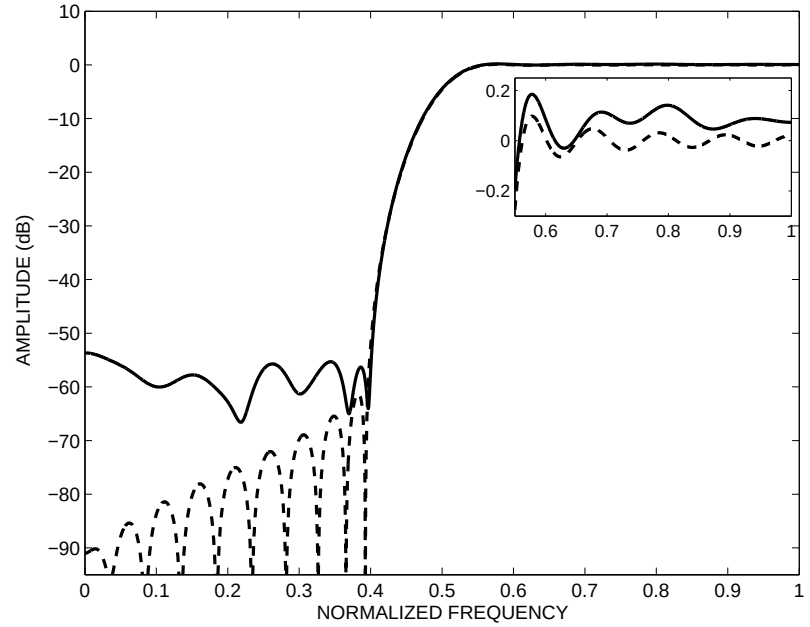


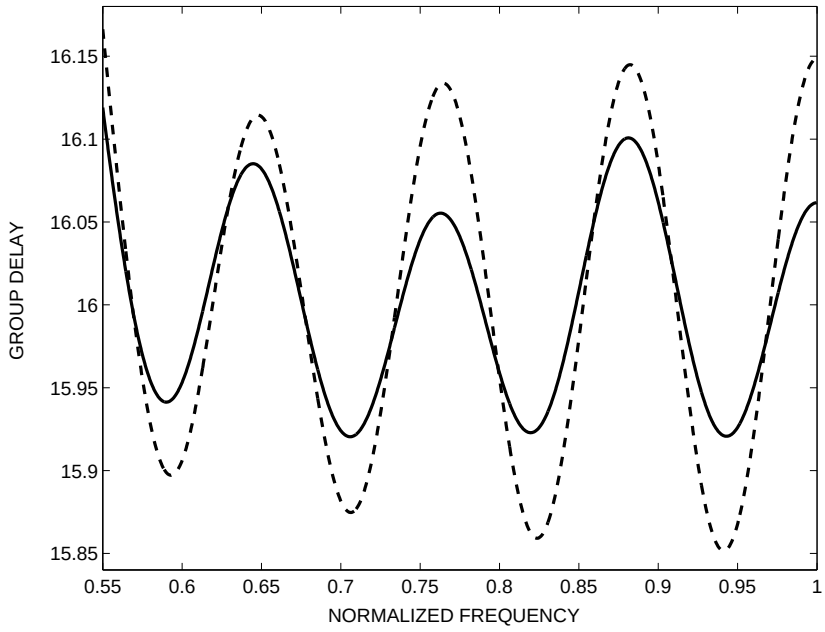
Figure 5.8: 3-D scatter plot of the Pareto-optimal solutions obtained by using the ENSGA (Example 1).

Table 5.2: Specifications and Results for Design Example 2

	Method	*Solution a*	*Solution b*
Stopband edge (rad/s)	0.40		
Passband edge (rad/s)	0.55		
Filter length, N	35		
Maximum Passband ripple (dB)	WLS	0.275	
	ENSGA	0.197	0.210
Minimum stopband attenuation (dB)	WLS	52.376	
	ENSGA	53.670	53.000
Passband group-delay (Q)	WLS	0.984	
	ENSGA	0.621	0.524

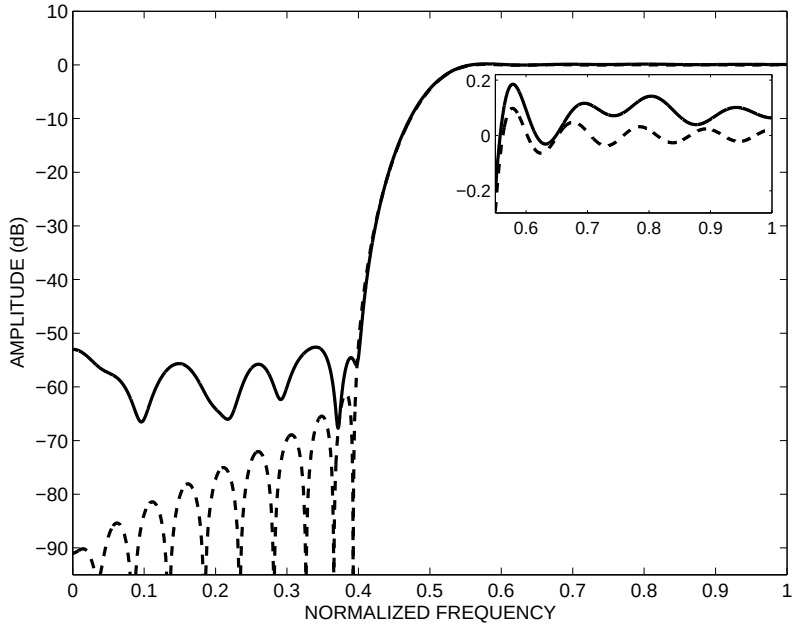


(a)

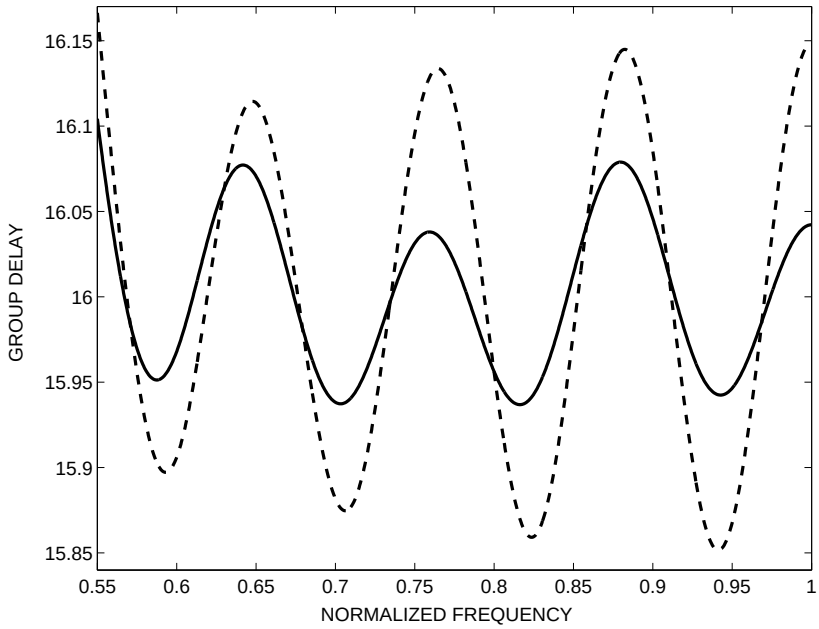


(b)

Figure 5.9: (a) Amplitude response and (b) group-delay characteristic for the highpass FIR filter designed by using the WLS (dashed curves) and ENSGA (solid curves) methods of *Solution a* (Example 2).



(a)



(b)

Figure 5.10: (a) Amplitude response and (b) group-delay characteristic for the highpass FIR filter designed by using the WLS (dashed curves) and ENSGA (solid curves) methods of *Solution b* (Example 2).

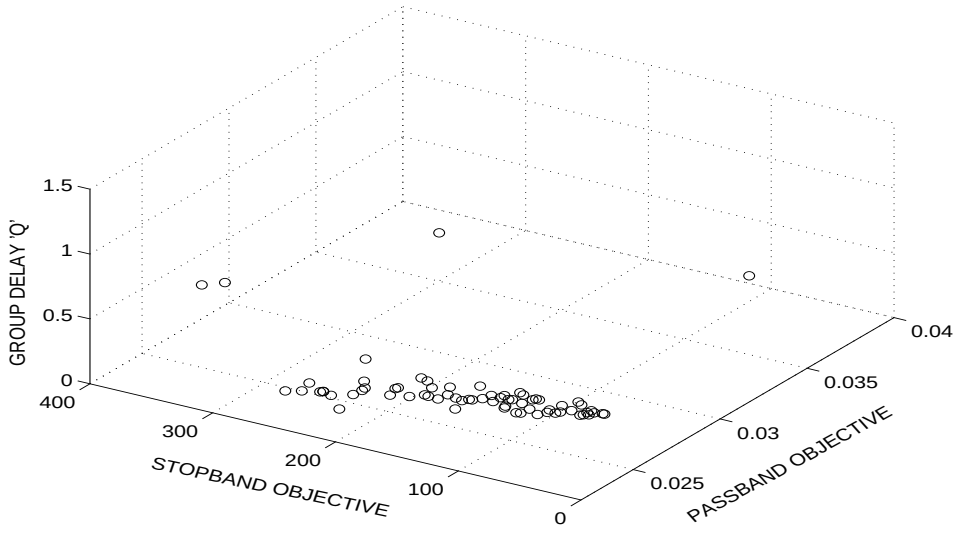


Figure 5.11: 3-D scatter plot of the Pareto-optimal solutions obtained by using the ENSGA (Example 2).

5.6 Conclusions

An ENSGA that can be used for the design of asymmetric FIR filters which would satisfy multiple requirements imposed on the amplitude response and the delay characteristic has been proposed. Three objective functions based on the amplitude-response error and the flatness of the group-delay characteristic have been explored in the design examples considered, and the designs achieved using a state-of-the-art WLS design as seed were compared with the initial WLS design itself. As was shown through simulations, the ENSGA typically led to improvements in the minimum stopband attenuation and reduced the peak passband amplitude-response ripple relative to that in the initial filter. Furthermore, a flatter group-delay passband characteristic was achieved. The multiobjective approach also offers an opportunity of choosing a design that is best-suited to a specific application from the set of Pareto-

optimal solutions obtained. In effect, the proposed algorithm has been shown to be a useful tool for the design of FIR filters with asymmetric coefficients.

Chapter 6

Hybrid Design Approach for IIR Filters

6.1 Introduction

In this chapter, a hybrid genetic algorithm is formulated by using a GA along with a quasi-Newton (QN) algorithm. The proposed algorithm will be referred to hereafter as the *genetic quasi-Newton* (GQN) algorithm. The algorithm combines the flexibility and reliability inherent in the GA with the fast convergence and precision of the QN algorithm. The GA is used as a global search tool to explore different regions in the parameter space whereas the QN algorithm is used to exploit its efficiency in locating local solutions. The GQN algorithm starts with a randomly created initial population of chromosomes which are taken to be strings of decimal-valued coefficients. The values of the coefficients are optimized with the algorithm so as to achieve desired amplitude-response specifications. An L_2-norm error criterion is used as an objective function for the QN optimization, which is also used by the GA for fitness evaluation.

The chapter is organized as follows. In Section 6.2, the problem of designing an IIR filter is briefly introduced. Section 6.3 describes the underlying mechanism and details regarding the methodology of the GQN algorithm. A design example and results are presented in Section 6.4 and conclusions are drawn in Section 6.5.

6.2 IIR Filter Design

An Nth-order IIR filter can be represented by a transfer function of the form

$$H(z) \;=\; H_0 \frac{a_0 + a_1 z + \cdots + a_{N-1} z^{N-1} + z^N}{b_0 + b_1 z + \cdots + b_{N-1} z^{N-1} + z^N} \tag{6.1}$$

The amplitude response of the filter can be expressed as

$$M(\mathbf{x}, \omega) \;=\; |H(e^{j\omega T})| \tag{6.2}$$

where

$$\mathbf{x} \;=\; [a_0 \;\; b_0 \;\; a_1 \;\; b_1 \;\; \cdots \;\; a_{N-1} \;\; b_{N-1} \;\; H_0]^{\mathrm{T}} \tag{6.3}$$

is the filter-coefficient vector and ω is the frequency in rad/s.

The IIR filter can be designed by optimizing the coefficients such that the error function

$$e(\mathbf{x}, \omega) \;=\; M(\mathbf{x}, \omega) - M_d(\omega) \tag{6.4}$$

where $M_d(\omega)$ is the desired amplitude response is minimized. A suitable objective function can be constructed in terms of the L_2-norm of the amplitude-response error as [1]

$$\Psi(\mathbf{x}) \;=\; \left[\sum_{i=1}^{K} |e_i(\mathbf{x})|^2 \right]^{1/2} \tag{6.5}$$

The approximation error is sampled at frequency points ω_i for $i = 1, 2, \ldots, K_1$ in the range $0 \leq \omega_i \leq \omega_p$ and $i = 1, 2, \ldots, K_2$ in the range $\omega_a \leq \omega_i \leq \omega_s/2$ for a lowpass filter where ω_p and ω_a are the passband and stopband edges, respectively, and ω_s is the sampling frequency. The gradient of the objective function, $\Psi(\mathbf{x})$, is given by [1]

$$\bigtriangledown \Psi(\mathbf{x}) \;=\; \left[\sum_{i=1}^{K} |e_i(\mathbf{x})|^2 \right]^{-\frac{1}{2}} \sum_{i=1}^{K} |e_i(\mathbf{x})| \bigtriangledown |e_i(\mathbf{x})| \tag{6.6}$$

6.3 Hybrid Genetic Algorithm

In the proposed GQN algorithm, a GA and a QN algorithm, form the upper- and lower-level elements of the algorithm, respectively. These two levels work in complement with each other to perform broad exploration and local search, respectively. The objective function in Eqn. 6.5 is minimized with respect to the filter coefficients $\mathbf{x}$ in Eqn. 6.3.

The GA has a simple structure and it operates on a small population in order to avoid excessive computation. The connection between the GA and the QN algorithm is made by using the best solution in the current population of the GA as the 'initial guess' for the QN algorithm. When the QN algorithm converges, the optimization routine returns to the GA which is used to obtain more initial guesses for the QN algorithm, and the process is repeated until the termination criterion is satisfied. The interweaving of the exploration by GA and local search by QN is illustared in Fig. 6.1.

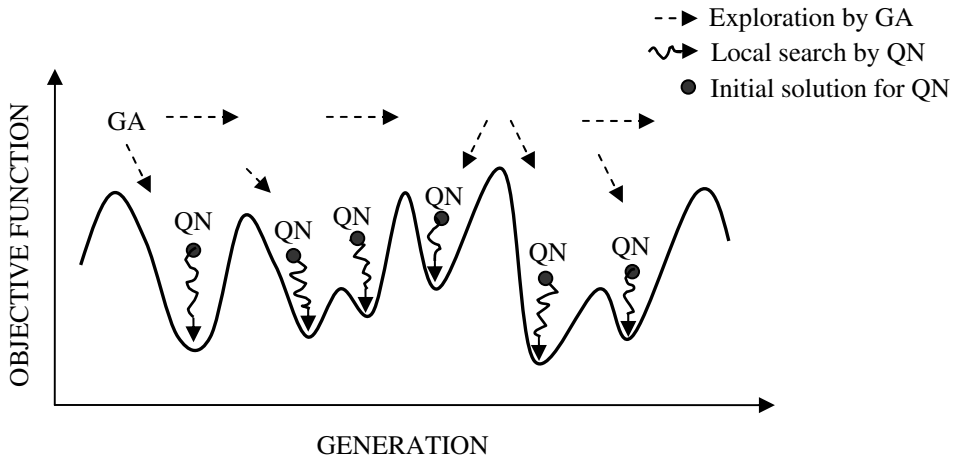


Figure 6.1: Schematic representation of the imterweaving principles of GQN algorithm.

The QN algorithm requires the calculation of the objective function and its

gradient in Eqns. 6.5 and 6.6, respectively. It does not need the Hessian matrix itself but it iteratively generates an approximation for the inverse Hessian. A QN algorithm that is readily applicable to the IIR filter design problem is Algorithm 5 described in Sec. 16.4.4 of [1].

6.3.1 Genetic Algorithm

In the GA process, the coefficient vector $\mathbf{x}$ (Eqn. 6.3) defines the structure of a chromosome for the optimization process. The GA creates new generations by applying genetic operators to the individuals of a population and the fitness of the new individuals is then checked.

For the problem under consideration, the obtained amplitude response characteristic is greatly affected by the precision of filter coefficients. Consequently, encoding the chromosomes would result in very long binary strings. To avoid this difficulty and achieve better precision, a floating-point decimal representation was used. This representation also helps to reduce the computation overhead involved in converting binary into decimal numbers.

The solution evolves from generation to generation through the creation of offsprings through crossovers and mutations. In a crossover operation, a parent chromosome is randomly selected according to a pre-set crossover probability of occurrence P_x. As described in Chap. 3, the algorithm adopts an adaptive perturbation-based crossover technique (see Sec. 3.4). Mutation is applied in a manner similar to that of crossover. A fixed occasional perturbation is applied to randomly chosen coefficients of a chromosome to obtain a mutated offspring. The frequency of mutations is controlled by probability of occurrence P_m. For the problem at hand, crossover plays the major role in improving the solution whereas mutation is essentially treated as a supporting operator to prevent premature convergence. A

relatively high rate in the range of 80 to 90% for crossover and a low rate for muation in the range of 1 to 5%.

In the selection process, chromosomes are ranked on the basis of their fitness. The population that competes for survival comprises the parents as well as the offsprings created during crossovers and mutations. The algorithm then retains a small number of top-ranked solutions as elite solutions and selects a fixed number of best-fit chromosomes as potential parents for the next generation.

The best fitness value in a generation is defined as the minimum of all fitness values for a population and can be expressed as

$$\Delta \;=\; \min\;(\Psi_i) \quad \text{for } i = 1, 2, \ldots, N_p$$

where N_p is the size of the population. If the reduction in Δ in a given generation is less than ε, where ε is a small constant, the generation is deemed to be an 'unproductive generation'. To avoid terminating the GA prematurely, the unproductive-generation count starts after a certain number of generations, say, G_l, is elapsed.

The initial chromosomes for the GA process are created randomly. In each successive generation, half of the chromosomes are taken from the best fits of the previous generation and the rest are generated randomly. The GA is terminated if it fails to improve the best fitness value in a specified number of successive unproductive generations, say, G_c, or if a pre-specified maximum number of generations, say, G_u, is performed.

6.3.2 The GQN Method

The GQN starts with the upper level element, the GA, which operates on a population of potential solutions of the decimal-valued coefficients. A number of best-fit solutions are selected in each generation for the lower level element, the QN algorithm, which

yields *local elite solutions* (LESs). The QN optimization is carried out at the end of a generation only when improvements are made in the solutions through the GA process. Although the good solutions explored through the GA process are used for the QN optimization, the LESs obtained by the QN algorithm are not included in the population of the GA. The basis behind this exclusion is to prevent the GA from getting stuck in a very narrow region of the parameter space. The GA-QN process is continued for a number prespecified generations G_u or until a prespecified number of distinct LESs, say, C_l, is found. We refer to this process as a global cycle (GC). Through this process, the GA gradually produces better initial points for the QN optimization in successive generations.

At the end of a GC, the next GA process is started with a randomly created initial population to proceed with another GC. However, the LESs obtained in each GC are recorded as *global elite solutions* to keep track of all local optima discovered though the optimization process. Eventually, the best solution is selected as the desired solution.

Based on our experiments with the proposed GQN, suitable values for parameters N_p, G_l, G_c, G_u, C_l, and ε for the problem at hand are 26, 25, 6, 45, 6, and 10^{-3}, respectively. Fig. 6.2 illustrates the operation of the GQN algorithm through a flowchart.

6.4 Design Example and Results

To explore the features of the proposed GQN algorithm, it was used to design a variety of lowpass, highpass, and bandpass IIR filters that would satisfy prescribed specifications. One of these designs was a bandpass IIR filter satisfying the following specifications: maximum passband ripple and minimum stopband attenuation of 0.1 and 40 dB; lower and upper stopband edges of 120 and 320 rad/s; lower and upper

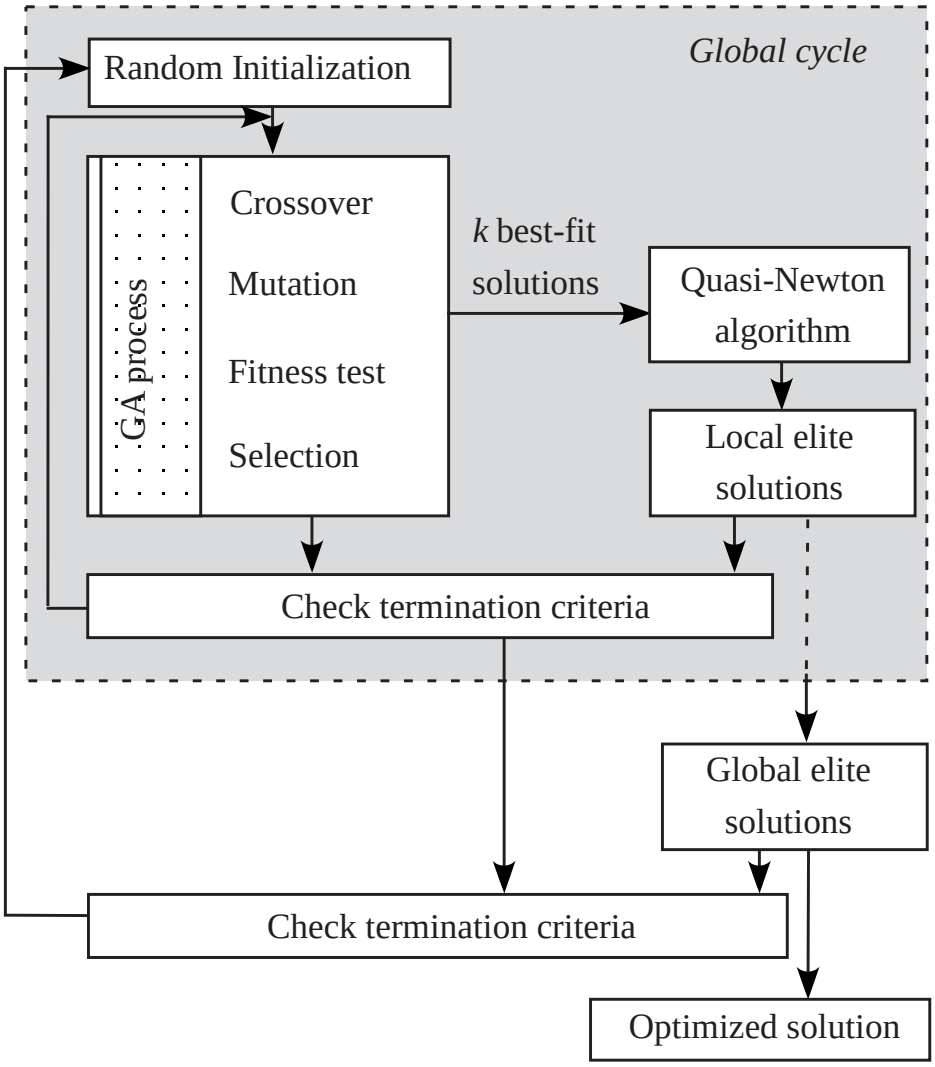
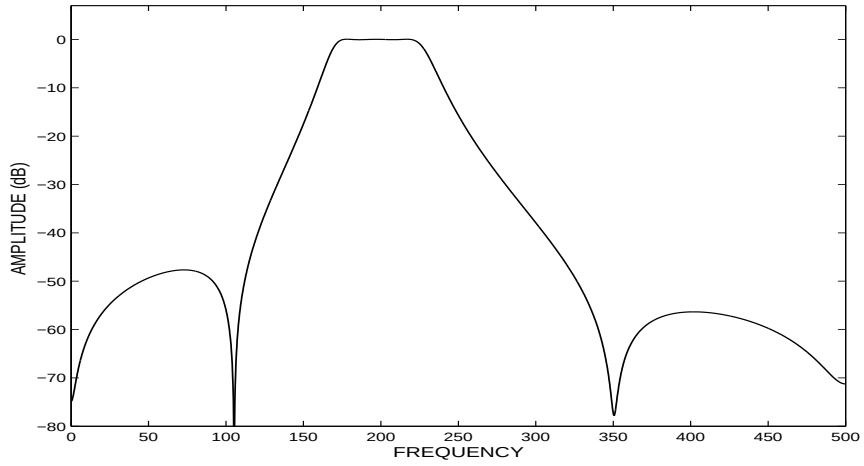


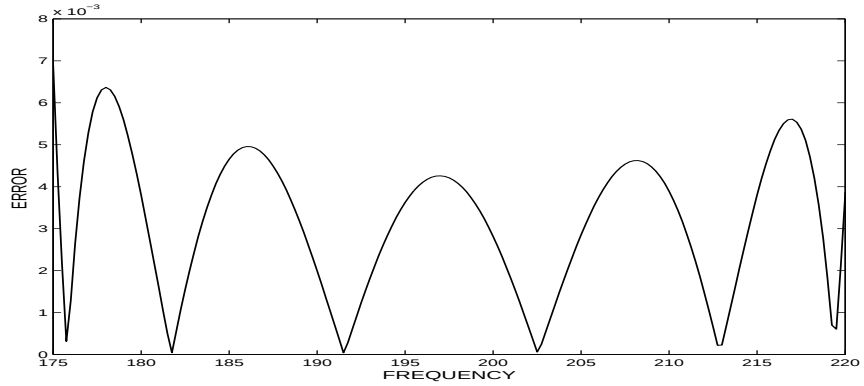
Figure 6.2: Flowchart of the GQN algorithm.

passband edges of 170 and 220 rad/s; and sampling frequency of 1000 rad/s. Crossover and mutation probabilities of 0.9 and 0.05, respectively, were used and the population size was 26. The amplitude response achieved and the corresponding passband and stopband errors are plotted in Fig. 6.3.

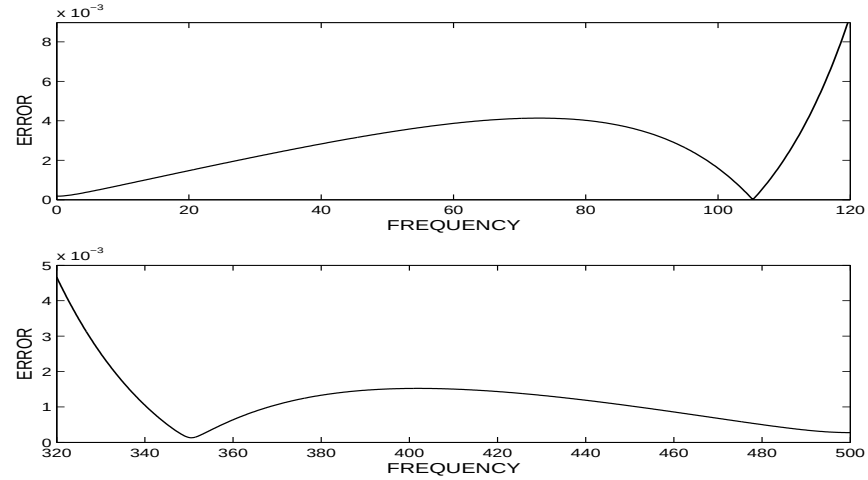
The proposed algorithm was run 50 times using the same specifications and it was found to produce consistent results. The total number of local QN optimizations in



(a)



(b)



(c)

Figure 6.3: Design of a bandpass filter using the GQN algorithm: (a) amplitude response, (b) magnitude of passband error, and (c) magnitude of stopband error.

each run varied from 1 to 32. The QN process plays the major role in the optimization and involves most of the computation whereas the GA process plays a supporting role by locating promising domains of the parameter space. A histogram of the number of QN optimizations required in the 50 runs of the algorithm is plotted in Fig. 6.4.

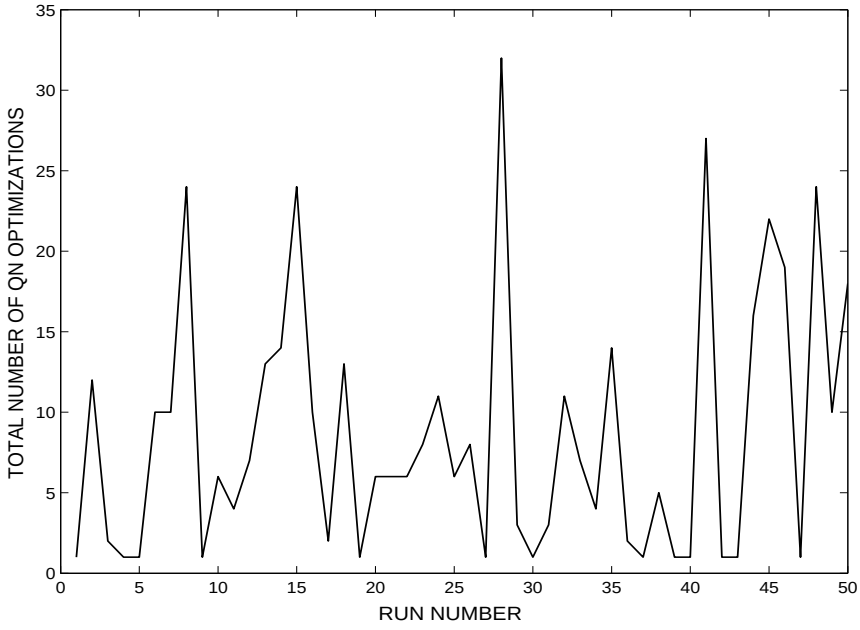


Figure 6.4: Histogram of the number of quasi-Newton optimizations required in the example.

Similar results were obtained for the lowpass and highpass filters.

6.5 Conclusions

A hybrid genetic algorithm has been proposed for the design of IIR digital filters. In the proposed algorithm, the search-based GA approach works in conjunction with the gradient-based quasi-Newton algorithm. The unique combination of broad exploration and local search provided by the two types of algorithms yields a powerful

option to attack multimodal optimization problems such as the design of IIR filters. As shown through experimental results, the GQN algorithm works well with an arbitrary random initialization and it can satisfy prescribed amplitude specifications consistently. Therefore, the new algorithm is a useful tool for the design of IIR filters. The GQN algorithm requires more computation than the quasi-Newton algorithm. However, it is usually more likely to yield a better design because of its ability to explore many promising regions of the parameter space through the operation of the GA.

Chapter 7

Conclusions and Directions for Further Research

The major objective of this dissertation has been to explore the flexible nature of GAs and their ability to obtain more promising solutions in multimodal optimization problems in digital filter design. Various GAs have been used to achieve these goals and the design of several types of digital filters has been investigated. In this chapter, the contributions of this dissertation are summarized and suggestions for further research are presented.

7.1 Conclusions

In Chapter 2, two approaches for the design of fractional-delay (FD) filters based on a GA were developed. In the first approach, the coefficients of an FD FIR filter are determined based on the Farrow structure. In the second approach, the FD filter is designed by using the allpass IIR-based Farrow structure. In both approaches, the designs obtained are free of quantization errors. Simulation results for both approaches were compared with corresponding results achieved with a design based

on a least-squares (LS) method. Results for the first approach have shown that the proposed algorithm leads to reductions in the largest value of maximum amplitude-response and delay errors (with varying FD) in the ranges 27.0-29.47% and 25.01-60.59%, respectively. A reduction in the largest maximum delay errors in the range 26.4-32.8% was achieved in the second approach.

In Chapter 3, a GA-based optimization approach was developed for the design of delay equalizers. In this approach, the equalizer coefficients are optimized using an objective function based on the passband filter-equalizer group delay. The equalizers are built by adding new second-order sections until the desired accuracy in terms of the flatness of the group delay with respect to the passband is achieved. Experimental results have shown that stable delay equalizers satisfying arbitrary prescribed specifications with the desired degree of group-delay flatness can be easily achieved.

In Chapter 4, a GA-based optimization approach for the design of multiplierless FIR filters was presented. A recently-introduced GA, called orthogonal GA (OGA), based on the so-called experimental design technique was exploited to obtain fixed-point implementations of linear-phase FIR filters. In this approach, the effects of finite word length are minimized by considering the filter as a cascade of two sections. Experimental results have shown that the OGA approach leads to improved amplitude response relative to that of an equivalent direct-form cascade filter obtained using the Remez exchange algorithm.

In Chapter 5, a GA for the design of asymmetric FIR filters was proposed. This GA uses a multiobjective approach to obtain so-called Pareto-optimal solutions of the design problem at hand. Flexibility is introduced in the design by imposing phase-response linearity only in the passband instead of the entire baseband as in conventional designs. Three objective functions based on the amplitude-response error and the flatness of the group-delay characteristic have been explored in the

design examples considered. The designs achieved using a state-of-the-art WLS design as seed were compared with the initial WLS design itself. As was shown through simulations, the elitist nondominated-sorting GA (ENSGA) led to improvements in the minimum stopband attenuation and reduced the peak passband amplitude-response ripple relative to that in the initial filter. Furthermore, a flatter group-delay passband characteristic was achieved. The multiobjective approach also offers an opportunity of choosing a design that is best suited to a specific application from the set of Pareto-optimal solutions obtained.

In Chapter 6, a hybrid approach for the design of IIR filters using a GA along with a quasi-Newton (QN) algorithm, referred to as the genetic-quai-Newton (GQN) algorithm was developed. The algorithm combines the flexibility and reliability inherent in the GA with the fast convergence and precision of the QN algorithm. The GA is used as a global search tool to explore different regions in the parameter space whereas the QN algorithm is used to exploit the efficiency of gradient-based algorithms in locating local solutions. Experimental results have shown that the GQN algorithm works well with an arbitrary random initialization and filters that can satisfy prescribed amplitude-response specifications consistently can easily be obtained.

In common with evolutionary algorithms in general, the GAs proposed in this dissertation require a very large amount of computation relative to that required by gradient-based algorithms. However, for non-real time off-line applications this is not a serious drawback since the cost of computation is relatively insignificant nowadays. For real- or quasi-real-time applications, gradient-based algorithms are the only choice.

7.2 Directions for Further Research

Our effort on the design of digital filters using GAs proved to be successful in that we were able to obtain better designs from an engineering viewpoint for a variety of specialized applications. The effort in this area should be continued to develop new GA-based design methodologies, as detailed below.

7.2.1 Use of Structured GA for Fractional Delay Filters

As part of the this dissertation, we proposed a GA for the design of FD FIR filters based on the Farrow structure. In order to design an FD filter with a specified subfilter length, a priori knowledge of the order of the interpolator is required. Otherwise, a lengthy sequential trial-and-error procedure needs to be undertaken. In order to eliminate this drawback, the application of a GA variant known as *structured genetic algorithm* (sGA) should be explored. The sGA postulates a new way of structuring the chromosomes in multilayer form to solve multistage problems in a single evolutionary process [85]. The unique structural form of the sGA also serves to eliminate the problem of premature convergence associated with classical GAs due to loss of diversity, which can restrict the range of the solution space. This approach offers the benefit that both the system structure, i.e., the number of subfilters and corresponding filter coefficients can be simultaneously optimized without extra computational cost and effort.

7.2.2 Design of Cascaded Low-Order Subfilters

Recent advances in *programmable logic devices* (PLDs) and *field programmable gate arrays* (FPGAs) and the in-circuit programmability of these devices can lead to low-cost solutions that offer speed advantage. Beacuse of the limited resources available

in PLDs and FPGAs, the digital filter must be implemented in terms of low-order subfilters [86].

The OGA presented in Chap. 4 can be used to design multiplierless FIR filters that are constructed by cascading two relatively high-order subfilters. To facilitate the use of PLDs and FPGAs, the OGA-based method could be extended to the design of FIR filters realized in terms of cascade connections of low-order (e.g., second- and fourth-order) subfilters.

7.2.3 Minimum-Order Asymmetric FIR Filter Design

Although empirical formulas exist for predicting the required filter order for linear-phase FIR filters [1], achieving the minimum order for asymmetric FIR filters usually requires that several filters of different orders be designed in sequence. This problem can be overcome by formulating an approach for asymmetric FIR filters that would yield the lowest-order filter satisfying the required specifications. The hierarchal structure of chromosome coding found in the sGA can be exploited to create a population of chromosomes (coefficient vectors) of variable lengths. The candidate solutions can be expressed in terms of two-level chromosomes with higher level control genes and subordinate level genes. The filter length can be determined by a number of control genes present in any chromosome that would dictate the activation status of corresponding coefficient genes.

7.2.4 Design of Frequency-Response Masking Filters

The frequency-response masking (FRM) technique is an efficient approach for the design of digital filters. FRM digital filters are realized by cascading interpolated *bandedge shaping* and *masking* subfilters to achieve very steep transition bands. Filters designed with this technique have very sparse coefficients and hence are

economical when compared with other types of filters [87]. By designing multistage FRM filters, the implementation complexity can be further reduced and narrower transition bands can be achieved [88]. However, these filters are very sensitive to quantization effects. Constraints could easily be incorporated in GAs to achieve acceptable sensitivity to filter coefficients. Moreover, GAs can be used to obtain multiplierless FRM filters by using sums of powers of two or canonic signed digit coefficients. In the past, GAs have been used to design single-stage FIR FRM filters [89–91]. This work can be extended to the design of multistage FRM FIR and IIR filters.

It is well known that the throughput in FRM filters depends on the longest subfilter, namely, the band edge shaping filter. To improve the throughput of the overall system, a modified FRM structure can be constructed by replacing the long subfilter by two or more cascaded subfilters [92]. The advantages of sGAs could also be explored to optimize FRM structures in terms of the number of subfilters and their orders together with their coefficients without extra effort.

Bibliography

[1] A. Antoniou, *Digital Signal Processing: Signals, Systems, and Filters.* New Jersey: McGraw-Hill, 2005.

[2] T. W. Parks and C. S. Burrus, *Digital Filter Design.* New York: Wiley, 1987.

[3] W.-S. Lu and A. Antoniou, *Two-Dimensional Digital Filters.* New York: Marcel Dekker, 1992.

[4] O. Franzen, H. Blume, and H. Schroder, "FIR filter Design with Spatial and Frequency Design Constraints Using Evolution Strategies," *Singal Processing*, vol. 68, pp. 295–306, August 1998.

[5] N. Karaboga, "Digital IIR filter Design Using Differential Evolution Algorithm," *EURASIP Jour. on Applied Singal Processing*, vol. 2005, no. 8, pp. 1269–1276, 2005.

[6] S. Chen, R. H. Stephanian, and B. L. Luk, "Digital IIR filter Design Using Adaptive Simulated Annealing," *Digital Singal Processing*, vol. 11, no. 3, pp. 241–251, 2001.

[7] A. Youunis, J. Gu, Z. Dong, and G. Li, "Trends, Features, and Test of Common and Recently Introduced Global Optimization Methods," in *Proceedings of the*

12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference,, vol. 1, Victoria, BC, September 2008, pp. 1–31.

[8] A. Zhigljavsky and A. ilinskas, *Stochastic Global Optimization.* New York: Springer, 2008.

[9] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning.* San Francisco: Addison-Wesley, 1989.

[10] M. Gen, R. Cheng, and L. Lin, *Network Models and Optimization: Multiobjective Genetic Algorithm Approach.* London, U.K.: Springer-Verlag, 1999.

[11] Y. Wang and C. Dang, "An Evolutionary Algorithm for Global Optimization Based on Level-Set Evolution and Latin Squares," *IEEE Trans. on Evolutionary Computation*, vol. 11, no. 5, pp. 579–595, October 2007.

[12] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design.* New York: Wiley, 1997.

[13] K. Deb, "Genetic algorithm optimization applied to electromagnetics: a review," *IEEE Trans. on Antennas and Propagation*, vol. 45, no. 3, pp. 343–353, March 1997.

[14] A. Ebrahimi and V. Tabatabavakili, "Solving multi-path time delay estimation problem in the presence of additive white gaussian noise using a genetic-algorithm," in *Proceedings of the IFIP International Conference on Wireless and Optical Communications Networks*, Singapore, July 2007, pp. 1–5.

[15] O. Myr, T. Ahola, and K. Leivisk, "Time delay estimation and variable grouping Using genetic algorithms," University of Oulu, Finland, Control Engineering Laboratory, Technical Report A No. 32, 2006. [Online]. Available: http://herkules.oulu.fi/isbn9514282973/isbn9514282973.pdf.

[16] A. S. Abutaleb and M. Kamel, "A genetic algorithm for the estimation of ridges in fingerprints," *IEEE Trans. on Image Processing*, vol. 8, no. 8, pp. 1134 –1139, August 1999.

[17] S. Kwong, Q. H. He, K. F. Man, K. S. Tang, and C. W. Chau, "Parallel genetic-based hybrid pattern matching algorithm for isolated word recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 12, no. 5, pp. 573–594, August 1998.

[18] S. Kwong, C. W. Chau, and W. A. Halang, "Genetic algorithm for optimizing the nonlinear time alignment of automatic speech recognition systems," *IEEE Trans. on Industrial Electronics*, vol. 43, no. 5, pp. 559–566, October 1996.

[19] J. Pittman and C. A. Murthy, "Fitting optimal picewise linear functions using genetic algorithms," *IEEE Tarns. on Pattern Analysis Machine Intelligence*, vol. 22, no. 7, pp. 701–718, July 2000.

[20] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press, 1975.

[21] K. D. Jong, "Hybrid methods using genetic algorithms for global optimization," *IEEE Trans. Systems Man Cybernatics*, vol. 10, no. 9, pp. 566 –574, September 1980.

[22] D. Dumitrescu, B. Lazzerni, L. Jain, and A. Dumitrescu, *Evolutionary Computation.* Boca Raton, FL: CRC Press, 2000.

[23] M. Mitchell, *An Introduction to Genetic Algorithms.* Cambridge, MA: MIT Press, 1996.

[24] J. M. Johnson and V. Rahmat-Samii, "Genetic algorithms in engineering electromagnetics," *IEEE Antennas and Propagation Magazine*, vol. 39, no. 4, pp. 1045–9243, August 1997.

[25] R. Nambiar, C. K. Tang, and P. Mars, "Genetic and learning automata algorithms for adaptive filter design," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 50, Atlanta, GA, March 1981, pp. 1253–1256.

[26] K. S. Tang, K. H. Man, S. Kwong, and Q. He, "Genetic algorithms and their applications," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 22–37, November 1996.

[27] N. E. Mastorakis, I. F. Gonos, and M. S. Swamy, "Design of two-dimensional recursive filters using genetic algorithms," *IEEE Trans. on Circuits and Systems I*, vol. 50, no. 5, pp. 634–639, May 2003.

[28] T. I. Laakso, V. Valimki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay - tools for fractional delay filter design," *IEEE Signal Processing Magazine*, vol. 13, no. 1, pp. 30–60, January 1996.

[29] C. W. Farrow, "A continuously variable digital delay element," in *Proceedings of the IEEE International Symposium Circuits Systems*, vol. 3, Espoo, Finland, June 1988, pp. 2641–2645.

[30] C. K. S. Pun, Y. C. Wu, C. C. Chan, and K. L. Ho, "On the design and efficient Implementation of the Farrow Structure," *IEEE Signal Processing Letters*, vol. 10, no. 7, pp. 189–192, July 2003.

[31] H. Johansson and P. Lowenborg, "On the design of adjustable fractional delay FIR filters," *IEEE Trans. on Circuits Systems II*, vol. 50, no. 4, pp. 164–169, April 2003.

[32] M. Makundi, T. I. Laakso, and V. Valimaki, "Efficient tunable IIR and allpass filter structures," *Electronics Letters*, vol. 37, no. 6, pp. 344–345, March 2001.

[33] C. C. Tseng, "Design of 1-D and 2-D variable fractional delay allpass filters using weighted least-squares method," *IEEE Trans. on Circuits Systems I*, vol. 49, no. 10, pp. 1413–1422, October 2002.

[34] J. Yli-Kaakinen and T. Saramki, "An algorithm for the optimization of adjustable fractional-delay all-pass filters," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, Vancouver, BC, May 2004, pp. III–153–III–156.

[35] T. Inukai, "A unified approach to optimal recursive digital filter design," *IEEE Trans. on Circuits and Systems*, vol. CAS-27, no. 7, pp. 646–649, July 1980.

[36] G. Cortelazzo and M. R. Lightner, "Simultaneous design in both magnitude and group-delay of IIR and FIR filters based on multiple criterion optimization," *IEEE Trans. on Circuits and Systems*, vol. 32, no. 5, pp. 949–967, October 1984.

[37] J. L. Sullivan and J. W. Adams, "PCLS IIR filters with simultaneous frequency response magnitude and phase related specifications," in *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers*, vol. 1, Pacific Grove, California, November 1997, pp. 705–709.

[38] M. C. Lang, "Least-squares design IIR filters with prescribed magnitude and phase response and a pole radius constraint," *IEEE Trans. Signal Processing*, vol. 48, no. 11, pp. 3109–3121, November 2000.

[39] W. Lertniphonphun and J. H. McClellan, "Unified design algorithm for complex FIR and IIR filters," in *Proceedings of the IEEE International Conference on Accoustics, Speech and Signal Processing*, vol. 6, Salt Lake City, Utah, May 2001, pp. 3801–3804.

[40] X. Zhang and H. Iwakura, "Design of IIR digital allpass filters based on eigenvalue problem," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 47, no. 2, pp. 554–559, February 1999.

[41] M. Lang and T. I. Laakso, "Simple and robust method for the design of allpass filters using least-squares phase error criterion," *IEEE Tarns. on Circuits and Systems II*, vol. 41, no. 1, pp. 40–48, January 1994.

[42] M. Ikehara, M. Funaishi, and H. Kuroda, "Design of complex all-pass networks using Remez algorithm," *IEEE Tarns. on Circuits and Systems II*, vol. 39, no. 8, pp. 549–556, January 1992.

[43] Z. Jing, "A New Method for Digital Allpass Filter Design," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 11, pp. 1557–1564, November 1987.

[44] V. Sreeram and P. Agathoklis, "Design of linear-phase IIR filters via impulse-response gramians," *IEEE Trans. on Singal Processing*, vol. 40, no. 2, pp. 389–394, February 1992.

[45] S. C. Peng, B. S. Chen, and B. W. Chiou, "Simultaneous design in both magnitude and group-delay of IIR and FIR filters based on multiple criterion optimization," *IEE Proceedings G*, vol. 139, no. 5, pp. 586–590, October 1992.

[46] B. Beliczynski, I. Kale, and G. D. Cain, "Approximation of FIR and IIR digital filters: an algorithm based on balanced model reduction," *IEEE Trans. on Singal Processing*, vol. 40, no. 3, pp. 532–542, March 1992.

[47] A. G. Deckzy, "Synthesis of recursive digital filters using the minimum p-error criterion," *IEEE Trans. on Audio and Electroacoustics*, vol. 20, no. 4, pp. 257–263, October 1972.

[48] A. G. Deckzy, "Equiripple and minimax (Chebyshev) approximations for recursive digital filters," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 22, no. 2, pp. 98–111, April 1974.

[49] C. Charalambous and A. Antoniou, "Equalization of recursive digital filters," *IEE Proceedings - G*, vol. 127, pp. 219–225, October 1980.

[50] A. Antoniou and R. Howell, "Design of phase equalizers for recursive filters," in *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, vol. 1, Victoria, BC, May 1983, pp. 104–107.

[51] N. Ko, D. Shpak, and A. Antoniou, "Design of delay equalizers using constrained optimization," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, vol. 1, Victoria, BC, August 1997, pp. 173–177.

[52] A. Yurdakul and G. Dundar, "Multiplierless realization of linear DSP transforms by using common two-term expressions," *Jour. VLSI Signal Processing*, vol. 22, no. 3, pp. 163–172, September 1999.

[53] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Trans. on Circuits Systems*, vol. 36, no. 7, pp. 1044–1047, July 1989.

[54] P. Gentili, F. Piazza, and A. Uncini, "Efficient genetic algorithm design for power-of-two FIR filters," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, Detroit, MI, May 1995, pp. 1268–1271.

[55] M. Yagyu, A. Nishihara, and N. Fujii, "Fast FIR digital filter structures using minimal number of adders and its application to filter design," *IEICE Trans. on Fundamentals*, vol. E79-A, no. 8, pp. 1120–1128, August 1996.

[56] Y. C. Lim, S. R. Parker, and A. G. Constantinides, "Finite wordlength FIR filter design using integer programming over a discrete coefficient space," *IEEE Trans. on Acoustics, Speech Signal Processing*, vol. ASSP-30, pp. 661–664, August 1982.

[57] Q. Zhao and Y. Tadokoro, "A simple design of FIR filters with power-of-two coefficients," *IEEE Trans. on Circuits Systems*, vol. 35, no. 5, pp. 566–570, May 1988.

[58] Y. C. Lim and S. R. Parker, "Discrete coefficient FIR digital filter design based upon LMS criteria," *IEEE Trans. on Circuits Systems*, vol. CAS-30, pp. 723–739, October 1983.

[59] R. Cemes and D. A. Boudaoud, "Genetic approach to design of multiplier-less FIR filters," *Electronics Letters*, vol. 29, no. 3, pp. 2090–2091, November 1993.

[60] Y. Leung and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Trans. on Evolutionary Computation*, vol. 5, no. 1, pp. 41–53, February 2001.

[61] Q. Zhang and Y. W. Leung, "An orthogonal genetic algorithm for multimedia multicast routing," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 1, pp. 53–62, April 1999.

[62] Y. C. Lim, "Design of discrete-coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude," *IEEE Trans. on Circuits Systems*, vol. 37, no. 12, pp. 1480–1486, December 1990.

[63] M. C. Lang, "An iterative reweighted least-squares algorithm for constraint design on nonlinear phase FIR filters," in *Proceedings of the IEEE International Symposium Circuits Systems*, vol. 5, Seattle, WA, May 1995, pp. 367–370.

[64] M. Lang and J. Bamberger, "Nonlinear phase FIR filter design according to the L_2 norm with constraints for the complex error," *Signal Processing*, vol. 36, no. 1, pp. 259–268, March 1994.

[65] J. W. Adams, "FIR digital filters with least-squares stopbands subject to peak-gain constraints," *IEEE Trans. on Circuits Systems*, vol. 39, no. 4, pp. 376–388, April 1991.

[66] N. Srinivas and K. Deb, "Multi-objective function optimization using non-dominated sorting genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[67] W.-S. Lu and A. Antoniou, "Design of digital filters and filter banks by optimization: a state of the art review," in *Proceedings of European Signal Processing Conference on Communications*, vol. 1, Tampere, Finland, September 2000, pp. 351–354.

[68] A. Tarczynski, G. D. Cain, E. Hermanowicz, and M. Rojewski, "A WISE method for designing IIR filters," *IEEE Trans. on Signal Processing*, vol. 49, no. 7, pp. 1421–1432, July 2001.

[69] W.-S. Lu, S.-C. Pei, and C.-C. Tseng, "A weighted least squares method for the design of 1-D and 2-D IIR filters," *IEEE Trans. Signal Processing*, vol. 46, no. 1, pp. 1–10, January 1998.

[70] A. T. Chottera and G. A. Jullien, "A linear programming approach to recursive digital filter design with linear phase," *IEEE Trans. Circuits and Systems*, vol. CAS-29, no. 3, pp. 139–149, March 1982.

[71] K. S. Tang, K. F. Man, S. Kwong, and Z. F. Liu, "Design and optimization of IIR structure using hierarchical genetic algorithms," *IEEE Trans. Industrial Electronics*, vol. 45, no. 3, pp. 481–487, June 1998.

[72] K. Uesaka and M. Kawamata, "Synthesis of low-sensitivity second-order digital filters using genetic programming with automatically defined functions," *IEEE Signal Processing Letters*, vol. 7, no. 4, pp. 83–85, April 2000.

[73] J. M. Renders and S. P. Flasse, "Hybrid methods using genetic algorithms for global optimization," *IEEE Trans. Systems Man Cyber.-Part B: Cybernatics*, vol. 26, no. 2, pp. 243–258, April 1996.

[74] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete Time Signal Processing, 2nd ed.* New Jersey: Prentice Hall, 1999.

[75] V. Valimki and T. I. Laakso, "Principals of fractional delay filters," in *Proceedings of the IEEE International Conference on Accoustics, Speech and Signal Processing*, vol. 6, Istanbul, Turkey, June 2000, pp. 3870–3873.

[76] H. Zhao and H. K. Kwan, "Design of 1-D stable variable fractional delay IIR filters," *IEEE Tarns. on Circuits and Systems II*, vol. 54, no. 1, pp. 86–90, January 2007.

[77] W.-S. Lu, "Design of recursive digital filters with prescribed stability margin: a parameterization approach," *IEEE Tarns. on Circuits and Systems II*, vol. 45, no. 9, pp. 1289–1298, September 1998.

[78] Y. C. Lim and B. Liu, "Design of cascade form FIR filters with discrete valued coefficients," *IEEE Trans. on Acoustics, Speech, Signal Processing*, vol. 36, no. 11, pp. 1735–1739, November 1988.

[79] D. C. Montgomery, *Design and Analysis of Experiments, 3rd ed.* New York: Wiley, 1991.

[80] K. Deb, S. A. A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

[81] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms - A comparative case study," in *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, vol. V, Amsterdam, September 1998, pp. 292–301.

[82] K. Deb and R. B. Agrawall, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 2, pp. 115–148, 1995.

[83] S. K. Kidambi and R. P. Ramachandran, "Design of nonrecursive filters satisfying arbitrary magnitude and phase specifications using a least-squares approach," *IEEE Tarns. on Circuits Systems II*, vol. 2, no. 11, pp. 711–716, November 1995.

[84] M. Lang, "Algorithms for the Constrained Design of Digital Filters with Arbitrary Magnitude and Phase Responses," Vienna University of Technology, Institute for Communications and Radio-Frequency Engineering,

Ph.D. Thesis CSD-TR No. 030056, 1999. [Online]. Available: http://www.nt.tuwien.ac.at/fileadmin/users/gerhard/diss_Lang.pdf.

[85] D. Dasgupta and D. R. McGregor, "Nonstationary function optimization using the structured genetic algorithm," in *Proceedings of Parallel Problem Solving from Nature Conference*, Brussels, Belgium, Sepember 1992, pp. 145–154.

[86] L. M. Smith, "Decomposition of FIR digital filters for realization via the cascade connections of subfilters," *IEEE Trans. on Singal Processing*, vol. 46, no. 6, pp. 1681–1684, June 1998.

[87] W. R. Lee, L. Caccetta, and K. L. Teo, "A unified approach to multistage frequency-response masking filter design using the WLS technique," *IEEE Trans. on Singal Processing*, vol. 54, no. 9, pp. 3459–3467, September 2002.

[88] W. S. Lu and T. Hinamoto, "Optimal design of IIR frequency-response masking filter design using second order cone programming," *IEEE Trans. on Singal Processing*, vol. 50, no. 11, pp. 1401–1411, November 2003.

[89] M. K. Sai and B.Nowrouzian, "A diversity controlled genetic algorithm for optimization of FRM digital filters over DBNS multiplier coefficient space," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, New Orleans, Louisiana, May 2007, pp. 2331–2334.

[90] P. Mercier and B.Nowrouzian, "Design of FRM digital filters over the CSD multiplier coefficient space employing genetic algorithms," in *Proceedings of the IEEE International Conf. on Accoustics, Speech and Signal Processing*, vol. 3, Toulouse, France, May 2006, pp. 968–971.

[91] L. Cen and Y. Lian, "A hybrid GA for the design of multiplication-free frequency response masking filters," in *Proceedings of the IEEE International Symposium on Circuits and Systems,*, vol. 1, Salt Lake City, Utah, May 2005, pp. 520–523.

[92] W. R. Lee, L. Caccetta, K. L. Teo, and V. Rehbock, "A weighted least squares approach to the design of FIR filters synthesized using the modified frequency-response masking structure," *IEEE Trans. on Ciecuits and Systems II*, vol. 53, no. 5, pp. 379–383, May 2006.